Universität Ulm
Institut für Datenbanken und Informationssysteme
Leiter: Prof. Dr. Peter Dadam

Unterstützung von Periodizität in Informationssystemen

Dissertation
zur Erlangung des Doktorgrades Dr.rer.nat.
der Fakultät für Ingenieurwissenschaften und Informatik
der Universität Ulm

vorgelegt von
Markus Patrick Kalb
aus Gotha
Juli 2010

Amtierender Dekan:	Prof. Dr. Michael Weber
Gutachter:	Prof. Dr. Peter Dadam
	Prof. Dr. Franz Schweiggert
Tag der Promotion:	n.n.

Periodizität: (griech.), die regelmäßige Wiederkehr gewisser Erscheinungen innerhalb bestimmter Zeitgrenzen; der Kreislauf der Natur.

Danksagung

Diese Arbeit entstand zum größten Teil im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Datenbanken und Informationssysteme (DBIS) der Universität Ulm. Den Anstoß zu dieser Arbeit gaben praktische Erfahrungen und Erkenntnisse bei der Entwicklung innovativer Workflow-Konzepte im Institut DBIS. An dieser Stelle möchte ich mich bei einigen Personen bedanken, ohne deren Unterstützung die Anfertigung dieser Arbeit so nicht möglich gewesen wäre.

Mein Dank gilt Herrn Prof. Dr. Peter Dadam, meinen Doktorvater, für seine tatkräftige und engagierte Betreuung und Unterstützung während meiner Promotionszeit in den vergangenen Jahren. Mit seiner unverwechselbaren Art, wissenschaftlichen Neugier und problemorientierter Sichtweise ("*What's the message*?") hat er die Entwicklung dieser Arbeit stets vorangetrieben. Ebenso möchte ich mich bei der Deutschen Forschungsgemeinschaft (DFG) bedanken für deren finanzielle Förderung meiner Arbeit im Projekt "*Periodizität in Informationssystemen*". Bedanken möchte ich mich auch bei Herrn Prof. Dr. Franz Schweiggert für die Übernahme des Zweitgutachtens.

Mein ausdrücklicher Dank gilt meinen Kollegen Prof. Dr. Manfred Reichert und PD Dr. Stefanie Rinderle-Ma, die in unzähligen Diskussionen wertvolle Anregungen gaben und durch ihr akribisches Lesen aller Schriftstücke mich in meiner Arbeit weitergebracht haben. Ohne ihre fortwährende Unterstützung wäre die Fertigstellung dieser Arbeit so nicht möglich gewesen. Gleichermaßen möchte ich Herrn Dr. Daniel Haase des Instituts für Reine Mathematik der Universität Ulm danken. Seine einzigartige Kombination aus mathematischem Sachverstand und nötigen Pragmatismus setzte wesentliche Impulse für die Umsetzung und Untermauerung meiner Ideen. Des Weiteren geht ein herzliches "Dankeschön!" an alle Kollegen und Kolleginnen des Instituts DBIS für das angenehme Arbeitsumfeld und den vielen Spaß den wir gemeinsam hatten. Ebenso bedanke ich mich bei allen Studenten für ihr Engagement, die im Rahmen von Praktika, Diplomarbeiten oder als wissenschaftliche Hilfskraft zu dieser Arbeit beigetragen haben.

Mein größter Dank gebührt meiner Familie und Freunden. Ihr Zuspruch, Rücksichtnahme und Unterstützung durch alle Höhen und Tiefen in den vergangen Jahren, haben die Erstellung dieser Arbeit erst ermöglicht. Insbesondere Danke ich meiner Lebensgefährtin Birgit und meinem Sohn Niklas für ihre Liebe und Verständnis während meiner Promotionszeit.

Ulm, im Juli 2010

Kurzfassung

In vielen Anwendungsgebieten kommt zunehmend der Wunsch auf, temporale Aspekte systemseitig zu unterstützen. Ein solches Gebiet ist die Unterstützung von Geschäftsprozessen durch prozess-orientierte Informationssysteme. Eine große Herausforderung bildet hier die adäquate Handhabung von Geschäftsprozessen, die aus regelmäßig (periodisch) wiederkehrenden elementaren Aktivitäten oder Abfolgen von Aktivitäten bestehen. Die systemseitige Unterstützung solcher periodischen Prozesse bzw. periodischen Spezifikationen (Periodizität) weist Anforderungen auf, die weit über die temporalen Fähigkeiten heutiger Informationssysteme hinausgehen. Gegenstand dieser Arbeit ist die Entwicklung eines ganzheitlichen Ansatzes zur integrierten Unterstützung von Periodizität in (prozessorientierten) Informationssystemen. Das zentrale Anliegen ist hierbei, Konzepte und Modelle zu entwickeln, die als Basistechnologien zur Unterstützung periodischer Zusammenhänge in unternehmensweiten (prozessorientierten) Informationssystemen (auch bei tausenden von Benutzern und aktiven Prozess-Instanzen), wesentlich sein werden.

In dieser Arbeit wird mit dem zyklenorientierten Modell ein logisches (Meta-) Zeitmodell eingeführt, das eine ausdrucksstarke und anwenderorientierte Beschreibung von komplexen periodischen Spezifikationen ermöglicht. Neben einfachen (äquidistanten) periodischen Angaben können auch komplexe (nichtäquidistante) periodische Angaben beschrieben werden. Des Weiteren wird mit dem HSP-Baum eine kompakte und berechenbare Repräsentationsform des (theoretischen) zyklenorientierten Modells vorgestellt. Der HSP-Baum bildet mit seiner präzisen formalen Syntax und Semantik die Grundlage dafür, dass semantisch falsche, inkonsistente oder widersprüchliche Angaben erkannt und korrigiert werden können. Zur Steigerung der temporalen Fähigkeiten heutiger Informationssysteme ist eine operationale Unterstützung unumgänglich, die einen effizienten Umgang mit periodischen Spezifikationen ermöglicht. Mit der in dieser Arbeit entwickelten Bitletrepräsentation wird ein entsprechender Lösungsansatz vorgestellt. Die Bitletrepräsentation begegnet dem Hauptproblem bestehender Realisierungsverfahren, das zu einem starken Anstieg sowohl des operationalen Aufwands als auch des Repräsentationsumfangs führte. Darüber hinaus kann die Bitletrepräsentation als eine Form der (verlustfreien) Komprimierung von Wiederholungsmustern verstanden werden, was zu äußerst kompakten Repräsentationen periodischer Spezifikationen führen kann.

Insgesamt erlauben die entwickelten Lösungen und Algorithmen eine durchgängige Unterstützung periodischer Zusammenhänge. Auf Basis prototypischer Realisierungen können wir ebenfalls zeigen, dass die Skalierbarkeit und Effizienz der Lösungen auch bei großem Datenvolumen gewährleistet wird.

Inhaltsverzeichnis

I Einleitung, Anforderungen und bestehende Lösungsansätze

1 Einleitung **3**
- 1.1 Temporale Aspekte in prozessorientierten Informationssystemen 3
- 1.2 Periodizität in Informationssystemen 4
  - 1.2.1 Umfassende Spezifikation von Periodizität 5
  - 1.2.2 Operationale Unterstützung periodischer Spezifikationen 6
- 1.3 Ziele und Beitrag der Arbeit 8
- 1.4 Aufbau der Arbeit 9

2 Anforderungen, Problemstellungen und Stand der Technik **11**
- 2.1 Grundlegende Begriffe und Anforderungen 11
  - 2.1.1 Klassen von Periodizitätsschemas 12
  - 2.1.2 Ausprägungen von quantitativen exakten Periodizitätsschemas . 13
  - 2.1.3 Operationale Unterstützung 16
- 2.2 Ansätze zur Beschreibung periodischer Spezifikationen 18
  - 2.2.1 Algebraische Ansätze 19
  - 2.2.2 Constraint-basierte Ansätze 21
- 2.3 Realisierungsansätze der operationalen Unterstützung 23
  - 2.3.1 Teilmaterialisierung 23
  - 2.3.2 Verwendung der formalen Beschreibung 25
  - 2.3.3 Musterbasierte Realisierungsformen 29
- 2.4 Bewertung der bestehenden Lösungsansätze 34
  - 2.4.1 Umfassende Beschreibung 34
  - 2.4.2 Operationale Unterstützung 35
  - 2.4.3 Effiziente interne Repräsentation 35

II Beschreibung von periodischen Spezifikationen

3 Ein Beschreibungsmodell für Periodizitätsschemas **39**
- 3.1 Anforderungen und Entwurfsziele 39
  - 3.1.1 Verständlichkeit 39
  - 3.1.2 Unterstützung von Ausnahmen 40
  - 3.1.3 Granularitätsfähigkeit 43
  - 3.1.4 Strukturierbarkeit 43

3.1.5 Korrektheit 44
3.1.6 Auswertbarkeit 44
3.2 Betrachtung vorliegender Zeitmodelle 45
3.2.1 Generalisierte Intervalle 45
3.2.2 Time-Granularity Modell 46
3.2.3 Partitionen-Modell 47
3.2.4 Bewertung 48
3.3 Vorstellung des zyklenorientierten Zeitmodells 50
3.3.1 Zyklus und Zykluspositionen 51
3.3.2 Labels 52
3.3.3 Strukturierte ("geschachtelte") Zyklen 53
3.3.4 Abbildung von Periodizitätsschemas 56
3.4 Algebraische Notation von Periodizitätsschemas 58
3.4.1 Labelausdrücke 58
3.4.2 Änderungsoperationen 64
3.4.3 Auswahloperationen 68
3.4.4 Beispiele 78
3.5 Zusammenfassung 80

4 Der HSP-Baum **83**
4.1 Grundidee des HSP-Baums 83
4.2 Definition des HSP-Baums 86
4.2.1 H-Knoten 89
4.2.2 S-Knoten 93
4.2.3 P-Knoten 107
4.2.4 0-Knoten 114
4.2.5 Auswahl- und Projektionsparameter 116
4.2.6 Korrektheit 117
4.3 Realisierung der algebraischen Notationsform 117
4.3.1 Labelausdrücke 117
4.3.2 Änderungsoperationen 143
4.3.3 Auswahloperationen 148
4.4 Beispiele 157
4.5 Ausblick 162
4.6 Zusammenfassung 165

III Operationale Unterstützung für Periodizität

5 Auswertung von HSP-Bäumen **169**
5.1 HSP-Bäume und Wiederholungsmuster 169
5.2 Allgemeine Vorgehen zur Ableitung von Wiederholungsmustern 172
5.2.1 Kurzübersicht 172

5.2.2 Schritt 1: Angepasste Level-Order Traversierung 173
5.2.3 Schritt 2.1: Bestimmung der Wiederholungsmuster von Auswahl- und Projektionsebenen 175
5.2.4 Schritt 2.2: Transformation 178
5.2.5 Schritt 2.3: Entpacken der kompakten Leveldarstellung 180
5.3 Theoretische Grundlagen der Auswertung von Projektionsebenen 185
5.3.1 Semantik von Projektionsebenen 185
5.3.2 Theorie der periodischen Projektion 186
5.3.3 Umgang mit unterschiedlichen Instanziierungspunkten 190
5.4 Transformation: Auswertung von Projektionsebenen 194
5.4.1 Einelementige Positionsmenge 196
5.4.2 Mehrelementige Positionsmengen 197
5.5 Transformation: Auswertung von Auswahlebenen 203
5.5.1 Abhängige Auswahlebenen und nicht-auswahlerhaltende Operationsreihenfolgen 203
5.5.2 Abhängige Auswahlebenen und die periodische Projektion 206
5.5.3 Auswahlebenen und Auswahlparameter 208
5.6 Transformation: Berücksichtigung von Levelabhängigkeiten 212
5.6.1 Entstehung von Levelabhängigkeiten 212
5.6.2 Berücksichtigung von Levelabhängigkeiten 214
5.6.3 Umgang mit den Übertrag von Level zu Level 216
5.7 Algorithmus zur Ableitung von Wiederholungsmustern 217
5.8 Beispiel 218
5.9 Zusammenfassung 223

6 Die Bitletrepräsentation: Eine musterbasierte Realisierung von Vergleichsoperationen 225
6.1 Ausgangssituation der operationalen Unterstützung mit Wiederholungsmustern 226
6.1.1 Semantik von Vergleichsoperationen 226
6.1.2 Varianten der linearen Musterrepräsentation 227
6.1.3 Musterbasierte Realisierung von Vergleichsoperationen 228
6.2 Allgemeine Theorie der Bitletrepräsentation 232
6.2.1 Grundidee der Bitletrepräsentation 232
6.2.2 Grundlegende Begriffe und Eigenschaften 233
6.2.3 Wiederholungsmuster und Vektorräume 236
6.2.4 Induktives Basensystem 239
6.2.5 Bitletbasis 244
6.2.6 Bitlet-Transformation 249
6.2.7 Bitletrepräsentation 250
6.3 Fast-Bitlet-Transformation 255
6.3.1 Grundidee 255
6.3.2 Realisierung der inversen Fast-Bitlet-Transformation 256

6.3.3 Realisierung der Fast-Bitlet-Transformation 262
6.4 Realisierung von Vergleichsoperationen im Bitletbereich 265
6.4.1 Vorbetrachtungen zu Vergleichsoperationen 265
6.4.2 Grundidee der bitletbasierten Realisierung 267
6.4.3 Vergleichsoperationen im $\mathbb{Z}^2$-Vektorraum 269
6.4.4 Vergleichsoperationen im $\mathbb{Z}$-Vektorraum 272
6.4.5 Eingeschränkte Auswertung von Vergleichsoperationen 283
6.5 Diskussion 288
6.5.1 Unbeschränkte Auswertung von Vergleichsoperationen 288
6.5.2 Eingeschränkte Auswertung von Vergleichsoperationen 292
6.6 Ausblick 296
6.7 Zusammenfassung 299

IV Proof-of-Concept Prototyp, Zusammenfassung und Anhang

7 Proof-of-Concept Prototyp 305
7.1 Gesamtübersicht 305
7.2 Funktionsumfang und Bedienung des Prototyps 306
7.2.1 Beschreibung periodischer Spezifikationen 306
7.2.2 Auswertungen und Bitletrepräsentation 310
7.3 Vorgehensmodell 310
7.4 Anwendungsbeispiel 314

8 Zusammenfassung und Ausblick 319

A Algorithmus zur Ableitung von Wiederholungsmustern bei HSP-Bäumen 329
A.1 Die Funktion *getPattern*() 329
A.2 Bestimmung der Knoten des nächsttieferen Levels 330
A.3 Erzeugung des Wiederholungsmusters pro Level 330
A.4 Erzeugung der kompakten transformierten Leveldarstellung 331
A.4.1 Berücksichtigung von Projektionsebenen 332
A.4.2 Projektionsfunktion 333
A.4.3 Rotationsfunktion 333
A.5 Entpacken der kompakten Leveldarstellung 335

B Beweise 337
B.1 Beweis von Korollar 5.2 337
B.2 Beweis von Satz 6.1 338
B.3 Beweis von Satz 6.2 339

Abbildungsverzeichnis 341

Tabellenverzeichnis 347

Teil I

Einleitung, Anforderungen und bestehende Lösungsansätze

1 Einleitung

Durch die zunehmende Verbreitung von Informationssystemen in immer anspruchsvolleren Einsatzgebieten sind die Anforderungen an ihre Fähigkeiten zur Unterstützung temporaler Aspekte stark gestiegen [30, 50, 51]. Ein solches Gebiet ist die Unterstützung von Geschäftsprozessen durch prozess-orientierte Informationssysteme (engl. process-aware information systems, kurz PAIS) [9, 29, 30, 35, 92, 93]. Aufgrund der Tatsache, dass zeitliche Aspekte überall im unternehmerischen Prozess anzutreffen sind und maßgeblich über den Erfolg oder Misserfolg eines Unternehmens mit entscheiden (z.B. Überwachung der fristgerechten Bearbeitung von Kundenaufträgen), sind die Anforderungen an Informationssysteme zur Unterstützung entsprechender Aspekte hoch. Sie reichen von der zeitabhängigen Steuerung (d.h. Vorlage der richtigen Aktivitäten zur richtigen Zeit in Benutzerarbeitslisten), über die zeitbezogene Verwaltung von Ressourcen, bis zum Umgang mit Ressourcenkonflikten. Letztere gilt es, frühzeitig erkennen und behandeln zu können.

Aus diesen Gründen werden prozessorientierte Informationssysteme zunehmend mit Anforderungen konfrontiert, die so bislang nur in Projektmanagement-Werkzeugen bzw. speziellen Planungssystemen realisiert worden sind (z.B. Scheduling von Aktivitäten, Spezifikation und Überwachung von zeitlichen Minimal- bzw. Maximalabständen zwischen Vorgängen) [15, 37, 3, 22]. Für die Geschäftsprozessunterstützung ist die adäquate Handhabung zeitgebundener Beschränkungen bzw. Angaben jedoch essentiell [60, 75]. Insbesondere zählen hierzu auch die in der Praxis häufig anzutreffenden Prozesse, in denen elementare Aktivitäten oder Abfolgen von Aktivitäten in bestimmten zeitlichen Abständen (periodisch) wiederholt werden müssen. Die mangelhafte Unterstützung solcher periodischen Zusammenhänge (Periodizität) bzw. periodizitätsbehafteter Abläufe in prozess-orientierten Informationssystemen ist für viele Anwendungen ein großes Manko [48].

1.1 Temporale Aspekte in prozessorientierten Informationssystemen

Die elektronische Unterstützung ihrer Geschäftsprozesse gewinnt für Unternehmen zunehmend an Bedeutung [62, 54, 63]. Der Realisierung prozessorientierter Informationssysteme liegt die Idee zugrunde, die Prozesslogik vom eigentlichen Anwendungscode zu trennen [90, 92]. Zu diesem Zweck wird zur Entwicklungszeit für jeden zu unterstützenden Prozesstyp eine Prozess-Vorlage erstellt und im PAIS hinterlegt. Davon ausgehend können zur Laufzeit neue Prozess-Instanzen erzeugt werden, deren Ausführung über ihre komplette Lebensdauer hinweg vom PAIS gesteuert und überwacht wird.

Erfahrungen aus der Praxis haben gezeigt, dass eine Limitation heutiger PAIS die mangelhafte Unterstützung temporaler Aspekte bildet [60, 75, 30]. Für die Geschäftsprozessunterstützung ist die adäquate Handhabung zeitgebundener Beschränkungen jedoch essentiell [55, 24, 20, 22]. So sollten z.B. zeitliche Mindest-/Maximalabstände (unterschiedlicher Semantik) zwischen Aktivitäten oder exogene Termine in die WF-Steuerung und WF-Überwachung einbezogen werden können, um Anwender wirklich sinnvoll unterstützen zu können [57, 30, 2]. Entsprechende Beschränkungen sind oftmals technisch bedingt, gesetzlich vorgeschrieben oder wirtschaftlich motiviert. Ohne systemseitige Unterstützung sind solche temporalen Prozesse nur sehr schwer zu handhaben und aufkommende Konflikte, z.B. in Termin- und Arbeitsplänen oder bezüglich Personal- und sonstigem Ressourcenbedarf, sind kaum identifizierbar. Entsprechende Überprüfungen sind für Unternehmen jedoch von enormer Bedeutung, um Störungen im betrieblichen Ablauf zu vermeiden.

Die Berücksichtigung temporaler Aspekte erfordert, neben der Erweiterung von Prozess-Metamodelle um zeitliche Konstrukte, auch geeignete Methoden für die Spezifikation, Verifikation, Koordination und Überwachung zeitlicher Beschränkungen in Prozessen. Allerdings bedarf es einer flexiblen Unterstützung, da entsprechende Prozesse in der Praxis häufig Änderungen unterworfen sind und in Ausnahmefällen eventuell auch einmal ad hoc vom vorgeplanten Ablauf abgewichen können werden muss [70, 92, 69, 2, 91, 71]. Die hierzu verwendeten adaptiven PAIS [72, 73] müssen deshalb notwendige Überprüfungen bzw. Anpassungen von dynamischen Änderungen auch von temporalen Zusammenhängen selbst bei einer großen Zahl laufender Instanzen effizient bewerkstelligen können.

1.2 Periodizität in Informationssystemen

Die sehr schwach ausgeprägte (oder teilweise überhaupt nicht angebotene) Unterstützung temporaler Aspekte ist eine erhebliche Schwachstelle heutiger PAIS [37, 43, 60, 75, 7, 5] und limitiert deren Einsatzfelder in ganz erheblichem Maße. Beispielsweise weisen medizinisch-organisatorische Abläufe und Entwicklungsprozesse Aktivitäten auf, die in ihrer Bearbeitungsdauer beschränkt sind oder die untereinander bestimmte zeitliche Abstände einhalten müssen [30, 29, 60, 21, 20, 23, 22]. Besonders häufig und charakteristisch sind in diesem Umfeld Prozesse, in denen elementare Aktivitäten oder Abfolgen von Aktivitäten in bestimmten zeitlichen Abständen (periodisch) wiederholt werden müssen [85, 18, 1]. Wie oben erwähnt, ist die systemseitige Unterstützung von temporalen Prozessen im Allgemeinen und periodizitätsbehafteten Prozessen im Besonderen für viele Anwendungen essentiell. Ohne eine entsprechende Unterstützung ist es für Anwender oftmals sehr schwer, sich potenziell anbahnende Konflikte, wie zeitliche Kollisionen in Termin- und Arbeitsplänen oder Engpässe im Ressourcenbereich (z.B. Geräte, Räume, Personal), im Vorhinein zu erkennen. Das heißt, es geht nicht nur darum, periodische Angaben in Bezug auf die Ausführung eines Prozess spezifizieren zu können, sondern es geht im wesentlichen auch darum, periodische Angaben für eine möglicherweise sehr

große Anzahl von Prozessen (gleichen und verschiedenen Typs) im Zusammenwirken bewerten zu können.

Heutige auf dem Markt verfügbare Kalender- und Zeitplanungssysteme ermöglichen bereits eine Unterstützung einfacher periodischer Spezifikationen. In einigen Systemen können periodisch wiederkehrende Tätigkeiten anhand eines Schemas beschrieben werden (z.B. "*Jeden 1. Montag im Monat*" oder "*Jede Woche Mittwoch*"). Diese Angaben werden dazu benutzt, um innerhalb eines vorgegebenen Zeitintervalls die Menge entsprechender Tage zu bestimmen. Aus bzw. zu dieser "Treffermenge" kann der Benutzer dann noch manuell Tage abwählen oder Tage hinzufügen. Gespeichert wird dann das materialisierte Ergebnis, d.h. es werden alle Tage explizit im Kalender eingetragen, der ursprüngliche periodische Zusammenhang geht verloren. Ein ähnliches Vorgehen findet sich z.B. auch in der temporalen Erweiterung ORACLE8i™ Time Series des Datenbank-Management-System ORACLE8i™ [67]. Anhand eines gegebenen Periodizitätsschemas können aus einer bereits existierenden (expliziten) Menge von zeitlichen Angaben die jenigen Angaben ausgewählt werden, die dem gegeben Periodizitätsschema entsprechen. Periodischen Aspekten kann bei diesem Vorgehen somit nur im Fall bereits bestehender Wertemengen begegnet werden.

In heutigen PAIS findet man keine Funktionalität für den Umgang mit periodischen Aspekten. Es wäre auch nicht sinnvoll, sie in den oben skizzierten Formen direkt auf PAIS zu übertragen, da diese oftmals eine Vielzahl von Prozess-Typen und Tausende von Prozess-Instanzen verwalten müssen [4]. Die oben skizzierte Vorgehensweise der expliziten Materialisierung periodischer Ereignisse stößt wegen dem anfallenden Speicherplatzbedarf und Verwaltungsaufwand sehr rasch an seine Grenzen [48]. Darüber hinaus weisen die Einsatzszenarien der oben erwähnten Systeme meist nur eine geringe Anzahl an periodischen Spezifikationen auf. Das heißt, es bedarf zur Konflikterkennung lediglich einiger weniger Einzelvergleiche, wobei auftretende Konflikte meist interaktiv durch den Anwender aufgelöst werden müssen.

Vor diesem Hintergrund erwächst die Forderung nach einer integrierten und flexiblen Unterstützung periodischer Prozesse in PAIS. Eine solche Unterstützung erfordert geeignete Methoden zur direkten und umfassenden Spezifikation periodischer Aspekte auf semantisch hoher Ebene [59, 48]. Des Weiteren – und dies bildet die eigentliche Herausforderung – wird eine effiziente operationale Unterstützung von periodischen Spezifikationen benötigt, die als Grundlage für eine umfassende Überwachung, Koordination und Steuerung von periodizitätbehafteten Prozessen dient und die auch bei großer Zahl von Prozessinstanzen anwendbar bleibt [48].

1.2.1 Umfassende Spezifikation von Periodizität

Periodizität ist ein weit verbreitetes Phänomen der realen Welt und allgegenwärtig in vielen Anwendungsgebieten (z.B. Finanzsektor oder medizinischen Anwendungsdomänen) [31, 66, 28, 48, 49, 5]. Periodizität bzw. periodische Spezifikationen charakterisieren

einen Vorgang, der aus regelmäßig (periodisch) wiederkehrenden Aktivitäten besteht. So enthalten medizinisch-organisatorische Abläufe und Therapieprozesse häufig Aktivitäten oder Abfolgen von Aktivitäten, die periodisch wiederholt werden müssen. Ein Beispiel ist eine einfache Angabe der Art "*Alle 4 Stunden Gabe von Medikament X*" oder ein komplexes Therapieschema "*Eine Chemotherapie besteht aus Zyklen zu je 28 Tagen, wobei vom 1.-14. Tag Medikament A sowie am 1. und 8. Tag Medikament B verabreicht wird. Nach der letzten Gabe von Medikament A findet eine 14-tägige Pause statt. Es finden insgesamt 6 Zyklen statt.*".

Periodische Angaben, etwa "*jeden 3. Tag*" oder "*jede 4. Stunde*", gehören einer einfachen Klasse periodischer Spezifikationen an. In dieser Klasse der *äquidistanten* periodischen Spezifikationen bleibt der Abstand zwischen den einzelnen, wiederkehrenden Elementen gleich. Im Gegensatz dazu variiert in der Klasse der *nichtäquidistanten* Spezifikationen der Abstand zwischen den Elementen, wie in obigem Beispiel der Chemotherapie. Solche Abweichungen können, sowohl durch die Verwendung von Granularitäten wie Monat oder Jahr (z.B. "*jeden ersten Tag im Monat*") als auch durch Ausnahmen in den periodischen Spezifikationen (z.B. "*Gabe von Medikament B entfällt bei jedem 3. Zyklus*") hervorgerufen werden. Auch wenn entsprechende Abweichungen sehr komplex sein können, ist ihr grundsätzliches Verhalten in der Mehrzahl der Fälle wohlbekannt (z.B. Definition der Schaltjahresregel). Im Gegensatz dazu stehen unscharfe periodische Spezifikationen. Die Abstände zwischen den Elementen entsprechen hier einer Menge von möglichen Abständen. Beispielsweise führt die zusätzliche Bedingung "*Je nach Verfassung des Patienten wird nach jedem Zyklus eine Pause von 0-5 Tagen eingeräumt.*" zu Unschärfen im Verlauf einer Therapie.

Die allgemeinen Aspekte ausdruckstarker Spezifikationen von periodischen Zusammenhänge wurden bereits in verschiedenen Arbeiten näher untersucht (z.B. [87, 19, 39]). Hierbei wurden die obigen Klassen von Spezifikationen identifiziert, die für den praktischen Einsatz unumgänglich sind. Die allgemeinen Aspekte der Spezifikation von Periodizität sind deshalb mittlerweile schon recht gut verstanden. Im Gegensatz dazu fehlt ein solches Verständnis umfassender Spezifikationen auf semantisch hoher Ebene [48]. Ein Benutzer sollte periodische Spezifikationen bzw. Periodizitätsschemas auf semantisch hoher Ebene beschreiben können. So müssen entsprechende Beschreibungen sowohl für "Nicht-Experten" intuitiv nachvollziehbar sein als auch für die "Fachanwender" transparent und leicht anpassbar sein. Anpassungen werden z.B. für medizinische Therapien häufig erforderlich, da vorgegebene Periodizitätsschemas nur als Vorlagen dienen, die nachträglich an patientenabhängige Gegebenheiten angepasst werden müssen.

1.2.2 Operationale Unterstützung periodischer Spezifikationen

Im Umfeld von PAIS ist es in keinem Fall ausreichen, die Überprüfung zeitlicher Bedingungen oder Ressourcenzuweisungen in Prozessen nur unmittelbar vor Bearbeitung entsprechender Aktivitäten vorzunehmen. Dies würde zwar die Ausführung der Aktivi-

tät verhindern, falls eine zugehörige Bedingung verletzt ist, eine vorausschauende Überprüfung ist damit aber nicht möglich [60]. Dies gilt auch bei Überprüfungen lediglich einzelner Wiederholungen von periodizitätsbehafteten Prozessen, wie in [85, 25] vorgeschlagen. So können zwar Inkonsistenzen oder Probleme innerhalb der einzelnen Wiederholung identifiziert werden, Auswirkungen auf die gesamte Periodizität bzw. den Prozess sind aber nicht erkennbar. Vorausschauende Überprüfungen sind jedoch essentiell für die Geschäftsprozessunterstützung [55, 20, 88, 22]. So müssen etwa im Zusammenhang mit Ressourcenzuweisungen auch bei großer Zahl (periodizitätsbehafteter) Prozesse effizient und vorausschauend Konflikte erkannt werden können. Aufgrund der möglicherweise großen Anzahl an Prozessen ist es weiterhin unumgänglich, dass bestehende Konflikte weitestgehend automatisch aufgelöst werden können.

Zur Steigerung der temporalen Fähigkeiten heutiger Informationssysteme ist eine operationale Unterstützung unumgänglich, die einen effizienten Umgang mit periodischen Spezifikationen ermöglicht. Eine große Herausforderung bildet in diesem Zusammenhang die Realisierung generischer Vergleichsoperationen. Sie sollen es erlauben, das Zusammenwirken mehrerer periodischer Spezifikationen untereinander zu bewerten. So müssen unter anderem Aussagen zu Kongruenz, Überlappung oder Disjunktheit periodischer Angaben getroffen werden können, um z.B. vorausschauend (potentielle) Ressourcen- oder Terminkonflikte zu erkennen. Erschwert wird dies durch die Verwendung unterschiedlicher (zeitlicher) Granularitäten sowie in Folge von Ausnahmen und Unschärfen bei periodischen Angaben. Beispielsweise kann die zusätzliche Bedingung (siehe obige Chemotherapie) "*Medikament B muss von einem Spezialisten gegeben werden, der aber jeden ersten Mittwoch im Monat dienstfrei hat*" zu einem Konflikt zwischen dem Dienstplan des Spezialisten und der Gabe von Medikament B führen. Das heißt, in Abhängigkeit vom Startpunkt der Therapie können Überlappungen zwischen den dienstfreien Tagen und den Medikamentengaben auftreten.

Hieraus erwächst die Forderung nach einer adäquaten Repräsentationsform, die neben einer kompakten (d.h. speichereffizienten) Darstellung auch eine effiziente Operationsrealisierung ermöglicht. Für die zu entwickelnden Verfahren und Konzepte reichen rein konzeptuelle Betrachtungen alleine jedoch nicht aus. Stattdessen müssen die zur Realisierung von Vergleichen verwendeten logischen Modelle gleichzeitig auch eine speichereffiziente interne Darstellung gewährleisten können, um auch im Kontext von großen Datenmengen einsetzbar zu sein. Unter anderem gilt es, bei Vergleichen aufwendige Transformationen in andere Darstellungen zu vermeiden und diese möglichst auf einer geeigneten internen Darstellungsform durchzuführen. Gleichzeitig muss die (interne) Darstellungsform aber auch für Zugriffs- und Indexstrukturen geeignet sein, um so die Effizienz von Vergleichen bei großen Datenmengen sicherzustellen.

1.3 Ziele und Beitrag der Arbeit

Ziel der Arbeit ist die Entwicklung eines ganzheitlichen Ansatzes zur integrierten Unterstützung von Periodizität in (prozessorientierten) Informationssystemen. Hierzu werden Konzepte entwickelt die, neben einer ausdruckstarken Spezifikation, die effiziente Vergleichbarkeit von periodischen Zusammenhängen erlauben. Im Einzelnen ergeben sich die folgenden Forschungsfragestellungen:

- Welche logischen Zeitmodelle bzw. Formalismen ermöglichen eine ausdrucksstarke Spezifikation äquidistanter und nichtäquidistanter Periodizität, so dass auch Ausnahmen und Granularitäten berücksichtigt werden können?

- Welche Ansätze erlauben es, periodische Spezifikationen auf semantisch hoher Ebene beschreiben zu können, damit diese sowohl für "Nicht-Experten" intuitiv verständlich als auch für "Fachanwender" leicht anpassbar sind?

- Wie kann der Umgang mit periodischen Spezifikationen im Zusammenhang mit unterschiedlichen zeitlichen Einheiten (Granularitäten) adäquat unterstützt werden?

- Welche logische Repräsentation kann gleichzeitig eine effiziente Vergleichbarkeit periodischer Spezifikationen und eine kompakte (interne) Repräsentation gewährleisten?

- Wie können periodische Spezifikationen persistent und effizient gespeichert werden, so dass Vergleiche direkt auf dieser (internen) Repräsentationsform ausführbar sind?

- Welche Zugriffs- und Indexstrukturen können verwendet werden, um die Effizienz und Anwendbarkeit der entwickelten Lösungen auch bei großer Anzahl an Daten sicherzustellen?

Das zentrale Anliegen dieser Arbeit ist die integrierte Unterstützung von Periodizität in Informationssystemen. Die in der Arbeit entwickelten Konzepte und Modelle stellen Basistechnologien dar, die zur Unterstützung periodischer Zusammenhänge in unternehmensweiten (prozessorientierten) Informationssystemen (auch bei tausenden von Benutzern und aktiven Prozess-Instanzen), wesentlich sein werden. Die Umsetzung und Weiterentwicklung dieser Konzepte eröffnet völlig neue Möglichkeiten in Bezug auf die Realisierung prozessorientierter Anwendungen.

1.4 Aufbau der Arbeit

In Kapitel 2 werden zunächst grundlegende Begriffe eingeführt sowie konkrete Anforderungen an die Ausdrucksstärke und operationale Unterstützung vorgestellt. Im Anschluss daran werden in der Literatur existierende Ansätze beschrieben und entlang den identifizierten Anforderungen verglichen.

Teil II der Arbeit (Kapitel 3 – Kapitel 4) beschäftigt sich mit der adäquaten Beschreibung periodischer Spezifikationen.

Zu Beginn von Kapitel 3 werden bestehende Zeitmodelle hinsichtlich ihrer Erweiterbarkeit im Hinblick auf die in Kapitel 2 identifizierten Schwächen untersucht und bewertet. Im Anschluss daran werden die Ziele für den Entwurf eines adäquaten Zeitmodells diskutiert und im Anschluss das *zyklenorientierte Zeitmodell* vorgestellt. Es findet zunächst eine informelle Einführung der grundlegenden Konzepte und Eigenschaften des Zeitmodells anhand einer infiniten Modelldarstellung statt. Basierend auf dem neuen Zeitmodell wird dann eine algebraische Notationsform zur Beschreibung periodischer Spezifikationen eingeführt.

In Kapitel 4 erfolgt die Vorstellung des HSP-Baums als kompakte und berechenbare Repräsentationsform des zyklenorientierten Zeitmodells. Zusätzlich werden die zuvor informell vorgestellten Basisoperationen der algebraischen Notation auf eine Menge von Basisoperationen des HSP-Baumes mit wohldefinierter Semantik zurückgeführt und formal charakterisiert. Insbesondere soll ihre Anwendung die vom HSP-Baum bzw. dem zyklenorientierten Zeitmodell geforderten Korrektheits- und Konsistenzeigenschaften garantieren.

Teil III der Arbeit (Kapitel 5 – Kapitel 6) beschäftigt sich mit der operationalen Unterstützung periodischer Spezifikationen.

In Kapitel 5 werden die theoretischen Grundlagen sowie die konkreten Algorithmen zur Interpretation des HSP-Baums vorgestellt. Diese Auswertung ermöglicht es, Wiederholungsmuster von HSP-Bäumen abzuleiten, die für die operationale Unterstützung von periodischen Spezifikationen essentiell sind.

In Kapitel 6 wird ein innovativer Lösungsansatz zur Repräsentation von Wiederholungsmustern – die *Bitletrepräsentation* – vorgestellt. Die zugehörige Bitlet-Transformation verwendet das Prinzip linearer Transformationen und basiert auf Wiederholungsmustern. Auf Basis der Bitlet-Transformation ermöglicht die Bitletrepräsentation die kompakte Darstellung und effiziente Operationsrealisierung periodischer Spezifikationen bzw. entsprechender Wiederholungsmuster. Insbesondere die effiziente Realisierung von Vergleichsoperationen zwischen periodischen Spezifikationen bildet eine Stärke der Bitletrepräsentation, die in ideser Arbeit im Vordergrund stehen wird.

Teil IV beinhaltet die Vorstellung des Proof-of-Concept Prototyps sowie Zusammenfassung, Ausblick und Anhang der Arbeit.

In Kapitel 7 wird der Proof-of-Concept Prototyp vorgestellt, der im Rahmen dieser Arbeit entwickelt wurde. Neben des Funktionsumfangs wird auch dessen Handhabung an konkreten Beispielen erläutert.

In Kapitel 8 werden die Hauptergebnisse der Arbeit zusammengefasst und ein Ausblick auf weiterführende Themen gegeben.

2 Anforderungen, Problemstellungen und Stand der Technik

Dieses Kapitel gibt einen detaillierten Überblick über die Anforderungen und Problemstellungen sowie über die in der aktuellen Literatur vorliegenden Lösungsansätze zur Unterstützung periodischer Spezifikationen. Hierzu werden zunächst grundlegende Begriffe eingeführt, bevor Anforderungen an die Ausdrucksstärke und operationale Unterstützung vorgestellt werden. Im ersten Schritt werden zunächst wesentliche Ansätze zur Beschreibung periodischer Spezifikation in Bezug auf die Anforderungen Ausdruckstärke untersucht. Im zweiten Schritt werden dann die Ansätze zur Realisierung von Operationen bzw. Vergleichen zwischen periodischen Spezifikationen vorgestellt und bzgl. der bestehenden Anforderungen untersucht. Die Ergebnisse der Betrachtungen zum Stand der Technik werden abschließend zusammengefasst.

2.1 Grundlegende Begriffe und Anforderungen

Periodische Spezifikationen beschreiben Vorgänge oder Aktivitäten, die sich regelmäßig wiederholen. Ein typisches Beispiel für periodische Spezifikationen, etwa im klinischen Umfeld, stellt die bereits in Kapitel 1.1 erwähnte Chemotherapie dar. Ein anderes Beispiel bilden Fahrpläne für Züge oder Busse mit ihren stündlichen oder täglichen Ankunfts- bzw. Abfahrtszeiten. Sehr häufig variieren diese stündlichen oder täglichen Zyklen. So gelten am Abend oder am Wochenende oftmals andere Abfahrtszeiten als während des Tages oder an Wochentagen. Kennzeichnend für solche *zyklen-orientierten* Angaben ist die Verwendung eines *Zyklus* bestimmter Länge, der sich aus *Zykluselementen* zusammensetzt, z.B. "*1. und 8. Tag einer Chemotherapie*" oder "*die 15. und 45. Minute in einer Stunde bei Fahrplänen*".

Neben zyklen-orientierten Angaben stellen *kalender-orientierte* Angaben eine weitere Form von periodischen Spezifikationen dar, etwa "*Jeder 1. Tag im Monat*" oder "*Jeden Montag*". Ein weiteres Beispiel stellt die folgende Beschreibung von regelmäßig stattfindenden Projekttreffen dar: "*Während des Projektes finden Treffen jeden Mittwoch statt sowie zusätzlich jeden ersten Montag im Monat. Falls entsprechende Treffen auf einen Feiertag fallen, werden diese auf den nächsten Arbeitstag verschoben.*".

Auch wenn obige Beispiele für periodische Spezifikationen aus sehr unterschiedlichen Anwendungsgebieten stammen, sind entsprechende Angaben immer durch die folgenden drei Komponenten gekennzeichnet:

- **Periodizitätsschema:** Ein Periodizitätsschema legt das periodische Verhalten einer Spezifikation fest. Das heißt, das Periodizitätsschema beschreibt die gewünschte Abfolge von Zeitwerten in einem zugrundeliegenden Wertebereich. Zusammen mit der Angabe des Instanziierungszeitpunktes (siehe unten) kann das Schema "ausgeführt" bzw. angewendet werden und so eine Menge an konkreten Zeitwerten abgeleitet werden.
- **Instanziierungspunkt:** Der Instanziierungszeitpunkt gibt an, ab welchem Zeitpunkt ein Periodizitätsschema gestartet bzw. instanziiert werden soll.
- **Gültigkeitszeit:** Je nach Anwendung kann eine periodische Spezifikation in ihrem zeitlichen Gültigkeitsbereich bzw. in ihrer Wiederholungszahl eingeschränkt sein.

Eine besondere Rolle im Umgang mit periodischen Spezifikationen nimmt die Beschreibung und Repräsentation von Periodizitätsschemas ein. Im Allgemeinen können hier unterschiedliche Klassen von Periodizitätsschemas unterschieden werden.

2.1.1 Klassen von Periodizitätsschemas

Grundsätzlich unterscheiden wir drei Klassen von Periodizitätsschemas:

- **Quantitative Periodizitätsschemas:** Periodizitätsschemas dieser Klasse repräsentieren eine konkrete Menge von Elementen auf Basis von exakten und bzgl. eines Wertebereichs abzählbaren Informationen. Ein Beispiel hierfür bildet die Spezifikation *"An jedem 5.Tag"*.
- **Unscharfe (quantitative) Periodizitätsschemas:** Periodizitätsschemas, in denen Unschärfen bzw. Ungenauigkeiten enthalten sind, die sich jedoch bzgl. eines Wertebereichs noch abzählen lassen, werden durch diese Klasse repräsentiert. Die Aussage *"Jeden Tag zwischen 10 und 13 Uhr"* ist ein Beispiel für eine unscharfe Regel.
- **Qualitative Periodizitätsschemas:** Alle Periodizitätsschemas, die lediglich auf Basis von qualitativen Zusammenhängen periodische Vorgänge repräsentieren, fallen in diese Klasse. Bezüglich eines Wertebereichs werden in solchen Schemas keine abzählbaren Informationen verwendet. Das Beispiel *"Am Ende eines Seminars ist immer eine Kaffeepause."*, beschreibt einen periodischen, aber rein qualitativen Zusammenhang zwischen *Seminar* und *Kaffeepause*.

Quantitativen Schemas (auch unscharfen) liegt die Semantik zugrunde, dass eine einzelne Regel jeweils eine Menge von Werten repräsentiert. Bei qualitativen Schemas wird dagegen keine solche konkrete Wertemenge beschrieben. Vielmehr werden ausschließlich qualitative Eigenschaften einer beliebigen Menge von Werten festgelegt, ohne diese Werte konkret zu benennen. Mit qualitativen Schemas kann jedoch überprüft werden, ob die von ihr definierten (periodischen) Eigenschaften von einer gegebenen Wertemenge eingehalten werden. Eine Übersicht der drei Klassen zeigt Abbildung 2.1.

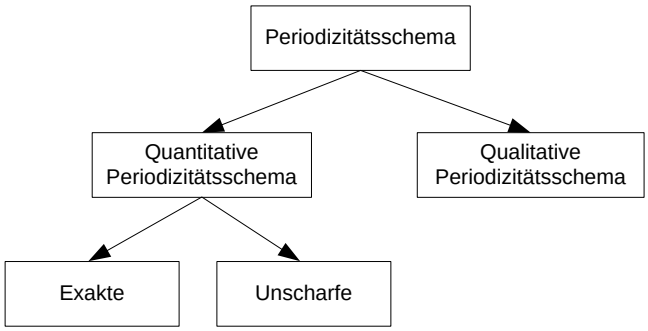


Abbildung 2.1: Übersicht der Klassen von Periodizitätsschemas

In der vorliegenden Arbeit wird der Fokus auf quantitativen, exakten Periodizitätsschemas liegen. Wie bereits in der Einleitung erwähnt, besteht das vorrangige Ziel dieser Arbeit in einem durchgängigen Ansatz zur Beschreibung und operationalen Unterstützung von periodischen Spezifikationen. Die vollständige Abdeckung aller drei Klassen von Periodizität steht hier nicht im Vordergrund. Auch wenn sich die im Folgenden entwickelten Modelle und Verfahren lediglich auf quantitative (exakte) Periodizitätsschemas beziehen, werden wir diese in späteren Betrachtungen zu unscharfen und qualitativen Periodizitätsschemas als Basis verwenden.

2.1.2 Ausprägungen von quantitativen exakten Periodizitätsschemas

Generell können zwei Formen von Periodizitätsschemas[1] unterschieden werden: **äquidistante** und **nicht-äquidistante**. Im äquidistanten Fall bleibt der Abstand zwischen den einzelnen (Zeit-) Werten des Schemas stets konstant, wie etwa Angaben "*Jeden 2. Tag*" oder "*Jede 3. Stunde*". Dies trifft für nichtäquidistante Periodizitätsschemas nicht zu, da hier der Abstand zwischen (Zeit-) Werten variiert. Beispielsweise ist dies in der zyklenorientierten Angaben "*1. und 8. Tag einer Chemotherapie mit der Länge 28 Tage*" oder in der kalender-orientierten Angabe "*Jeder 1. Tag im Monat*" der Fall. Allerdings kann insbesondere das nichtäquidistante Verhalten eines Schemas unterschiedliche "Ursachen" haben, was letztlich zu verschiedenen Ausprägungen von Periodizitätsschemas führt.

Die einfachste Form nichtäquidistanter Periodizitätsschemas bilden Angaben, in denen die Reihenfolgen der variierenden Abstände vorgegeben sind. Beispielsweise führt in der Chemotherapie (Zykluslänge = 28 Tage) die Gabe von Medikament B am 1. und 8. Tag zu den variierenden Abständen von 8 bzw. 20 Tagen (siehe oben) und somit zu einem nichtäquidistanten Verhalten. Komplexere Formen von nichtäquidistantem Verhalten entstehen in Verbindung mit Ausnahmen sowie der Verwendung von Granularitäten unterschiedlicher Dauer in Periodizitätsschemas.

[1] Im Folgenden sowie im weiteren Verlauf der Arbeit sind mit dem Begriff Periodizitätsschemas stets quantitative exakte Periodizitätsschemas gemeint.

2.1.2.1 Berücksichtigung von Ausnahmen

Eine Ausnahme in einem Periodizitätsschema entspricht einer zeitweisen Abweichung bzw. Änderung vom vorgegebenen Grundschema. Von der Änderung kann sowohl eine *periodische* als auch *nicht-periodische Auswahl* an Elementen des Grundschemas betroffen sein. Diese Auswahl entspricht entweder einer Menge an Zyklen oder einer Menge von Werten des Grundschemas, unabhängig von ihrer Zugehörigkeit zu einem Zyklus.

Die Angaben "*Bei jedem 2. Zyklus erfolgt am 10. Tag eine zusätzliche Gabe von Medikament B*" oder "*Jedem 2. Zyklus wird ein Bluttest nach der Gabe von Medikament A benötigt*" entsprechen periodischen Ausnahmen, von der die Zyklusauswahl "*Jeden 2. Zyklus*" betroffen ist. Bei der periodischen Ausnahme "*Nach jeder 10. Gabe von Medikament B wird eine Laboruntersuchung benötigt*", werden die Zyklen, in dem die Gabe von Medikament B stattfindet, vernachlässigt. Stattdessen werden die Gaben von Medikament B als Menge betrachtet, in der "*Jede 10.Gabe von Medikament B*" von der Ausnahme (*Laboruntersuchung*) betroffen ist.

Periodizitätsschemas weisen häufig nicht nur eine einzelne Ausnahme auf. Somit kann die Auswahl von Zyklen oder Elementen auch mehrfach pro Schema erfolgen. Bezüglich der Ausdrucksstärke der periodischen Auswahl bestehen die gleichen Anforderungen wie im allgemeinen Fall von Periodizitätsschemas. Beispielweise kann bei einer Chemotherapie eine Ausnahme auf Basis eines periodischen Feiertags definiert sein. Das Auswahlschema, d.h. das Periodizitätsschema für den Feiertag, kann wiederum nichtäquidistantes Verhalten aufweisen und Ausnahmen beinhalten.

Im Gegensatz zu periodischen Ausnahmen entsprechen Projekttreffen, die an bestimmten Tagen annulliert werden, oder vereinzelte Verlängerungen von Therapiezyklen, nichtperiodischen Ausnahmen. Bei solchen Ausnahmen ist lediglich eine endliche Menge an einzelnen Werten oder Zyklen von einer Änderung betroffen.

Im Hinblick auf die konkrete Änderung an der periodischen oder nicht-periodischen Auswahl eines Periodizitätsschemas können drei Ausnahmeklassen unterschieden werden:

- **Löschung/Einfügung:** Das Löschen oder Einfügen von Zykluselementen in bestimmte Zyklen oder von Werten in der Wertemenge sind Änderungen, die in diese Ausnahmeklasse fallen. Typische Beispiele hierfür sind die vorherigen Angaben mit dem Einfügen von zusätzlichen Medikamentengaben oder weiteren Untersuchungen (Bluttest) sowie das Annullieren bzw. Löschen von Projekttreffen.
- **Selektive Verschiebung:** Das selektive Verschieben einzelner Zykluselemente oder einzelner Werte ohne Beeinflussung nachfolgender Elemente oder Werte fällt in diese Ausnahmeklasse. Beispiele hierfür bilden die Angabe "*Projekttreffen an Feiertagen werden auf den nächsten Arbeitstag verschoben*" sowie das Verschieben der Gabe von Medikament B in bestimmten Zyklen.
- **Zusammenhängende Verschiebung**: Die Änderungen führen zu einem zusammenhängenden Verschieben aller nachfolgenden Zyklen oder Werte. Beispielsweise

führt die Änderung von Zykluslängen zu einer Verschiebung aller nachfolgenden Zyklen inklusive ihrer Zykluselemente. Ein weiteres Beispiel bildet die Anpassung von Projekttreffen an die Sommer- bzw. Winterzeit. Die Berücksichtigung der Umstellung um eine Stunde führt zu einem zusammenhängenden Verschieben aller nachfolgenden Werte bzw. Projekttreffen. Die gleiche Situation ergibt sich bei Berücksichtigung der Schaltjahresproblematik, d.h. einer Änderung der Dauer des Monats Februars.

In den ersten beiden Klassen haben Ausnahmen lediglich "lokale" Auswirkungen in der korrespondierenden Wertemenge eines Periodizitätsschemas. In der letzten Klasse hingegen haben Ausnahmen "globale" Auswirkungen in der Wertemenge.

Die Berücksichtigung von Ausnahmen in Periodizitätsschemas führt, je nach Ausnahmeklasse zu komplexem, nichtäquidistantem Verhalten. Allerdings bilden Ausnahmen nur eine Ursache für nichtäquidistante Periodizitätsschemas. Auch die Verwendung bestimmter Zeiteinheiten (sog. Granularitäten) kann diesen Effekt hervorrufen.

2.1.2.2 Verwendung von Granularitäten

Granularitäten (Zeiteinheiten) entsprechen unterschiedlichen zeitlichen Abstraktionsebenen zur Beschreibung von Zusammenhängen. So sind beispielsweise in Chemotherapien Angaben in Tagesabständen ausreichend, während in anderen Therapien stündliche Angaben benötigt werden. Im Finanzsektor wiederum sind monatliche oder jährliche periodische Aussagen typisch. Welche Granularität bzw. Granularitäten im Einzelnen verwendet werden, ist somit anwendungsabhängig. Allerdings müssen grundsätzlich zwei Verwendungsformen von Granularitäten innerhalb von Periodizitätsschemas unterschieden werden: *relative* und *absolute*.

Bei der **relativen Verwendung** dienen Granularitäten lediglich als unterschiedliche Maßeinheiten für die Angabe von Abständen. Beispielsweise entspricht die Angabe der Länge einer Chemotherapie in *Tagen* oder die Angabe "*jede 3.Stunde*" einer relativen Verwendung von Granularitäten.

Bei der **absoluten Verwendung** hingegen (z.B. "*Jeden 1.Tag im Monat*"), übernehmen die Granules der Granularität *Monat* die Rolle von Zyklen und die Granules der Granularität *Tag* die Rolle von Zykluselementen. Ein Granule (engl.) bezeichnet die einzelnen Einheiten einer Granularität, d.h. ein einzelner *Tag* ist ein Granule der Granularität *Tag* bzw. der Monat *Januar* ein Granule der Granularität *Monat* [8]. Kennzeichnend ist hierbei, dass sich das periodische Verhalten der Granularitäten auf das Periodizitätsschema "überträgt". Im gegebenen Beispiel etwa müssen die jeweiligen Monatslängen sowie die Schaltjahresregel für Februar im Schema berücksichtigt werden. Nur so sind die korrekten Monatslängen und somit die Abstände zwischen den einzelnen Werten sichergestellt. Allerdings zeigt die nähere Betrachtung, dass es sich bei der Schaltjahresregel um das bereits beschriebene zusammenhängende Verschieben handelt. Das heißt, "*Jedes*

4. Jahr, nicht jedes 100. Jahr und jedes 400. Jahr" entspricht einem periodischen Auswahlschema für Februar zur Verlängerung der Monatslänge bzw. der Zyklen um einen Tag. In der absoluten Verwendung können Granularitäten somit als ein vom Kalender vorgegebenes Periodizitätsschema mit Ausnahmen verstanden werden. Die Ursache für das nichtäquidistante Verhalten eines entsprechenden Periodizitätsschemas bilden somit der verwendete Kalender sowie die verwendete Granularität.

Die Berücksichtigung von Ausnahmen ermöglicht eine einheitliche Unterstützung von Periodizitätsschemas sowohl für kalender- als auch für zyklen-orientierte Spezifikationen. Allerdings gilt es, einen wesentlichen Unterschied zwischen beiden Spezifikationsformen zu beachten: Die Beschreibung der in kalender-orientierten Spezifikationen verwendeten Granularitäten erfolgt stets in Abhängigkeit vom Startpunkt des verwendeten Kalenders. Dieser Startpunkt ist für einen Kalender bzw. eine Granularität jedoch fest vorgegeben und darf nicht (beliebig) geändert werden. Für kalender-orientierte Spezifikationen bedeutet dies, dass diese zwar einen eigenen Instanziierungspunkt aufweisen können, "intern" jedoch stets den festen Startpunkt des Kalenders verwenden müssen. Im Gegensatz dazu wird bei zyklen-orientierten Angaben stets der vorgegebene Instanziierungspunkt verwendet.

Die vorgestellten Formen von Periodizitätsschemas geben einen ersten Überblick über die benötigte Ausdruckstärke von Beschreibungsformalismen. Wir erheben keinen Anspruch auf Vollständigkeit. Allerdings stellt die ausdrucksstarke Beschreibung von Periodizitätsschemas bzw. periodischen Spezifikationen einen, wichtigen Aspekt der systemseitigen Unterstützung dar. Der zweite Aspekt und zugleich größere Herausforderung ist die im Folgenden vorgestellte operationale Unterstützung von periodischen Spezifikationen.

2.1.3 Operationale Unterstützung

Die operationale Unterstützung umfasst eine Menge von Operationen, die den grundsätzlichen Umgang mit periodischen Spezifikationen ermöglichen. Hierzu zählen Operationen, die z.B. konkrete Zeitwerte für eine Spezifikation erzeugen oder überprüfen, ob eine Spezifikation einzelne Werte bzw. eine vorgegebene Menge von Werten beinhaltet.

Die größte Herausforderung bilden Operationen zum Vergleich periodischer Spezifikationen. In vielen Anwendungen ist es unumgänglich, Aussagen zu Überlappungen, Kongruenz oder Disjunktheit von Werten zwischen periodischen Spezifikationen zu treffen, etwa im Hinblick auf eine vorausschauende Konflikterkennung.

Im Allgemeinen können die folgenden drei Klassen von Operationen unterschieden werden:

- **Ableitungsoperationen** dienen zur Ableitung konkreter Zeitwerte einer periodischen Spezifikation. Diese Operationen können sowohl Formen von "Zählern" (z.B. *3. Wert einer Spezifikation* oder *alle Werte des 4. Zyklus*) aufweisen als auch

Semantiken der Form "*nächster Wert der Spezifikation*" bzgl. eines gegebenen Zeitpunktes (mit vorgegebener Granularität).

- **Überprüfungsoperationen** testen für einzelne Werte bzw. Mengen von Werten, ob diese in einer periodischen Spezifikation enthalten sind oder durch diese erzeugt werden können. Oftmals will man auch vorausschauend feststellen, ob bestimme Zeitpunkte bzw. Termine in einer Spezifikation enthalten sind. Darüber hinaus dienen Überprüfungsoperationen auch dazu, die Ausführungshistorie einer Spezifikation nachträglich auf ihre Korrektheit zu testen bzw. diese zu bestätigen.
- **Vergleichsoperationen** dienen der Bestimmung von gemeinsamen oder verschiedenen Werten periodischer Spezifikationen mit gleichen oder verschiedenen Periodizitätsschema. Diese Operationen werden benötigt, um Aussagen zu Konflikten in entsprechenden Anwendungen treffen zu können, wie etwa im einleitenden Beispiel der Chemotherapie und dem Dienstplan des Arztes.

Wie erwähnt, stellt die Realisierung von Vergleichsoperationen die größere Herausforderung dar und wird deshalb noch einen Schwerpunkt in dieser Arbeit bilden. Im Folgenden werden hierzu weitere Anforderungen aufgezeigt.

Vergleichsoperationen zwischen periodischen Spezifikationen realisieren – sofern keine Schranken explizit angegeben werden – einen Vergleich zwischen zwei (theoretisch unendlich großen) Wertemengen. Das heißt, es werden bestimmte Mengenbeziehungen zwischen den Spezifikationen überprüft. Beispielsweise entspricht die Teilmengenbeziehung $Dienstfrei \subset Chemotherapie$ überlappenden Wertemengen zwischen den beiden Spezifikationen, d.h. einige Termine der Chemotherapie stehen mit dienstfreien Tagen des Arztes in Konflikt.

Das *Auswertungsergebnis* eines Vergleichs bildet einen wichtigen Aspekt entsprechender Operationen. Im einfachsten Fall ist dies lediglich eine *boolesche Auswertung,* die überprüft, ob entsprechende Beziehungen vorliegen oder nicht. Allerdings trifft eine solche Auswertung keine Aussage, welche konkreten Werte die entsprechende Mengenbeziehung erfüllen bzw. nicht erfüllen. Bezogen auf das obige Beispiel bedeutet das, dass die konkreten Ereignisse, die den Konflikt verursachen, nicht lokalisiert werden können. Im Allgemeinen ist eine solche Lokalisierung jedoch unabdingbar, etwa um im Konfliktfall alternative Personen bzw. Ressourcen zu finden und zuteilen zu können. Eine *quantitative Vergleichsauswertung*, welche die Werte einer Mengenbeziehung identifiziert, stellt das in der Regel benötigte Auswertungsergebnis dar.

Neben dem Auswertungsergebnis ist auch der *Auswertungszeitraum* von Vergleichen ein wichtiger Aspekt. Wie erwähnt, werden bei einem Vergleich zwei möglicherweise unendlich große Wertemengen miteinander verglichen. Somit können entsprechende Konflikte zwischen periodischen Spezifikationen erst "sehr spät" in deren Verlauf auftreten. In der Praxis sind solche "späten" Konfliktaussagen meist nicht relevant. Beispielsweise ist die Erkennung eines Konflikts in einer Chemotherapie, der sich erst in 10 Jahren etabliert, im Normalfall uninteressant. Im Gegensatz zu einer solchen *unbeschränkten Auswertung* ist es häufig ausreichend und sinnvoller, den Auswertungszeitraum eines Vergleichs

einzuschränken. Grundlage für eine solche *eingeschränkte Auswertung* bildet ein *Ereignishorizont,* der den Gültigkeitszeitraum einer Konfliktaussage anwendungsabhängig festlegt. Der Ereignishorizont kann konstant sein oder aber von Vergleich zu Vergleich variieren.

Es reicht allerdings nicht aus, lediglich die prinzipielle Realisierbarkeit von Vergleichsoperationen bezogen auf einen bestimmten Spezifikationsformalismus zu betrachten. Auch die Effizienz der resultierenden Implementierungen ist sehr wichtig und diese ist wiederum sehr stark abhängig von der physischen Repräsentation von periodischen Spezifikationen. Neben einem möglichst *kompakten Speicherverhalten*, muss diese auch eine möglichst *effiziente Ausführung der Operationen* (Speicher- und Laufzeitverhalten) gewährleisten können. Insbesondere gilt es, aufwendige Transformationen zum Vergleichs- bzw. Anfragezeitpunkt zu vermeiden.

2.2 Ansätze zur Beschreibung periodischer Spezifikationen

In der Literatur findet sich eine Vielzahl an Ansätzen zur logischen Beschreibung von Periodizitätsschemas.

Wir konzentrieren uns im Folgenden auf Ansätze zur Beschreibung quantitativer Periodizitätsschemas. Nicht vertieft werden Ansätze, die ausschließlich qualitative periodische Zusammenhänge berücksichtigen (z.B. [79, 27, 28]). "Unscharfe" quantitative Periodizitätsschemas, wie etwa "*an einem beliebigen Montag innerhalb von jedem Monat*", werden nicht berücksichtigt. Hierzu zählen auch Periodizitätsschemas, in denen Aktivitäten zusätzlich bestimmte Durchführungsdauern aufweisen. Die alleinige Angabe einer Durchführungsdauer etwa "*Die Gabe von Medikament B an jedem 1. und 8.Tag eines Chemotherapiezyklus benötigt 6 Stunden*", ohne Aussage, wann genau diese Aktivität innerhalb der einzelnen Tage stattfinden wird, führt zu einer Unschärfe. Allerdings weisen entsprechende Periodizitätsschemas ohne Berücksichtigung der Durchführungsdauern wieder ein exaktes quantitatives periodisches Verhalten auf, welches im Folgenden im Vordergrund stehen wird.

Viele der im Folgenden erwähnten Ansätze entstammen dem Umfeld der Beschreibung von Granularitäten bzw. allgemein dem Themenfeld "Temporal Knowledge Representation and Reasoning", d.h. der Repräsentation von zeitabhängigem Wissen und seine Auswertung. Dies ist dadurch bedingt (siehe Abschnitt 2.1.2), dass Granularitäten und Periodizitätsschemas logisch betrachtet lediglich zwei Seiten derselben Medaille darstellen. Auch wenn an periodische Spezifikationen operationale und interne Anforderungen bestehen, die weit über die von Granularitäten hinausgehen, können sie auf Ebene der formalen Beschreibung als gleich betrachtet werden.

Bezüglich des verwendeten Beschreibungsparadigmas können *algebraische* und *constraint-basierte* Ansätze unterschieden werden. Im Folgenden steht hierbei die generelle Vorgehensweise entsprechender Ansätze im Vordergrund.

2.2.1 Algebraische Ansätze

Kennzeichnend für diese Form des Beschreibungsparadigmas sind formale Operationen, deren sukzessive Anwendung die periodischen Eigenschaften eines Periodizitätsschemas schrittweise konstruieren. Vertreter dieser Kategorie sind z.B. der *Collection-Formalismus* [53] (bzw. erweiterte oder angepasste Varianten davon [17, 26, 12, 81, 16, 38]), der *Slice-Formalismus* [64, 46, 45, 6] sowie die *Calendar-Algebra* [8, 65].

Im Allgemeinen können bei diesen Ansätzen zwei Grundformen von Operationen unterschieden werden: Gruppierungs- und Selektionsoperationen. In Bezug auf Granularitäten dienen Gruppierungsoperationen der Bildung neuer Granularitäten durch "Zusammenfassen" bzw. "Gruppieren" einer bestimmten Anzahl von aufeinanderfolgenden Granules einer bestehenden Granularität. Der algebraische Ausdruck $Tage = Group_{24}(Stunden)$ beispielsweise beschreibt die Granularität *Tage* durch Gruppieren von jeweils 24 *Stunden*. Im Zusammenhang mit Periodizitätsschemas dienen diese Operationen der Beschreibung von Zykluslängen, wie bei der Chemotherapie durch das Gruppieren von 28 *Tagen* zu einem Zyklus (d.h. $Chemo = Group_{28}(Tage)$).

Für die Unterstützung nichtäquidistanter Granularitäten bzw. Zyklen finden sich spezielle Gruppierungsoperationen. Beispielsweise werden bei der Gruppierungsoperation $Alter^{m}_{p_1 \ldots p_n, k}(G_1, G_2)$ die Granules von G_1 jeweils in Gruppen zu je m Granules (logisch) zusammengefasst und die Granules an Position $p_1, ..., p_n$ um den Wert k von G_2 verkleinert bzw. vergrößert [8]. Das nachfolgende Beispiel zeigt eine mögliche Beschreibung der Granularität $Monate$. Hierbei werden in M_1 zunächst Granules durch Gruppieren von jeweils 30 Tagen gebildet und anschließend die Granules bzw. Monate mit 31 Tagen mit der Operation *Alter* entsprechend verlängert. Im Anschluss wird in M_2 der Monat Februar auf 28 Tage verkürzt. Im letzten Schritt wird in der Granularität $Monate$ basierend auf einer vereinfachten Schaltjahrregel ("*jedes 4.Jahr*") die Länge der Februare in Schaltjahren angepasst, d.h. auf 29 Tage verlängert.

$$M_1 = Alter^{12}_{(1,3,5,7,8,10,12),+1}\,(Group_{30}(Tage), Tage)$$

$$M_2 = Alter^{12}_{2,-2}\,(M_1, Tage)$$

$$Monate = Alter^{12*4}_{2*4,+1}\,(M_2, Tage)$$

Die zweite Grundform von Operationen, die Selektionsoperationen, erzeugen neue Granularitäten durch "Selektieren" bestimmter Granules aus bestehenden Granularitäten. Die Operation $Select_{p_1,...,p_2}(G, G_2)$ etwa, wählt die Granules an Positionen $p_1, ..., p_2$ der Granularität G_2 aus, die in einem Granule der Granularität G zusammengefasst sind.

So beschreibt zum Beispiel der Ausdruck $Select_{1-5}(Woche, Tage)$ die Granularität *Arbeitswoche* durch Selektieren der Granules *Montag (1.)-Freitag (5.)* aus der Granularität *Woche*. In Bezug auf Periodizitätsschemas ermöglichen Selektionsoperationen die Auswahl bestimmter Zykluselemente aus einem Zyklus. So entspricht beispielsweise die Gabe von Medikament B einer Chemotherapie ($Chemo = Group_{28}(Tage)$) am 1. und 8. Tag eines jeden Zyklus dem Ausdruck $Select_{1,8}(Chemo, Tage)$ oder die Angabe "*Jeden 1. Tag im Monat*" dem Ausdruck $Select_1(Monate, Tage)$.

In den Arbeiten zur algebraischen Beschreibung finden sich die oben dargestellten Grundoperationen Selektion und Gruppierung in unterschiedlichster Form wieder. In der *Calendar-Algebra* entsprechen diese Operationen den oben dargestellten Grundoperationen [8, 65]. Zusätzlich finden sich weitere spezialisierte Gruppierungs- und Selektierungsoperationen, wie zur Berücksichtigung von Zeitverschiebungen.

Im *Slice-Formalismus* [64, 12] gibt es diese Grundoperationen in Form eines normierten Operationsausdruck über mehrere Granularitätsebenen. Ein solcher Ausdruck (d.h. ein Slice) hat die Form $O_n.G_n + ... + O_1.G_1$, wobei ein Ausdruck $O_i.G_i$ jeweils eine Selektionsoperation für die Granularität G_i beschreibt, d.h. $O_i.G_i \equiv select_{O_i}(G_{i+1}, G_i)$. Es werden jeweils alle Granules der Granularität G_i selektiert, die in Bezug auf die Granularität G_{i+1} an den relativen, in O_i angegebenen Positionen stehen. Eine Granularität G_i wird als Menge von Intervallen verstanden, die durch verschiedene Gruppierungsoperationen gebildet werden können. Diese sind in ihrer Semantik identisch zu den Gruppierungsoperationen der Calendar-Algebra [8, 12]. Die Granularitäten des Ausdrucks sind geordnet und jede Granularität G_i besteht aus ganzzahligen Granules der Granularität G_{i-1}, wobei die Anzahl variieren kann (siehe $Monate$). Die einzelnen Ausdrücke sind jeweils "Und"-verknüpft. Der Ausdruck $*.Jahre + \{1, 2\}.Monate + \{1\}.Tage$ etwa entspricht der Angabe "*an jedem 1. Tag des 1. und 2. Monats (Jan, Feb) eines jeden Jahres*". Einen umfassenden Überblick sowie Vergleich der Ausdruckstärke von sowohl dem Slice- als auch des Collection-Formalismus gegenüber der Calendar-Algebra (welche die beiden anderen subsumiert) findet sich in [8].

Grundsätzlich ermöglicht es das auf Gruppierungs- und Selektierungsoperationen basierende Vorgehen, äquidistante und nicht-äquidistante Granularitäten bzw. Periodizitätsschemas zu beschreiben. Allerdings weisen die vorliegenden Vorschläge einige Schwächen im Umgang mit Ausnahmen in Periodizitätsschemas auf. So entspricht etwa die oben verwendete Operation $Alter()$ lediglich Ausnahmen mit äquidistanter Elementauswahl. Ausnahmen mit nichtäquidistanten Auswahlschema oder Zyklusauswahl werden dagegen nicht berücksichtigt, etwa bei der Angabe "*An jedem 3. Zyklus erfolgt am letzten Tag sowie an den ersten drei Tagen des darauffolgenden Zyklus eine zusätzliche Vitamingabe*". Dies stellt jedoch kein Problem des betreffenden Ansatzes dar, sondern ist eine inhärente Schwäche aller vorliegenden algebraischen Ansätze.

Bei Erweiterung entsprechender Formalismen um noch nicht unterstützte Formen von Ausnahmen, erweist sich die dargestellte Spezifikationslogik als hinderlich. Kennzeichnend für diese Logik ist, dass bei Ausnahmen sowohl die Art und Form des Auswahlsche-

mas als auch die konkrete Änderungsoperation in einer einzelnen Operation "verschmolzen" werden, wie bei *Alter*() mit äquidistanter Elementauswahl und Vergrößern/Verkleinern von Granules. Das allgemeine Vorgehen bei Ausnahmen, auf Basis eines Auswahlschemas bestimmte Elemente bzw. Zyklen zu "markieren" und gezielt mit Änderungsoperationen zu manipulieren, ist lediglich implizit in der Semantik der Operation hinterlegt. Eine Erweiterung entsprechender Formalismen, unter Beibehaltung dieser "versteckten" bzw. impliziten Spezifikationslogik, führt durch die möglichen Kombinationen von Änderungsoperationen, Auswahlschema und Auswahlart zu einer Vielzahl an Einzeloperationen. Neben dem Problem der Benutzerfreundlichkeit erweist sich die Vielzahl an Einzeloperationen, insbesondere bei einigen relevanten Realisierungsstrategien für Operationen, als hinderlich (siehe später). Eine explizite und generische Unterstützung von Ausnahmen ist deshalb wünschenswert.

2.2.2 Constraint-basierte Ansätze

Die Beschreibung periodischer Zusammenhänge erfolgt bei dieser Klasse von Ansätzen durch logisch verknüpfte Terme (Prädikate, Constraints), die deklarativ die Eigenschaften eines Periodizitätsschemas festlegen. In der Literatur basieren constraintbasierte Ansätze auf Logiken erster Ordnung (z.B. [86, 87, 39]), temporalen Logiken (z.B. [18, 34, 19, 61]) und deduktiven Logiken (z.B. [95, 89]).

In einfachen Ansätzen beziehen sich die Terme unmittelbar auf einen zugrundeliegenden Wertebereich einer Granularität bzw. eines Periodizitätsschemas. Beispielsweise beschreiben das einstellige Prädikat $periodic_{m,p}(x)$ bzw. der Ausdruck $periodic_{28,1}(x) \vee periodic_{28,8}(x)$ die Gabe von Medikament B der Chemotherapie am 1. und 8. Tag eines Zyklus. Die Semantik solcher Constraints ist mit einer Menge linearer Geradengleichungen $m * x + p$ identisch. Das obige Beispiel kann somit als Gleichungen $x = 28 * i + 1$ und $x = 28 * i + 8$, $i = 0, 1, 2, \ldots$ verstanden werden. Die erfüllenden Belegungen des Prädikats bzw. der Variable x – in diesem Fall $\{1, 8, 29, 36, \ldots\}$ – bilden die Wertemenge der Granularität oder des Schemas. Kennzeichnend für diese Ansätze ist die ausschließliche Verwendung einer einzigen Granularität, die den zugrundeliegenden Wertebereich korrespondiert. Die Verwendung unterschiedlicher Granularitäten ist nur durch das explizite "Ausrechnen" der Längen von (äquidistanten) Granularitäten möglich. Die Gabe von Medikament B etwa kann in einem mit *Stunden* korrespondierenden Wertebereich durch den folgenden Ausdruck beschrieben werden: $periodic_{(24Stunden*28),1} \vee periodic_{(24Stunden*28),8}$.

Zur impliziten Beschreibung von Granularitäten sowie deren hierarchischen Abhängigkeiten werden Prädikate in Bezug auf eine zugrundeliegende abstrakte Struktur definiert. Die gewünschte Granularität entspricht genau der Struktur, die diese Prädikate erfüllt. Beispielsweise bildet eine solche Struktur die beiden (logischen) Mengen von Start- und Endpunkten der Granules einer Granularität [19]. Auf Basis der Prädikate werden implizit Eigenschaften zwischen Start- und Endpunkten bzw. Abhängigkeiten

zwischen Granules unterschiedlicher Strukturen beschrieben. So beschreibt etwa das Prädikat $Group_p(G_1, G_2)$ eine Granularität G_2, in der jedes Granule aus einer Menge aufeinander folgender Granules der Granularität G_1 besteht. Das Prädikat entspricht lediglich einer Abkürzung für die konkreten logischen Formeln, die seine formale Semantik auf Basis einer geeigneten Logik festlegen. Die Granularität $Tage$ etwa kann durch den Ausdruck $Tage(\mathbf{x}) = Group_{24}(Stunden(\mathbf{y}), \mathbf{x})$ und die Granularität $Woche$ mit $Woche(\mathbf{z}) = Group_7(Tage(\mathbf{x}), \mathbf{z})$ beschrieben werden. Der erste Ausdruck $Tage(x)$ ist ein Prädikat, das genau dann erfüllt ist, wenn eine Belegung der Variable x das Prädikat $Group_{24}(y, x)$ mit einer zweiten Struktur y erfüllt. Diese Struktur y wiederum muss zusätzlich das Prädikat $Stunden(y)$ erfüllen, d.h. y entspricht der Granularität *Stunden*. Die gleiche Semantik findet sich bei dem zweiten Ausdruck zur Beschreibung der Granularität $Woche$. Die erfüllenden Belegungen für die Variablen x und z bzw. die jeweiligen Strukturen entsprechen den Granularitäten $Tage$ und $Woche$.

Die Semantik eines solchen Prädikats ist vergleichbar mit der Semantik der zuvor vorgestellten Gruppierungsoperation in algebraischen Ansätzen. Ein anderes Prädikat wiederum, nämlich $Group_pSkip_q(G_1, G_2)$, ist mit Selektionsoperationen vergleichbar. Es ist genau dann erfüllt, wenn jedes Granule von G_2 aus genau p aufeinander folgenden Granules von G_1 besteht und bis zum nächsten Granule jeweils q Granules von G_1 ausgelassen werden. Die Granularität *Arbeitswoche* entspricht einer erfüllenden Belegung des Ausdrucks $Arbeitswoche(\mathbf{x}) = Group_5Skip_2(\text{Tage}(\mathbf{y}), \mathbf{x})$. Die Gabe von Medikament A in einer Chemotherapie (d.h. in den ersten 14 Tagen eines jeden Zyklus) entspricht einer erfüllenden Belegung des Ausdruck $Chemo(\mathbf{x}) = Group_{14}Skip_{14}(Tage(\mathbf{y}), \mathbf{x})$.

Allerdings weisen vorliegende Vorschläge für constraint-basierte Beschreibungen vor allem Schwächen im Umgang mit nichtäquidistanten Granularitäten auf, etwa $Monate$ bzw. bei Ausnahmen in Periodizitätsschemas. Insbesondere die Anzahl der notwendigen Prädikate im Falle komplexer periodischer Spezifikationen bildet hier ein Problem. Zum Beispiel wird der Schaltmonat Februar durch explizite Auflistung aller Monatslängen innerhalb der Periode von 400 Jahren beschrieben[19]. Hierzu werden im obigen Beispiel die Monatslängen, d.h. die Abstände zwischen den Start- und Endpunkten der Granules durch das Prädikat $M_n(x)$ beschrieben. Somit entspricht die Kette von 400 Prädikaten, d.h. $Monate(\mathbf{x}) = M_{31}(M_{28} \ldots (M_{31}(M_{29} \ldots (\mathbf{x}))))$, der Granularität *Monate*. Eine solche Vorgehensweise ist kein Einzelfall, sondern findet sich in ähnlicher Form in allen vorliegenden constraint-basierten Ansätzen. Im Zusammenhang mit Ausnahmen in Periodizitätsschemas führt dieses Vorgehen dazu, dass das "Resultat" einer Ausnahme, d.h. die Menge der variierenden Abstände einer Periode, zunächst explizit "ausgerechnet" und dann deklarativ mit Prädikaten beschrieben werden muss. Ein kompaktes Vorgehen im Sinne einer deklarativen Beschreibung der Änderungen für "ausgewählte" Zyklen, ohne eine damit verbundene explizite Auflistung des Resultats bzw. entsprechender Prädikate, wird bisher nicht unterstützt.

Zusammenfassend betrachtet sind die vorliegenden algebraischen und constraint-basierten Vorschläge zur Beschreibung von Periodizitätsschemas nur sehr eingeschränkt taug-

lich. In keinem der Vorschläge findet sich eine adäquate und generische Unterstützung von Ausnahmen in Periodizitätsschemas. Das heißt, es fehlt sowohl eine einheitliche Beschreibung von äquidistanten und nichtäquidistanten Auswahlschemas, als auch eine Unterstützung von Auswahloperationen sowie die Bereitstellung generischer Änderungsoperationen.

Allerdings bildet die Erweiterung dieser Vorschläge, etwa um neue Operationen oder adäquate Prädikate, nicht das Hauptproblem. Das eigentliche Problem ist eine geeignete operationale Unterstützung für Periodizitätsschemas. Es werden Algorithmen benötigt, die effizient algebraische bzw. logische Ausdrücke evaluieren können. Entsprechende Algorithmen auch bei vollständiger Ausdrucksstärke von Schemas zur Verfügung zu stellen, entspricht jedoch keiner trivialen Erweiterung, wie im nachfolgenden Kapitel noch gezeigt wird.

2.3 Realisierungsansätze der operationalen Unterstützung

Die Realisierung einer geeigneten operationalen Unterstützung für periodische Spezifikationen bildet eine der größten Herausforderungen. Wie in Abschnitt 2.1.3 gezeigt, können die drei Klassen *Ableitungs-*, *Überprüfungs-* und *Vergleichsoperationen* unterschieden werden. Im folgenden Abschnitt werden verschiedene generelle Möglichkeiten für die Realisierung dieser Operationsklassen vorgestellt sowie die Vor- und Nachteile diskutiert.

2.3.1 Teilmaterialisierung

Eine Basisstrategie zur Realisierung von *Überprüfungs- und Vergleichsoperationen* stellt die Teilmaterialisierung einer periodischen Spezifikationen dar. Für einen vorgegeben begrenzten Zeitbereich, etwa dem Ereignishorizont bei Vergleichsoperationen (siehe Abschnitt 2.1.3), werden konkrete zeitliche Werte der Spezifikation erzeugt. Diese expliziten Wertemengen werden dann gemäß der Operationssemantik entweder auf einzelne Werte überprüft oder bezogen auf Gleichheit, Ungleichheit oder sonstige Eigenschaften verglichen. Voraussetzung für eine solche Strategie ist eine adäquate Realisierung von *Ableitungsoperationen* zur Erzeugung der Wertemengen ausgehend von der logischen Beschreibung (siehe später). Die Teilmaterialisierung entspricht im Kern dem Vorgehen wie es etwa in heutigen Kalender- bzw. Zeitplanungssystemen überwiegend anzutreffen ist.

Bei einer Teilmaterialisierung ist es nicht sinnvoll, speziell bei großer Anzahl an Spezifikationen, für jeden einzelnen Vergleich eine ”on-demand” Materialisierung zum Vergleichszeitpunkt vorzunehmen. Ein solches Vorgehen würde zu einem sehr hohen Aufwand pro Vergleich führen und erweist sich darüber hinaus für später noch benötigte Zugriffs- und Indexstrukturen als hinderlich. Stattdessen gilt es, die für einen begrenzten Zeitbereich relevanten Spezifikationen ”vorzumaterialisieren”. Zum Vergleichszeitpunkt müssen dann

lediglich die vormaterialisierten Wertemengen verglichen werden. Grundvoraussetzung für diese Teilmaterialisierungsstrategie ist eine (zeitliche) Synchronisation des aktuell gültigen (begrenzten) Zeitbereichs mit den im Zeitbereich relevanten periodischen Spezifikationen und ihren Wertemengen. Allerdings bedarf es aufgrund des Fortschreitens der Zeit und der damit verbundenen "Verschiebung" des (begrenzten) Zeitbereichs einer kontinuierlichen Synchronisation. Das heißt, es müssen kontinuierlich "Nachmaterialisierung" und Löschungen von Werten vorgenommen werden, um die entsprechenden Wertemengen der (relevanten) periodischen Spezifikationen auf dem aktuellen Stand zu halten.

Die Teilmaterialisierung kann grundsätzlich die Realisierung von Überprüfungs- und Vergleichsoperationen gewährleisten. Allerdings weist die Teilmaterialisierung a priori verschiedene Nachteile und Einschränkungen auf:

1. Der bei einer kontinuierlichen Synchronisation anfallende Verwaltungsaufwand, entweder auf Seiten des Informationssystems oder in den zugrundeliegenden Datenbanken ist sehr hoch und aufgrund des Fortschreitens der Zeit beständig vorhanden. Ähnlich verhält es sich auch bei nachträglichen Änderungen an entsprechenden Spezifikationen (Updateproblem). Insbesondere im Zusammenhang mit einer umfangreichen Menge an periodischen Spezifikationen, wie in prozessorientierten Informationssystemen, stellt dieser Verwaltungsaufwand eine kritische Größe dar.
2. Eine Teilmaterialisierung erlaubt es kaum, vorkommenden individuellen Abweichungen in der Größe des Zeitbereichs bzw. Ereignishorizonts zu begegnen. Einzelne über den Horizont hinausreichende Konfliktprüfungen, etwa bei wichtigen "VIP-Prozessen", können somit lediglich durch eine Nachmaterialisierung realisiert werden. Dies führt erneut zu einem sehr hohen Aufwand pro Vergleich.
3. Das Laufzeit- und Speicherverhalten dieser Realisierungsstrategie ist unmittelbar von der Größe der zeitlichen Begrenzung sowie der Anzahl enthaltener Werte abhängig. Je nach Anwendung, Art der periodischen Spezifikation sowie Größe der Begrenzung kann dieses Verhalten bereits schon für die einzelne Spezifikation, insbesondere jedoch für die Gesamtheit an periodischen Spezifikationen kritische Werte erreichen.

Im Gegensatz zur Teilmaterialisierung, d.h. eine zeitabhängige Repräsentation der periodischen Spezifikation durch dynamische Wertemengen, stellt eine "statische" Repräsentation eine wünschenswertere Alternative dar. Das heißt die periodischen Spezifikationen werden zeitunabhängig repräsentiert, können jedoch zeitabhängig bezüglich eines Vergleichs abgefragt werden. Bei dieser Repräsentationsalternative entfällt somit der hohe Verwaltungsaufwand für dynamische Wertemengen und die weiteren damit verbundenen Probleme.

Im Folgenden werden zwei Ansatzklassen von zeitunabhängigen Repräsentationsformen und entsprechende Operationsrealisierungen vorgestellt: die Verwendung der formalen Beschreibung und die Verwendung von Wiederholungsmustern.

2.3.2 Verwendung der formalen Beschreibung

Die Grundlage der folgenden Realisierungsstrategien bildet die logische Beschreibung der periodischen Spezifikationen, d.h. die temporalen Operationen beziehen sich direkt auf die algebraischen Operationsketten oder logischen Formeln eines Periodizitätsschemas. Allerdings bedarf es für die algebraischen und constraint-basierten Beschreibungen geeigneter Evaluationsstrategien, so dass die verschiedenen Operationsklassen auch effizient realisiert werden können.

2.3.2.1 Algebra- bzw. Symbolbasierte Realisierung

Diese Realisierung basiert auf der Auswertung der Operationsketten einer algebraischen Beschreibung von Periodizitätsschemas.

Ableitungs- und Überprüfungsoperationen: Grundidee ist, "Umrechnungsoperationen" für jede einzelne Beschreibungsoperation bereitzustellen. Diese definieren dann explizit, wie Angaben von Zyklenelementen des "Ausgabeschemas" in Werte des zugrundeliegenden "Eingabeschemas" einer Beschreibungsoperation umgerechnet werden können und umgekehrt. Die Umrechnungen der einzelnen Beschreibungsoperationen bilden die Grundlage für Operationen über dem gesamten algebraischen Ausdruck eines Periodizitätsschemas.

Bei Ableitungsoperationen gilt es für eine Gruppierungsoperation wie $G = Group_m(H)$ das Intervall von Werten aus H zu ermitteln, aus dem sich ein gegebener Wert in G zusammensetzt. Bei der "Umrechnung" $G(i) = [H_{first}, H_{last}]$ von einem Wert i aus G, ergibt sich für das Intervall in H als erster Wert $H_{first} = i * m$ bzw. als letzter Wert $H_{last} = H_{first} + (m - 1)$. So entspricht der 3. Tag der Granularität $Tage = Group_{24}(Stunden)$ dem Intervall $Tage(3) = [72, 95]$ in $Stunden$. Für die weiteren Gruppierungs- und Selektionsoperationen können vergleichbare, wenn auch komplexere Umrechnungen gefunden werden. Ausgehend von einer konkreten Angabe zu einem Zykluselement bzw. Granule können entsprechende Umrechnungen rekursiv für alle Beschreibungsoperationen durchgeführt werden, solange bis die "tiefste" mit dem Wertebereich korrespondierende Ebene erreicht ist. Auf dieser Ebene, d.h. auf Basis der Umrechnung der "tiefsten" bzw. "ersten" Beschreibungsoperation, können konkrete Werte aus dem zeitlichen Wertebereich abgeleitet werden. Im Fall von $Stunden$ als tiefste Ebene ergibt sich für $Woche(3)$ mit $Woche = Group_7(Tage)$ etwa das Intervall $[Tage(21), Tage(27)]$ auf Stundenebene bzw. im zugrundeliegenden Wertebereich der Intervall $[504, 671]$.

Bei Überprüfungsoperationen wird das gleiche Vorgehen verwendet, jedoch in Gegenrichtung, d.h. ausgehend von Angaben des "Eingabeschemas" ermittelt man die Angaben zum "Ausgabeschema" einer Beschreibungsoperation. Allerdings ist bei Periodizitätsschemas bzw. bei Granularitäten mit "Löchern" nicht jeder Wert des Wertebereichs

einem Zykluselement bzw. Granule der Granularität zuordbar. Beispielsweise sind zeitliche Werte, die mit *Samstag* oder *Sonntag* korrelieren, nicht in der Granularität *Arbeitswoche* enthalten. In solchen Fällen schlägt die Umwandlung an einer Stelle in der algebraischen Operationskette fehl. Nur wenn bis zur "obersten Ebene" bzw. letzten Beschreibungsoperation stets eine Umwandlung des Wertes gefunden werden kann, bedeutet dies, dass der Wert im Periodizitätsschema bzw. Granularität enthalten und die Überprüfungsoperation erfolgreich sind.

Die symbol-basierte Realisierung bildet eine adäquate Strategie für die Ableitungs- und Überprüfungsoperationen periodischer Spezifikationen. Auch wenn diese ausschließlich auf dem algebraisch beschriebenen Periodizitätsschema basiert, können die zusätzlichen Aspekte einer Spezifikation, d.h. Gültigkeitszeitraum und individuelle Instanziierungszeitpunkte, durch entsprechende Vor- bzw. Nachbereitungsschritte berücksichtigt werden.

Allerdings sind bei einer symbol-basierten Realisierung im Fall großer Mengen periodischer Spezifikationen möglichst effiziente Einzelumrechnungen zwingend notwendig. Des Weiteren müssen eventuell bestehende äquivalente oder redundante Umrechnungen in Bezug auf den gesamten algebraischen Ausdruck erkannt und eingespart werden können. Die in der Literatur vorliegenden Vorschläge zur symbol-basierten Realisierung von algebraischen Beschreibungen (z.B. [8, 65, 66]) weisen in diesem Zusammenhang Schwächen auf. Aufgrund der "semantisch reichhaltigen" Beschreibungsoperationen (siehe Abschnitt 2.2.1), d.h. der "implizit" in der Operation angenommenen Schritte zur Bildung des Ausgabeschemas, ergeben sich ebenfalls komplexe Umrechnungen. So basiert etwa in der Operation *Alter*() (siehe Abschnitt 2.2.1) die Umrechnung auf einem Näherungsverfahren, das zwar korrekt, jedoch sehr aufwendig in seiner Ausführung ist (z.B. [8, 65]). Darüber hinaus liegt häufig eine Vielzahl an einzelnen Beschreibungsoperationen mit vergleichbarer bzw. ähnlicher Semantik, jedoch jeweils eigenen Einzelumrechnungen, vor. In Bezug auf den gesamten algebraischen Ausdrucks eines Periodizitätsschemas ermöglichen einzelne komplexe Umrechnungen oder eine Vielzahl separater Einzelumrechnungen es kaum, äquivalente oder redundante Schritte bzw. Umrechnungen zu erkennen und einzusparen.

Vergleichsoperationen: Eine Ansatz zur algebraischen Realisierung von Vergleichs- bzw. Mengenoperationen ist vergleichbar mit dem vorherigen Vorgehen bei Ableitungs- und Überprüfungsoperationen (z.B. [81, 84]). Es werden für jede Beschreibungsoperation formale qualitative Beziehungen zwischen dem "Ein- und Ausgabeschema" angegeben. So führt die Selektionsoperation *Select* zu einer Teilmengenbeziehung des Ein- und Ausgabeschemas, etwa der Ausdruck $Arbeitswoche = Select_{1-5}(Woche, Tage)$ zur Beziehung $Arbeitswoche \subseteq Woche$. Ein weiteres Beispiel bilden die Angaben "*Jeden Montag*" ($Mo = select_1(Woche, Tage)$) und "*Jeden 2. Montag*" ($2tenMo = select_1(Group_2(Mo), Tage)$) mit der Beziehung $2tenMo \subseteq Mo$.

Bei einer Mengenoperation zwischen zwei Schemas werden deren direkte bzw. transitiv

ableitbare Beziehungen ermittelt. Für jede Mengenoperation sind Regeln hinterlegt, wie basierend auf den qualitativen Beziehungen der Operanden das Ergebnis der Operation unmittelbar abgeleitet werden kann. Beispielsweise kann bei einer Schnittoperation und einer Teilmengenbeziehung zwischen beiden Operanden G_1, G_2 das Ergebnis durch die Regel $G_1 \subseteq G_2 \Rightarrow G_1 \cap G_2 = G_1$ abgeleitet werden bzw. bei einer Vereinigungsoperation durch $G_1 \cup G_2 = G_2$. So führt im oberen Beispiel eine Mengenoperation zwischen "*Jeden Montag*" und "*Jeden 2. Montag*" aufgrund der Teilmengenbeziehung $2tenMo \subseteq Mo$ zu dem Ergebnis $Mo \cap 2tenMo = 2tenMo$ bzw. $Mo \cup 2tenMo = Mo$.

Allerdings ist dieses Vorgehen nicht immer anwendbar, da sich nicht in jedem Fall geeignete Beziehungen zwischen algebraischen Ausdrücken herleiten lassen. Insbesondere gilt dies bei Ausdrücken mit teilweise "überlappenden" Wertemengen, wie "*Jeden Montag* " und "*Jeden 1. Tag im Monat*". In solchen Fällen reichen qualitative "Vereinfachungsregeln" nicht aus, um das konkrete Ergebnis einer Mengenoperation der beiden überlappenden Wertemengen anzugeben. Genauer gesagt, es müsste jede mögliche Überlappung zweier Mengen durch eine entsprechende qualitative Beziehung beschrieben werden, was jedoch kaum realisierbar ist.

Darüber hinaus erlaubt diese Realisierungsmöglichkeit lediglich den Vergleich kalenderorientierter Spezifikationen. Aufgrund der individuellen Instanziierungszeitpunkte bei zyklen-orientierten Spezifikationen können identische Operationsketten gleiche, disjunkte oder überlappende Wertemengen bilden. Dies jedoch alleine anhand der Operationsketten erkennen zu können, ist nicht möglich. Für zyklen-orientierte Spezifikationen ist eine solche Vorgehensweise deshalb nicht anwendbar.

Eine alternative Möglichkeit zur algebraischen Realisierung von Vergleichsoperationen basiert auf dem Slice-Formalismus (siehe Abschnitt 2.2.1). Ein Slice ist ein Ausdruck der Form $O_n.G_n + ... + O_1.G_1$ wobei $O_i.G_i$ jeweils eine Selektionsoperation für die Granularität G_i beschreibt. Der Ausdruck $*.Jahre + \{1,2\}.Monate + \{1\}.Tage$ etwa entspricht der Angabe "*an jeden 1. Tag des 1. und 2. Monats (Jan, Feb) eines jeden Jahres*".

Die Realisierung von Mengenoperationen zwischen zwei Sliceausdrücken basiert auf Mengenoperationen zwischen den Mengen O_i, O_k der beiden identischen Teilausdrücke $O_i.G_j$ und $O_k.G_j$. Das Ergebnis einer Schnittoperation etwa zwischen dem Sliceausdruck $*.Monate + \{1,8,10\}.Tage$ und dem Ausdruck $*.Monate + \{3,8,9\}.Tage$, ist der Ausdruck $*.Monate + \{8\}.Tage$. Bei Ausdrücken, in denen die jeweiligen Granularitätsreihenfolgen nicht zueinander "kompatibel" sind, bedarf es zusätzlicher Umwandlungen, wie zum Beispiel bei den Ausdrücken $Monate + 5.Tage$ und $Monate + Woche + 1.Tage$. Der erste Ausdruck beschreibt die Angabe "*jeden 5. Tag eines Monat*s" und der zweite Ausdruck die Angabe "*jeden Montag*". Die Wertemengen beider Ausdrücke überlappen sich teilweise. Zur Erkennung der gemeinsamen Werte muss die Beziehung zwischen $Wochen$ und $Monaten$ explizit durch eine entsprechende Umwandlung bzw. Erweiterung von einem der beiden Ausdrücke realisiert werden. In obigem Beispiel kann der zweite Ausdruck umgewandelt werden in $Monate + \{\}.Tage$ und innerhalb der Periode

von $Wochen$ und $Monaten$ (28 Jahre) erweitert bzw. explizit aufgeführt werden. Es wird eine Menge von Ausdrücken der Form: $1.Monate + 3.Tage \cup 2.Monate + 6.Tage \cup \ldots \cup 336.Monate + 6.Tage$ erzeugt. Dieser erweiterte Ausdruck wird dann mit dem ersten Ausdruck $Monate + 5.Tage$ geschnitten.

Die Realisierung von Vergleichsoperationen basierend auf normierten Ausdrücken für Selektionsoperationen ist bei kalender-orientierten Spezifikationen grundsätzlich möglich. Allerdings kann dabei (wie oben gezeigt) eine sehr umfangreiche Anzahl an Ausdrücken entstehen, was sich bezüglich der Effizienz des Verfahrens als großer Nachteil erweist. Allerdings bildet das Kernproblem dieser Vorgehensweise erneut die Tatsache, dass lediglich kalender-orientierte Spezifikationen unterstützt werden können. Die individuellen Instanziierungszeitpunkte von zyklen-orientierten Spezifikationen bzw. die resultierenden Probleme (siehe oben) können auch mit dieser algebraischen Realisierungsmöglichkeit nicht adressiert werden. Die praktische Anwendbarkeit der vorliegenden Vorschläge (z.B. [64, 12]) ist daher stark beschränkt.

2.3.2.2 Constraint-basierte Realisierung

Eine Realisierung von Operationen auf Basis der constraint-basierten Beschreibung von periodischen Spezifikationen basiert im Wesentlichen auf der Überprüfung der Erfüllbarkeit entsprechender logischer Ausdrücke.

Überprüfungsoperationen: Zur Realisierung dieser Operation gilt es, für einen gegebenen zeitlichen Wert zu überprüfen, ob dieser eine erfüllende Belegung für den logischen Ausdruck der periodischen Spezifikation darstellt. Es hängt von der Art der logischen Terme bzw. der verwendeten Logik ab, ob eine solche Überprüfung grundsätzlich effizient erfolgen kann.

Bei Verwendung einfacher logischer Terme kann die Überprüfung auf Basis der semantischen Äquivalenz zu linearen Geradengleichungen erfolgen. Ein gegebener Wert x für den Ausdruck $periodic_{28,1}(x) \vee periodic_{28,8}(x)$ muss eine Lösung der Gleichungen $x = 28 * i + 1$ oder $x = 28 * i + 8$ darstellen, was hier effizient überprüfbar ist. Darüber hinaus ermöglichen die Gleichungen es, konkrete Werte einer periodischen Spezifikation abzuleiten. Im Gegensatz zur allgemeinen Realisierung von Ableitungsoperationen (siehe unten) ist dies bei einfachen logischen Ausdrücken somit grundsätzlich möglich.

Wie bereits in Abschnitt 2.2.2 erwähnt, reichen einfache logische Ausdrücke allerdings nicht aus, unterschiedliche Granularitäten adäquat zu unterstützen. Zur (impliziten) Unterstützung unterschiedlicher Granularitäten werden erweiterte Logiken verwendet. Im Gegensatz zu den oberen "einfachen" Logiken, weist jedoch die Überprüfung erfüllender Belegungen in den erweiterten Logiken grundsätzlich hohe Komplexität auf (z.B. PSPACE [19]).

Eine Realisierung von Überprüfungsoperationen ist lediglich für "einfache" constraint-basierte Beschreibungen, d.h. Ansätze mit expliziter Unterstützung von Granularitäten,

grundsätzlich möglich (z.B. [86, 87]). Erweiterte Logiken, d.h. Ansätze mit impliziter Granularitätsunterstützung, sind aufgrund der Komplexität der Realisierung ungeeignet (z.B. [18, 19, 34]).

Ableitungs- und Vergleichsoperationen: Bei diesen Operationen basiert die Realisierung auf dem Finden erfüllender Belegungen für die logischen Ausdrücke periodischer Spezifikationen. So entspricht etwa die Vergleichs- bzw. Schnittoperation (Konjunktion) der "*Gabe von Medikament B am 1. und 8.Tag eines Chemotherapiezyklus von 28 Tagen*" und "*Jeden 5. Tag*" dem Finden von Werten x, die den logischen Ausdruck erfüllen: $(periodic_{28,1}(x) \vee periodic_{28,8}(x)) \wedge periodic_{5,1}(x)$.

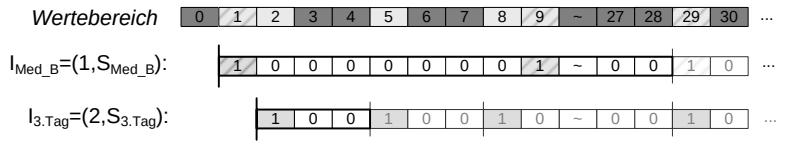
Im Allgemeinen sind für das Finden erfüllender Belegungen (Erfüllbarkeitsproblem) lediglich Verfahren mit nicht-polynomialer Komplexität bekannt, die für den praktischen Einsatz nicht geeignet sind. Jedoch ermöglichen spezielle Verfahren, wie *Temporal Constraint Networks* (TCN), für bestimmte Formen von (zeitlichen) Contraints, die Erfüllbarkeit effektiver zu entscheiden sowie eine Lösungsmenge anzugeben. In diesem Zusammenhang finden sich in der Literatur Vorschläge zur Auswertung contraint-basierter Beschreibungen von periodischen Spezifikationen bzw. Granularitäten auf Basis von TCN [8, 85, 80, 82, 1, 84, 83, 25].

Allerdings weist die Vorgehensweise bei diesen Vorschlägen ein grundsätzliches Problem auf. In diesen Arbeiten werden die "periodischen" Constraints, wie $periodic_{28,1}(x)$, nicht unmittelbar ausgewertet. Stattdessen werden solche Constraints lediglich für einen begrenzten Zeitbereich in "klassische" Zeitconstraints (Abstandsconstraints) "umgewandelt" bzw. expandiert. Dieses Vorgehen entspricht jedoch dem Vorgehen der Strategie der Teilmaterialisierung von Werten. Anstelle konkreter Werte wird eine Menge an Abstandsconstraints in einem bestimmten Zeitbereich materialisiert. Somit weist ein solches Vorgehen die gleichen Probleme wie eine Teilmaterialisierung auf.

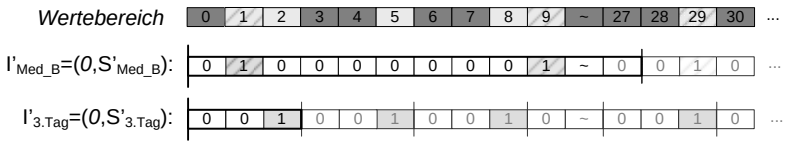
2.3.3 Musterbasierte Realisierungsformen

Die Grundlage dieser Realisierungsform bilden Wiederholungsmuster, die auf Basis der logischen Beschreibung periodischer Spezifikationen erzeugt werden (z.B. in [11, 10] am Beispiel der Calender Algebra [65] gezeigt). Ein Wiederholungsmuster repräsentiert die kleinste Periode einer Spezifikation, d.h. den kleinsten Zeitraum der Spezifikation, nach dem sich die möglichen Abstände zwischen Werten sowie deren Reihenfolgen wiederholen.

Ein Wiederholungsmuster "verläuft" über einen (diskreten) Wertebereich, wobei die Positionen des Musters mit den Werten des Wertebereichs korrespondieren. Die konkreten Belegungen der Positionen, d.h. die Stützstellen des Musters, treffen Aussagen, ob der zur Position korrespondierende Wert in der Spezifikation enthalten ist (oder nicht). Die Länge des Musters, d.h. die Periodenlänge, entspricht der kleinsten Periode der



(a) unterschiedlicher Instanziierungspunkt



(b) vereinheitlichter Instanziierungspunkt

Abbildung 2.2: Bitvektordarstellung zweier Wiederholungsmuster

Spezifikation. Beispielsweise ist in Abbildung 2.2a eine Bitvektordarstellung der Wiederholungsmuster für die Beispiele "*Gabe vom Medikament B* (S_{Med_B})" (Periodenlänge=28, Instanziierungspkt.=1) und "*Jeden 3. Wert* ($S_{3.Tag}$)" (Periodenlänge=3, Instanziierungspkt.=2) dargestellt.

Ein Vorteil muster-basierter Realisierungen, insbesondere in Hinblick auf Vergleichsoperationen, bildet die Möglichkeit, unterschiedliche Instanziierungszeitpunkte der Spezifikationen "herauszurechnen" bzw. zu vereinheitlichen. Ermöglicht wird dies durch eine Verschiebung auf einen gewünschten, einheitlichen Instanziierungszeitpunkt bei gleichzeitiger zyklischer Rotation des Musters (jeder Wert der rechts "heraus fällt", wird links wieder eingefügt). Die Rotation ist werteäquivalent, d.h. ausgehend von dem vereinheitlichten Instanziierungspunkt ist die Wertemenge des rotierten Musters identisch zur ursprünglichen Wertemenge. Der ursprüngliche Instanziierungszeitpunkt dient später lediglich als untere Schranke, beispielsweise für Ableitungsoperationen. In Abbildung 2.2b ist die Vereinheitlichung auf den Instanziierungspunkt=0 für das Beispiele aus Abbildung 2.2a dargestellt.

Allerdings können lediglich Spezifikationen mit periodischen Ausnahmen durch Wiederholungsmuster repräsentiert werden. Spezifikationen mit nicht-periodischen Ausnahmen, d.h. Ausnahmen, bei denen nur eine endliche Menge an Zyklen oder Elementen betroffen ist, können mit Wiederholungsmustern nicht dargestellt werden. Entsprechende Spezifikationen weisen keine kleinste Periode auf, d.h. es kann kein einzelnes konstantes Muster angegeben werden, das sich über den gesamten Zeitbereich wiederholt. Nicht-periodische Ausnahmen erfordern in diesem Fall eine andere Repräsentation bzw. eine separate Behandlung.

Die Realisierung von Operationen ist abhängig von der konkreten Repräsentation des Musters. In der Literatur finden sich zwei grundsätzliche Repräsentationsformen, *Lineare-* und *Automaten-basierte Musterrepräsentation*, die im Folgenden vorgestellt werden.

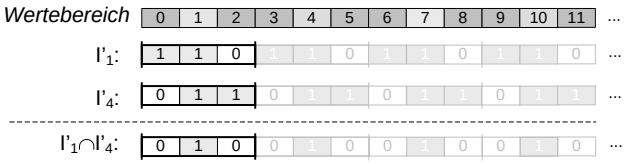


Abbildung 2.3: Schnittoperation von Wiederholungsmustern mit gleicher Periodenlänge

2.3.3.1 Lineare Musterrepräsentation

Bei dieser Repräsentationsform wird ein Wiederholungsmuster explizit durch seine erste Periode dargestellt. So finden sich Vorschläge zur Repräsentation des Wiederholungsmuster durch einfache (äquidistante) lineare Geradengleichungen [45, 86, 46, 6, 39, 66] oder als Bit- bzw. Stringvektor [94, 11, 10].

Ableitungs- und Überprüfungsoperationen: Unabhängig von der konkreten Repräsentation im Einzelnen, kann aufgrund der Äquivalenz zu einfachen mathematischen Strukturen, die Realisierung solcher Operationen stets als "Berechnung" einer Menge linearer Gleichungen verstanden werden. So kann das Wiederholungsmuster I'_{Med_B} aus Abb. 2.2b durch die linearen Gleichungen $g_1 : x = 28 * i + 2$ und $g_2 : x = 28 * i + 9$ repräsentiert werden (siehe Abschnitt 2.3.2.2). Neben der Ableitungsoperation auf Basis der Indexvariable i kann bei entsprechender Umstellung der Gleichungen auch die Überprüfungsoperation effizient realisiert werden.

Vergleichsoperationen: Generell gilt es beim Vergleich von Wiederholungsmustern zwei Fälle zu unterscheiden: Mustern mit identischer und mit unterschiedlicher Periodenlänge.

Bei identischer bzw. vielfacher Periodenlänge können die beiden bezüglich des Instanzierungszeitpunkts vereinheitlichten Wiederholungsmuster unmittelbar miteinander verglichen werden. Das heißt, die jeweiligen Stützstellen des Wiederholungsmusters werden gemäß der Semantik des Vergleichs direkt miteinander verglichen. Hierbei kann ebenfalls leicht ein Ergebnismuster für die quantitative Vergleichsauswertung angegeben werden, wie in Abbildung 2.3 anhand von Wiederholungsmustern mit Periodenlänge 3 dargestellt.

Für Wiederholungsmuster mit unterschiedlicher Periodenlänge findet sich, unabhängig von der konkreten Repräsentation, stets das gleiche Vorgehen. Es wird die *kleinste gemeinsame Periodenlänge* (least common period - LCP) der beiden Wiederholungsmuster gebildet. Das LCP entspricht dem kleinsten gemeinsamen Vielfachen der Periodenlängen der Muster und ist der Zeitbereich, den beide Muster nach ganzzahliger Anzahl von Wiederholungen durchlaufen haben. Die Muster werden innerhalb des LCP erweitert

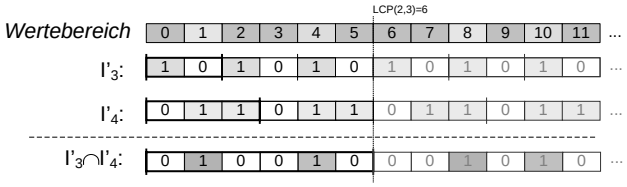


Abbildung 2.4: Schnittoperation von Wiederholungsmustern mit unterschiedlicher Periodenlänge

bzw. explizit aufgeführt. Ziel ist es, die unterschiedlichen Periodenlängen so zu verlängern, dass erneut identische Periodenlängen entstehen. Nach der Verlängerung kann die gleiche Realisierung wie im ersten trivialen Fall verwendet werden. In Abbildung 2.4 sind zwei Muster mit Periodenlänge 3 und Periodenlänge 2 dargestellt, sowie das Ergebnismuster der Schnittoperation mit Periodenlänge $6 = LCP(2, 3)$.

Das Vorgehen linearer Musterrepräsentationen gewährleistet es grundsätzlich, sowohl Ableitungs- und Überprüfungsoperationen als auch Vergleiche zu realisieren.

Allerdings hängt es von der Art der periodischen Spezifikation ab, ob die Realisierung im Einzelfall effizient erfolgen kann. Bei linearen Musterrepräsentationen ist die Effizienz der operationalen Unterstützung unmittelbar von der Größe des Musters abhängig, d.h. von der Periodenlänge und Anzahl an Stützstellen. Bereits bei einfachen, praxisrelevanten Beispielen ergeben sich Größenordnungen, die eine effiziente Auswertung kaum ermöglichen. Beispielsweise beträgt die Periodenlänge der Angabe "*Jeden 1. Tag im Monat*" 400 *Jahre* (146097 *Tage)*. Die Anzahl konkret belegter Stützstellen beträgt 4800 Werte (12 pro Jahr * 400 Jahre). Diese Einzelangabe entspricht einer Menge von 4800 linearen Gleichungen bzw. einem Bitvektor der Länge 146097 Bits ($\approx$ 4500 Integerwerte mit 32Bit). Bei einem Vergleich der oberen Angabe mit einer Chemotherapie der Länge von 28 Tagen beträgt das $LCP(28, 146097) = 584.388$ *Tage*, d.h. das Ergebnismuster weist eine Periodenlänge von ca. 2000 Jahren auf.

2.3.3.2 Automaten-basierte Repräsentation

Die Realisierung von Operationen wird in diesen Ansätzen auf die Theorien formaler Sprachen und Automaten zurückgeführt. Ein Muster wird als Wort der Sprache verstanden, die durch den Automaten repräsentiert wird. Die Automatendarstellung des Musters bildet die Grundlage für die Operationsrealisierung.

In der Literatur finden sich Ansätze, die ein Wiederholungsmuster bzw. jede Position des Musters durch einen entsprechenden Zustand im Automaten repräsentieren (z.B. [31, 14, 52]). Die Anzahl der Zustände korreliert hierbei mit der Periodenlänge des Musters. Die Symbole des Musters bilden die Zustandsübergangsfunktion.

Allerdings gibt es auch Vorschläge, die eine kompaktere Darstellung des Wiederholungsmusters ermöglichen [32, 31]. Hierbei werden unterschiedlich priorisierte Zustandsübergangsfunktionen verwendet, die Regelmäßigkeiten innerhalb des Wiederholungsmusters ausnutzen und deshalb mit weniger Zuständen auskommen. Die Semantik dieser Übergangsfunktionen ist mit Zählern der Art "*5 mal Symbol 0; 2 mal Symbol 1*" vergleichbar, d.h., sie entspricht einem automaten-basierten "*run length encoding*" des Musters.

Ableitungs- und Überprüfungsoperationen: Diese Operationen lassen sich durch Simulieren des Automaten realisieren. Ausgehend von dem Instanziierungspunkt einer periodischen Spezifikation wird das Muster auf Basis des Automaten "durchlaufen". Hierbei werden entweder konkrete Werte abgeleitet oder es wird überprüft, ob sich ein entsprechender Wert ergibt. Grundsätzlich ermöglicht eine Simulation des Automaten die Realisierung entsprechender Operationen. Allerdings handelt es sich um ein sehr aufwendiges Vorgehen, das bereits bei einer einzelnen Angabe zu Effizienzproblemen führt. Darüber hinaus entsprechen nicht-kompakte Automatendarstellungen einer Form der linearen Musterrepräsentationen und weisen die gleichen Probleme wie im vorherigen Abschnitt besprochen auf.

Vergleichsoperationen: Nicht-kompakte Automatendarstellungen sind äquivalent zu endlichen Automaten, wodurch grundsätzlich Algorithmen zur Realisierung von Vergleichs- bzw. Mengenoperationen angeben werden können. Zur Konstruktion des Ergebnisautomaten für den Schnitt bzw. die Vereinigung zweier Automaten wird zunächst der Kreuzproduktautomat gebildet, d.h. das Kreuzprodukt der beiden Zustandsmengen. Anschließend werden, je nach Semantik der Operation, die Endzustände angepasst. Allerdings fehlen entsprechende Betrachtungen bzw. Aussagen zur Realisierbarkeit solcher Operationen in Bezug auf kompakte Automatendarstellungen.

Die automaten-basierte Realisierung von Vergleichsoperationen ist mit existierenden Vorschlägen nur im Fall nicht-kompakter Automatendarstellung möglich. Allerdings sind der Umfang der Darstellung sowie das Vorgehen bei Vergleichsoperationen vergleichbar mit linearen Musterrepräsentationen. So entspricht die Bildung des Kreuzproduktautomaten im Kern der Bildung des LCP zweier Muster. Das heißt, die resultierende Anzahl von Zuständen des Produktautomaten ist identisch mit dem Produkt bzw. LCP der beiden Periodenlängen des Musters. Jede der Aussagen zu linearen Musterrepräsentationen gilt deshalb mindestens genauso stark für eine entsprechende Automatendarstellung. Aufgrund der aufwendigeren Graphdarstellung der Automaten sind entsprechende Aussagen eher noch schwerwiegender.

Die vollständige operationale Unterstützung von periodischen Spezifikationen kann durch eine musterbasierte Realisierung grundsätzlich gewährleistet werden. Allerdings fehlt es an adäquaten Ansätzen im Zusammenhang mit Wiederholungsmustern mit sehr langen Periodenlängen. Aufgrund des Repräsentationsumfangs und dem damit verbundenen operationalen Aufwand sind vorliegende Vorschläge für die Praxis nicht geeignet.

In Hinblick auf die Praxis erweist sich dieses Ergebnis deshalb als kritisch, da Muster mit sehr langen Periodenlängen im Allgemeinen nicht die Ausnahme, sondern vielmehr die Regel bilden. Periodische Angaben auf Basis der Granularität *Monat* sind häufig und überaus gebräuchlich. Darüber hinaus können Muster mit langen Periodenlängen relativ leicht durch Berücksichtigung von Ausnahmen in eigentlich "kurzen" Mustern (wie bei einer Chemotherapie) entstehen. Das Vorgehen zur Bildung des Wiederholungsmusters bei periodischen Spezifikationen mit Ausnahmen ist vergleichbar mit der Bildung des LCP bei Vergleichsoperationen. Beispielsweise führt die Ausnahme "*Zusätzlicher Labortest nach jedem 3. Zyklus*" zu einem Wiederholungsmuster der Länge 84 (3* 28 Tage = 84 Tage). Eine weitere zusätzliche Ausnahme, wie das "*Einfügen einer Pause von 5 Tagen nach jedem 2. Zyklus*", führt bereits zu einem Muster der Länge 178 (2*84 + 2* 5 Pause = 178 Tage). Weitere Ausnahmen an anderen Zyklen oder komplexere Auswahlschemas führen zu weiteren Verlängerungen der Periodenlänge, so dass auch in diesem Fall extrem lange Periodenlängen entstehen können.

2.4 Bewertung der bestehenden Lösungsansätze

Wir haben einen Überblick zu den Herausforderungen sowie zu den in der aktuellen Literatur vorliegenden Lösungsansätzen hinsichtlich einer systemseitigen Unterstützung von periodischen Spezifikationen gegeben. Wir fassen unsere Erkenntnisse kurz zusammen und nehmen eine Bewertung vor.

2.4.1 Umfassende Beschreibung

In Hinblick auf die benötigte Ausdrucksstärke hat sich gezeigt, dass die generische Unterstützung von Ausnahmen in Periodizitätsschemas bzw. periodischen Spezifikationen eine der wichtigsten Anforderungen darstellt. Erst die umfassende Beschreibung periodischer oder nicht-periodischer Abweichungen bzw. Änderungen von einem vorgegebenen Grundschema ermöglicht es, praxisrelevante Anforderungen zu erfüllen. Darüber hinaus bilden Ausnahmen die Grundvoraussetzung für die Verwendung unterschiedlicher Granularitäten und der Handhabung damit verbundener Kalenderphänomene (z.B. Sommer-/Winterzeit bei Fahrplänen).

Allerdings stellt die adäquate und generische Unterstützung von Ausnahmen ein großes Manko existierender Beschreibungsformalismen dar. In keinem der in der Literatur vorliegenden Ansätze finden sich adäquate Möglichkeiten, einheitlich periodische oder nicht-periodische Ausnahmen zu beschreiben. Entweder sind in den entsprechenden Formalismen Ausnahmen grundsätzlich nicht berücksichtigt oder aber nur in sehr einfachen Ausprägungen.

2.4.2 Operationale Unterstützung

In Hinblick auf die operationale Unterstützung periodischer Spezifikationen zeigt sich, dass neben den Ableitungs- und Überprüfungsoperationen insbesondere die Vergleichsoperationen eine der größten Herausforderungen darstellt. Neben der Basisstrategie der Teilmaterialisierung periodischer Spezifikationen haben wir zwei alternative Realisierungsstrategien für Operationen identifiziert. Im Gegensatz zur Teilmaterialisierung sind diese alternativen Strategien a priori nicht bzw. nicht im gleichen Maße durch einen hohen Verwaltungsaufwand aufgrund umfangreicher dynamischer Wertemengen oder bei Updates gekennzeichnet.

Die Realisierungsstrategie basierend auf der logischen Beschreibung von Spezifikationen, kann nur bei Ableitungs- und Überprüfungsoperationen den Anforderungen der operationalen Unterstützung genügen. Für Vergleichsoperationen hingegen stellt diese Strategie aufgrund ihrer sehr eingeschränkten Unterstützung einiger weniger Formen von periodischen Spezifikationen keine praktisch anwendbare Realisierungsalternative dar. Lediglich die dritte Ansatzklasse, d.h. die musterbasierte Realisierung, konnte den Anforderungen der operationalen Unterstützung grundsätzlich begegnen. Die Verwendung von Wiederholungsmustern als Repräsentant der (theoretisch) unendlichen Wertemenge einer periodischen Spezifikation stellt eine alternative Möglichkeit für die Realisierung sowohl von Ableitungs- und Überprüfungsoperationen als auch von Vergleichsoperationen dar.

2.4.3 Effiziente interne Repräsentation

Die prinzipielle Realisierbarkeit von Operationen alleine reicht nicht aus. Insbesondere im Fall einer großen Menge an periodischen Spezifikationen muss auch die Effizienz der resultierenden Implementierung berücksichtigt werden.

Wie gezeigt, können weder eine Teilmaterialisierung noch existierende Vorschläge zur musterbasierten Realisierung den Anforderungen im vollen Umfang begegnen. Das Laufzeit- und Speicherverhalten einer Teilmaterialisierung kann je nach Anwendung, Art der periodischen Spezifikation sowie Größe der Begrenzung sowohl für die einzelne als auch für die Gesamtheit an Spezifikationen kritische Werte aufweisen. Im Gegensatz dazu fehlt es bei der musterbasierten Realisierung an adäquaten Ansätzen für Wiederholungsmustern mit sehr langen Periodenlängen. Aufgrund des Repräsentationsumfangs und dem damit verbundenen operationalen Aufwand sind die vorliegenden Vorschläge nicht geeignet. Kompaktere Repräsentationen – unter Beibehaltung der Vergleichbarkeit – sind sowohl bezüglich des Speicherbedarfs des Einzelmusters als auch bezüglich der Effizienz der operationalen Unterstützung für den praktischen Einsatz unverzichtbar.

Die vorgestellten drei Punkte stellen offene Fragestellungen dar und bilden die Themenschwerpunkte der vorliegenden Arbeit. Erst die Adressierung dieser entscheidenden

Punkte ermöglicht es, periodische Spezifikationen grundsätzlich aber auch im Kontext einer großen Menge an Daten effizient verwalten und auswerten zu können.

Im Fokus des nun folgenden zweiten Teils dieser Arbeit steht die Entwicklung eines umfassenden Zeitmodells und zugehörigen Beschreibungsformalismus, die ausdrucksstark und mächtig genug sind, um periodische Spezifikationen der Realwelt mit ihren Ausnahmen und Granularitäten adäquat abbilden und beschreiben zu können. Den Kern dieser Betrachtungen wird ein algebraischer Beschreibungsansatz bilden. Eine der vielen Anforderungen ist in diesem Zusammenhang die Ableitbarkeit von Wiederholungsmustern; sie ist vor allem für die Bereitstellung von Vergleichsoperationen essentiell.

Im dritten Teil der Arbeit entwickeln wir dann ein logisches Modell zur Darstellung von Wiederholungsmustern. Dieses gewährleistet neben der effizienten Vergleichbarkeit auch eine speichereffiziente interne Darstellung von periodischen Zusammenhängen. Darüber hinaus werden Betrachtungen vorgenommen, wie periodische Zusammenhänge auch im Kontext einer großen Menge an Daten effizient verwaltet und ausgewertet werden können. Hierzu zählen auch Betrachtungen zur Entwicklung geeigneter Zugriffs- und Indexstrukturen für entsprechend repräsentierte periodische Spezifikationen.

Teil II

Beschreibung von periodischen Spezifikationen

3 Ein Beschreibungsmodell für Periodizitätsschemas

In diesem Kapitel werden ein Zeitmodell und ein zugehöriger Beschreibungsformalismus vorgestellt, die eine umfassende und benutzerfreundliche Beschreibung periodischer Spezifikationen (Periodizitätsschemas) ermöglichen. Insbesondere dem Problem der mangelhaften Unterstützung von Ausnahmen innerhalb solcher Spezifikationen (z.B. Schaltjahre), das von existierenden Ansätzen nur unzureichend gelöst ist, kann mit diesem neuen Beschreibungsmodell begegnet werden. Ausgangspunkt der nachfolgenden Betrachtungen bildet eine detaillierte Vorstellung der bestehenden Anforderungen und Entwurfsziele in Ergänzung zu den bereits beschriebenen Anforderungen aus Kapitel 2. Neben der Berücksichtigung von Ausnahmen stehen insbesondere auch Anforderungen nach Benutzerfreundlichkeit und Anwendbarkeit eines Beschreibungsmodells im Fokus des Interesses. Basierend hierauf werden in Abschnitt 3.2 bestehende Ansätze bezüglich ihrer Erweiterbarkeit bzw. Anpassbarkeit an die aufgezeigten Anforderungen und Entwurfsziele untersucht und bewertet. In Abschnitt 3.3 wird ein neues Zeitmodell – das *zyklenorientierte Zeitmodell* – eingeführt und in seinen Grundstrukturen und Eigenschaften informell vorgestellt. Abschnitt 3.4 führt anschließend eine algebraische Notation mit entsprechenden Beschreibungsoperationen ein. Den Abschluss des Kapitels bildet zum einen die Beschreibung verschiedener Beispiele aus der Praxis mit dem zyklenorientierten Modell (Abschnitt 3.4.4) und zum anderen eine abschließende Diskussion des vorgestellten Beschreibungsmodells (Abschnitt 3.5).

3.1 Anforderungen und Entwurfsziele

Die in der Praxis relevanten Anforderungen an eine umfassende logische Beschreibung von Periodizitätsschemas sind vielfältig. In diesem Abschnitt stellen wir diese Anforderungen und damit verbundene Entwurfsziele an ein entsprechendes Zeitmodell (bzw. an einen Beschreibungsformalismus) im Detail vor.

3.1.1 Verständlichkeit

Ein Benutzer sollte periodische Spezifikationen bzw. Periodizitätsschemas auf semantisch hoher Ebene beschreiben können. So müssen entsprechende Beschreibungen sowohl für "Nicht-Experten" intuitiv nachvollziehbar sein als auch für die "Fachanwender" transparent und leicht anpassbar sein. Anpassungen werden z.B. häufig für medizinische Therapien erforderlich, da vorgegebene Periodizitätsschemas häufig nur als Vorlagen dienen, die nachträglich an patientenabhängige Gegebenheiten angepasst werden müssen.

Um diesen Anforderungen begegnen zu können, wird ein "intuitiv verständliches" Zeitmodell bzw. ein Beschreibungsformalismus auf semantisch hoher Ebene benötigt. So gilt es, Strukturen wie Zyklen und Zyklenpositionen unmittelbar durch das Zeitmodell beschreiben zu können. Ein manuelles "Nachbauen" dieser durch den Benutzer dagegen, sollte im Hinblick auf Benutzerfreundlichkeit und Fehlerrisiko vermieden werden.

3.1.2 Unterstützung von Ausnahmen

Die Unterstützung von Ausnahmen, d.h. Abweichungen von regulären Periodizitätsschemas, stellt eine der wesentlichsten Anforderungen dar. Erst die Beschreibbarkeit sowohl äquidistanter als auch komplexer nichtäquidistanter Ausnahmen ermöglicht es, praxisrelevante Periodizitätsschemas adäquat unterstützen zu können.

Ausnahmen stellen Abweichungen bestimmter Elemente (z.B. Zyklen, Intervalle) von einem vorgegebenen Grundschema dar. Die Auswahl der von einer Ausnahme bzw. Änderung betroffenen Elemente des Grundschemas erfolgt durch ein entsprechendes "Auswahlschema". Basierend auf dieser Auswahl können die Elemente dann gezielt manipuliert werden. Ein Beispiel dafür ist die Verlängerung von Zyklen oder das Einfügen zusätzlicher Elemente bzw. Werte.

In diesem Zusammenhang unterscheiden wir periodische Auswahlschemas (z.B. "*Jeden 4. Zyklus*", "*Jeder 4. Wert*") und nichtperiodische (z.B. "*nur der 4. Zyklus*"). Im ersten Fall stellt ein Auswahlschema selber wieder eine periodische Angabe dar, so dass hinsichtlich der Ausdrucksstärke solcher Schemas dieselben Anforderungen bestehen, wie für Periodizitätsschemas im Allgemeinen. Das heißt, neben Auswahlschemas mit einfachen äquidistantem Verhalten werden auch Auswahlschemas mit komplexem nichtäquidistanten Verhalten benötigt. Letztere Auswahlschemas können selber wiederum Ausnahmen beinhalten. Beispielsweise kann bei einer Chemotherapie eine Ausnahme auf Basis eines periodischen Feiertags definiert sein. Das Auswahlschema, d.h. das Periodizitätsschema für den Feiertag, kann wiederum nichtäquidistantes Verhalten aufweisen und Ausnahmen beinhalten.

Hinsichtlich der Spezifikation von Ausnahmen bzw. entsprechender Auswahlschemas lassen sich zwei Spezifikationslogiken unterscheiden: *implizite* und *explizite Spezifikationslogik*.

3.1.2.1 Implizite Spezifikationslogik bei Ausnahmen

Das allgemeine Vorgehen zur Spezifikation von Ausnahmen wird bei dieser Spezifikationslogik implizit in der Semantik einer einzelnen Beschreibungsoperation hinterlegt. Das Auswahlschema sowie die spezifische Änderung werden untrennbar in einer einzelnen Operation "verschmolzen". Beispielsweise beschreibt für ein Periodizitschemas G der Ausdruck $ALTER(G, 3, 1, +1)$ die Ausnahme *"Verlängere jeden 3. Zyklus des Periodizitschemas G (beginnend beim 1.) um den Wert +1"* [8, 65].

Charakteristisch für diese Spezifikationslogik ist, dass Auswahlschemas bzw. deren Eigenschaften ausschließlich über Parameter der algebraischen Ausdrücke bzw. Operationen (siehe Abschnitt 2.2.1) charakterisiert werden, wie etwa durch den zweiten und dritten Parameter im obigen Beispiel. Allerdings führt diese Form der Beschreibung von Auswahlschemas zu einigen Nachteilen:

- Das konstruktive Vorgehen bei der Beschreibung von Ausnahmen wird innerhalb der Semantik einer einzelnen Operation "versteckt". Mögliche Kombinationen von Änderungen (z.B. Einfügen von Elementen, Verlängern von Zyklen) und Arten von Auswahlschemas (z.B. äquidistante und nichtäquidistante Schemas) führen dann zu einer Vielzahl an Einzeloperationen (eine für jede Kombination) mit diffizilen und für Anwender kaum nachvollziehbaren Unterschieden in deren Semantik.
- Das Auswahlschema stellt keine eigenständige periodische Angabe dar und wird ausschließlich durch Parameter der Operationen charakterisiert. Bei komplexen Auswahlschemas führt dies jedoch zu einer hohen Anzahl an Parametern für die Charakterisierung von dessen Verhalten. Ebenso entsteht eine Vielzahl an Varianten für eine einzelne Operation, die sich in Abhängigkeit von der Charakteristik des Auswahlschemas in Anzahl und Semantik ihrer Parameter unterscheiden.
- (Periodische) Auswahlschemas und Grundschemas werden, obwohl sie jeweils periodische Angaben darstellen, auf unterschiedliche Art und Weise beschrieben. Allerdings ist es kaum möglich, alle, in der Praxis vorkommenden bzw. vorstellbaren Formen von Auswahlschemas (inklusive evtl. enthaltener Ausnahmen) in Form von Parametern vorab zu definieren. Eine identische Ausdrucksstärke zwischen Auswahl- und Grundschema ist mit dieser Spezifikationslogik folglich nicht erreichbar.

Eine implizite Spezifikationslogik mit seiner Vielzahl an Einzeloperationen und Varianten erweist sich für Nicht-Experten jedoch als wenig transparent und ist deshalb schwierig nachvollziehbar. Es bedarf einiges an Aufwand und ein tieferes Verständnis, um die genaue Semantik und die feinen Unterschiede zwischen den einzelnen Operationen verstehen zu können und somit die Semantik eines entsprechend beschriebenen Periodizitätsschemas interpretieren zu können. Eine solche Situation ist jedoch kritisch im Zusammenhang mit der Anforderung *Verständlichkeit.*

3.1.2.2 Explizite Spezifikationslogik bei Ausnahmen

Die explizite Spezifikationslogik ist im Wesentlichen dadurch gekennzeichnet, dass die Auswahlschemas eigenständige Strukturen bzw. periodische Angaben darstellen und analog zu Periodizitätsschemas beschrieben werden. Das heißt, es werden die gleichen Operationen wie zur Spezifikation allgemeiner Periodizitätschemas verwendet.

Im Beschreibungsformalismus führt die explizite Verfügbarkeit der Auswahlschemas zu zwei grundlegenden Operationsklassen: *Auswahloperationen* und *Änderungsoperationen.*

Auswahloperationen wählen auf Basis eines übergebenen expliziten Auswahlschemas bestimmte Elemente innerhalb eines Grundschemas aus bzw. "markieren" diese. Mit Änderungsoperationen können die markierten Elemente dann gezielt geändert werden. Um das Prinzip der expliziten Spezifikationslogik zu verdeutlichen, wird im Folgenden ein Pseudo-Beschreibungsformalismus beispielhaft dargestellt.Die Symbole $\oplus$ und $\odot$ entsprechen Grund- bzw. Änderungsoperationen (z.B. Verlängern von Zyklen, Löschen von Elementen). Die Operation *Select* stellt eine Auswahloperation dar, etwa zur periodischen Auswahl bestimmter Zyklen. Im Folgenden sind die vier wesentlichen Grundschritte der Beschreibung einer Auswahl bei expliziter Spezifikationslogik dargestellt:

1. Grundschema $G = \oplus(\odot(\oplus(...)))$
2. Auswahlschema $A = \oplus(\odot(\oplus(\oplus...)))$
3. Auswahloperation $G' = Select(G, A)$
4. Änderungsoperation $G'' = \oplus(G', A, ...)$

In den ersten beiden Schritten werden zunächst das Grundschema und das Auswahlschema mittels entsprechender Grund- bzw. Änderungsoperationen beschrieben. Bei der Beschreibung des Auswahlschema werden die gleichen Operationen verwendet, wie für das Grundschema auch. Im 3. Schritt werden die Auswahloperation *Select* mit dem zuvor definierten Auswahlschema auf das Grundschema angewendet und die entsprechenden Elemente ausgewählt bzw. markiert. Im letzten Schritt werden, basierend auf dem Schema G', die zuvor markierten Elemente entsprechend geändert (Die zu ändernden Elemente werden durch den Parameter A festgelegt.). Im Vergleich zur impliziten Spezifikationslogik weist die explizite Spezifikationslogik mehrere Vorteile auf:

- Das konstruktive Vorgehen bei der Beschreibung von Ausnahmen liegt in Form einer expliziten "Konstruktionsvorschrift" vor (siehe Schritte 1. - 4. im Beispiel). Zum einen führt dies zu einigen wenigen generischen Beschreibungsoperationen, die klar in ihrer Semantik voneinander abgrenzbar sind. Zum anderen liegt die Konstruktionsvorschrift des Schemas vollständig in der Hand des Benutzers und wird nicht in der Semantik einzelner Operationen (wie im impliziten Fall) "versteckt".
- Periodizitätsschemas und Auswahlschemas werden auf die gleiche Art und Weise beschrieben. Infolgedessen ist die Ausdrucksstärke von Periodizitäts- und Auswahlschemas identisch, d.h. Auswahlschemas können beliebige Komplexität in ihrem Verhalten aufweisen.
- Die explizite Konstruktionsvorschrift sowie die wenigen generischen Operationen vereinfachen den Umgang sowie die nachträgliche Anpassbarkeit entsprechend beschriebener Periodizitätsschemas erheblich. Eine solche Spezifikationslogik hat somit das Potential, entsprechende Beschreibungen transparent und intuitiv nachvollziehbar zu gestalten (siehe Anforderung Verständlichkeit).

Zusammenfassend betrachtet bildet die Verwendung einer expliziten Spezifikationslogik eine Kernanforderung im Zusammenhang mit der Unterstützung von Ausnahmen. Nur

so können den Anforderungen bzgl. der Ausdrucksstärke von Ausnahmen und Verständlichkeit der Beschreibung adäquat begegnet werden.

3.1.3 Granularitätsfähigkeit

Die Verwendung unterschiedlicher zeitlicher Einheiten (Granularitäten), wie Tage und Wochen, ist weit verbreitet in der Praxis. So sind für eine Chemotherapie Angaben in Tagesabständen ausreichend, während andere Therapien (z.B. bestimmter Medikamentengaben) stündliche Angaben benötigen. Im Finanzsektor wiederum sind periodische Aussagen mit den Granularitäten Monat oder Jahr typisch. In Abhängigkeit von Anwendung und Einsatzgebiet ergibt sich zusätzlich die Notwendigkeit, mit verschiedenen Kalendern umgehen zu können (z.B. gregorianischer Kalender, persischer Kalender, islamischer Kalender, Mondkalender).

Welche Granularitäten bzw. Kalender im Einzelnen verwendet werden, ist – wie erwähnt – anwendungsabhängig. Es wird daher ein Zeitmodell benötigt, das es ermöglicht, unterschiedliche Granularitäten bzw. unterschiedliche Kalender zu unterstützen. In diesem Zusammenhang sollten periodische Spezifikationen, die mehrere (unterschiedliche) Granularitätsebenen verwenden, abbildbar sein (z.B. "*Alle 2 Stunden an jeden 3.Tag*").

3.1.4 Strukturierbarkeit

Komplexe periodische Spezifikationen können aus einer Vielzahl verschiedener Zyklen und Einzelaktivitäten (z.B. Gabe verschiedener Medikamente in verschiedenen Dosierung oder Rehabilitationsmaßnahmen) bestehen. So können innerhalb der Therapie bestimmte "Teilzyklen" unterschieden werden, d.h. eigenständige Abschnitte bzw. Phasen, in denen bestimmte Maßnahmen ergriffen werden. Ein einfaches Beispiel bilden "Pausenzyklen" die zwischen den einzelnen Zyklen der Medikamentengabe stattfinden. Die verschiedenen Teilzyklen und ihre Reihenfolgen stellen in ihrer Gesamtheit das vollständige Periodizitätsschema dar, auf die sich dann auch Wiederholungsangaben bei Therapien beziehen.

Ähnlich verhält es sich bezüglich der Vielzahl an Einzelaktivitäten. Die verschiedenen Einzelaktivitäten innerhalb der Zyklen und Teilzyklen stellen erst in ihrer Gesamtheit den vollständigen Chemotherapieablauf dar. Im Allgemeinen empfiehlt es sich nicht, die unterschiedlichen Aktivitäten jeweils als einzelne, eigenständiges Periodizitätsschema oder periodische Spezifikationen zu betrachten. So bestehen zwischen den einzelnen Aktivitäten häufig Abhängigkeiten. So kann die Änderung eines Zyklus etwa nicht nur Einfluss auf eine einzelne Gabe eines Medikaments haben, sondern auch auf andere Medikamentengaben in diesem (geänderten) Zyklus. Änderungen an einem einzelnen Schema müssten dann in den anderen betroffenen Spezifikationen explizit "nachgezogen" werden, was jedoch von Pflegeaufwand her kaum bewerkstelligbar ist.

Entsprechende Strukturierungen bzw. die Festlegung der Gesamtheit an Aktivitäten bei komplexen periodischen Spezifikation können grundsätzlich auch innerhalb einer Anwendung festgelegt werden. Verständlicher und bzgl. der Forderung nach *Korrektheit* (siehe unten) auch wünschenswerter ist jedoch ein Zeitmodell, das Strukturierungsmöglichkeiten für Periodizitätsschemas aufweist, die a priori einen gesamtheitlichen Umgang mit Zyklen, Teilzyklen sowie einer Vielzahl an Einzelaktivitäten ermöglichen.

3.1.5 Korrektheit

Das Zeitmodell muss die "Korrektheit" einer periodischen Spezifikation analysieren können. So müssen semantisch falsche, inkonsistente oder widersprüchliche Angaben erkannt bzw. vermieden werden. Ein einfaches Beispiel bildet etwa die Aussage "*Aktivität B findet immer dann statt, wenn Aktivität A des Periodizitätsschemas auf einem Montag fällt*". Allerdings ist es leicht möglich, dass Aktivität A niemals auf einen Montag fällt. In solchen Fällen gilt es, den Anwender darauf hinzuweisen, dass diese Ausnahme niemals eintreffen wird.

Während eine solche Inkonsistenz bei einfachen Beispielen noch leicht erkennbar ist, werden entsprechende "Seiteneffekte" im Fall umfangreicher Spezifikationen bzw. Ausnahmebeschreibungen sehr komplex und sind kaum mehr überschaubar. Ein entsprechendes Zeitmodell muss es daher gewährleisten, dass die Korrektheit von Schemas auch nachträglich überprüft werden kann oder aber per Konstruktion sichergestellt wird.

3.1.6 Auswertbarkeit

Die operationale Unterstützung periodischer Spezifikationen ist wesentlich für die Praxis. Wie in Abschnitt 2.1.3 dargestellt, stellt hier die Realisierung von Vergleichsoperationen die größte Herausforderung dar. In Abschnitt 2.4 wurde bereits gezeigt, dass nur eine auf Wiederholungsmustern basierende Realisierung der Anforderung nach operationaler Unterstützung gerecht werden kann. Daher fordern wir, dass ein entsprechendes Zeitmodell bzw. ein entsprechender Beschreibungsformalismus so ausgewertet werden kann, dass Wiederholungsmuster ableitbar sind.

Jede der vorgestellten Anforderungen und Entwurfsziele ist wesentlich im Zusammenhang mit der praktischen Anwendbarkeit der Beschreibung von Periodizitätsschemas. Allerdings nimmt die Anforderung nach expliziter Spezifikationslogik bei Ausnahmen eine besondere Stellung ein. Erst eine explizite Spezifikationslogik ermöglicht es, komplexe nichtäquidistante Ausnahmen beliebiger Komplexität zu beschreiben. In Abschnitt 2.2 wurde bereits gezeigt, dass in der Literatur vorliegende Ansätze ausschließlich eine implizite Spezifikationslogik verwenden und somit diese Kernanforderung nicht erfüllen. Im folgenden Abschnitt wird zunächst untersucht, ob die in der Literatur vorliegenden Ansätze um eine explizite Spezifikationslogik erweitert werden können.

3.2 Betrachtung vorliegender Zeitmodelle

Aufgrund der Anforderungen nach umfassender logischer Beschreibung von praxisrelevanten Periodizitätsschemas sind die Ansprüche an ein entsprechendes Zeitmodell (bzw. einen Beschreibungsformalismus) hoch. Wie im vorherigen Abschnitt gezeigt, stellt die Forderung nach expliziter Spezifikationslogik eine der Kernanforderungen dar. In der Literatur beschriebene Ansätze weisen ausschließlich eine implizite Spezifikationslogik auf (siehe Abschnitt 2.2.1). Hieraus ergibt sich die Frage, ob bestehende Ansätze so angepasst werden können, dass sie eine explizite Spezifikationslogik ermöglichen.

Um dies zu beantworten, müssen die jeweils zugrundeliegenden Zeitmodelle näher betrachtet werden. Generell beschreibt ein Zeitmodell, was im jeweiligen Ansatz unter dem Begriff "Zeit" verstanden wird, d.h welche semantischen Möglichkeiten zur Verfügung stehen, um den Zeitbereich (d.h. die unendliche Menge an zeitlichen Werten) zu strukturieren. Die formalen Modelle legen die Semantik der zeitlichen Strukturen sowie der algebraischen Beschreibungsoperationen exakt fest. Somit spiegeln sich die Eigenschaften des Zeitmodelle direkt auch in der Spezifikationslogik und dem auf ihr aufbauenden Beschreibungsformalismus (vgl. Abschnitt 2.2) wider. Die Frage nach Anpassung bestehender Formalismen bzgl. der Unterstützung einer expliziten Spezifikationslogik kann somit nur in Verbindung mit dem jeweiligen formalen Zeitmodell beantwortet werden.

Im Folgenden werden die in der Literatur vorliegenden relevanten Zeitmodelle in ihren wesentlichen Eigenschaften vorgestellt und anschließend bzgl. ihrer Unterstützung einer expliziten Spezifikationslogik bewertet. Trotz der Vielzahl vorliegender Beschreibungsformalismen können diese im Wesentlichen auf drei formale Zeitmodelle zurückgeführt werden: *Generalisierte Intervalle* (siehe Abschnitt 3.2.1), *Time-Granularity Modell* (siehe Abschnitt 3.2.2) und *Partitionen-Modell* (siehe Abschnitt 3.2.3).

3.2.1 Generalisierte Intervalle

In diesem einfachen Zeitmodell werden Periodizitätsschemas als (unendliche) Menge von zeitlichen Intervallen abgebildet (z.B. [53, 64, 87, 39]). Die Intervallmenge (z.B. Collections in [53]) stellt die konkreten Zeitwerte dar, d.h. sie legt fest, wann ein Ereignis eingetreten ist oder sich wiederholt hat. Eine Menge von Intervallen kann zusätzlich strukturiert werden, d.h. eine Teilmenge von semantisch zusammengehörigen Intervallen läßt sich erneut als Menge zusammenfassen. Dieses Zusammenfassen von Intervallmengen kann beliebig (tief) geschachtelt werden. Es wird entsprechend als Ordnung der Menge bezeichnet. Eine Order-1 Collection [53] etwa entspricht einer geordneten Menge von Intervallen und eine Order-2 Collection einer Menge von geordneten Intervallmengen (da Schachtelungstiefe = zwei).

Eine Chemotherapie mit Medikament A vom 1. bis zum 14. Tag und Medikament B am 1. und 8. Tag in einem Zyklus von 28 Tagen entspricht den folgenden strukturierten Intervallmengen (Order-2 Collections). Als Instanziierungspunkt wird die Eins

angenommen. Zum besseren Verständnis ist die Ordnung der Menge als Index an den schließenden Klammern aufgeführt. Die "inneren" Mengen, etwa $\{(1-1),(8-8)\}_1$ in Med_B, entsprechen den einzelnen Gaben von Medikamenten innerhalb eines Zyklus.

- $Med_A = \{...\{(1-1),(2-2),...,(14-14)\}_1,\{(29-29),...,(42-42)\}_1,...\}_2$
- $Med_B = \{...\{(1-1),(8-8)\}_1,\{(29-29),(37-37)\}_1,\{(57-57),(65-65)\}_1,...\}_2$

Mit einer weiteren Strukturierung zu einer Order3-Collection können die ersten sechs Therapiezyklen zu einer vollständigen Therapie zusammengefasst werden.

- $Med_A = \{...,\{\{(1-1),...,(14-14)\}_1,...,\{(141-141),..,(154-154)\}_1\}_2,\ ...\}_3$
- $Med_B = \{...\{\{(1-1),(8-8)\}_1,...,\{(141-141),(149-149)\}_1\}_2,...\}_3$

3.2.2 Time-Granularity Modell

Das *Time-Granularity* Zeitmodell entstand vor dem Hintergrund der Repräsentation und Auswertung zeitlicher Werte mit verschiedenen Granularitäten und Kalendern. Auf diesem Zeitmodell basieren eine Vielzahl sowohl algebraischer Beschreibungsansätze [8, 12, 36, 41, 65] als auch Constraint- und Automatenbasierter Ansätze [14, 19, 34, 32, 94].

Die Kernelemente des Zeitmodells bilden die Begriffe *Granularität*, *Granules* und (diskrete) *Indexmenge*. Eine *Granularität* wird als eine Partitionierung eines (diskreten) Wertebereichs in nicht zwingend unmittelbar aneinandergrenzende Partitionen verstanden. Diese einzelnen Partitionen werden als *Granules* bezeichnet. Eine Partition bzw. Granule stellt eine semantisch zusammengehörige Menge von Werten dar und wird (semantisch) als unteilbar erachtet. Die Indexmenge einer Granularität bzw. eine entsprechende bijektive Indexfunktion dient dem "Abzählen" der Granules. Der *Indexwert* identifiziert ein Granule eindeutig, d.h. es kann eine entsprechende Partition bzw. ein Intervall im zeitlichen Wertebereich zugeordnet werden. Beispielsweise stellen die einzelnen *Wochentage* (z.B. Montag, Dienstag,...) die Granules der Granularität *Woche* dar. Das Granule mit Index 1 entspricht hier dem ersten Montag bzw. das mit Index 8 dem zweiten Montag der Granularität *Woche*. In Bezug auf eine Chemotherapie etwa ein Granule der einzelnen Gabe eines Medikaments. Über die entsprechenden Indexwerte können dann die Gaben von Medikament A in einer Chemotherapie abgezählt und entsprechende Zeitwerte zugeordnet werden, d.h. die Granules mit Indexwerten 1-14 entsprechen den Gaben des Medikaments innerhalb des ersten Zyklus bzw. die Granules mit Indexwerten 15-29 den Medikamentengaben des zweiten Zyklus (vgl. Abbildung 3.1).

Neben äquidistanten und nichtäquidistanten Granularitäten (z.B. Granularität *Monate*) unterstützt das Modell auch Granularitäten, die "Löcher" enthalten. Ein Loch beschreibt hierbei eine Situation, bei der zwei aufeinanderfolgende Granules einer Granularität im zugrundeliegenden Wertebereich nicht unmittelbar aneinandergrenzende Intervalle bilden. Beispielsweise besteht bei der Granularität "Geschäftswoche" ein Loch von 2 Tagen,

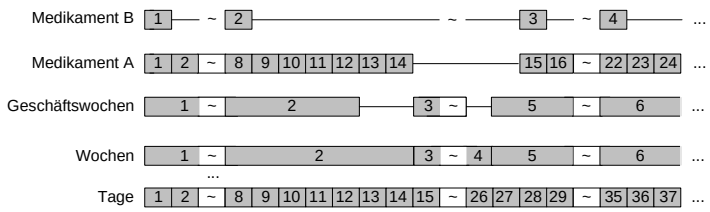


Abbildung 3.1: Darstellung von Granularitäten und periodischen Spezifikationen im Time-Granularity Modell

da die Tage Samstag und Sonntag im Allgemeinen nicht einer Geschäftswoche zugerechnet werden. Im Hinblick auf diese Semantik des Modells entsprechen die Medikamentengaben einer Chemotherapie einer Granularität mit Löchern. Die Löcher werden durch die Abstände zwischen den einzelnen Medikamentengaben "verursacht". Abbildung 3.1 zeigt verschiedene Granularitäten, inklusive des vorherigen Chemotherapiebeispiels und dessen Darstellung im Time-Granularity Modell.

3.2.3 Partitionen-Modell

Das *Partitionen-Modell* entstand vor dem Hintergrund, der umfassenden Beschreibung und Unterstützung zeitlicher Phänomene [78, 66]. Ein Ziel war es, komplexe Kalenderphänomene wie Schaltsekunden, Berechnung der Osterfeiertage oder Beschreibung des Mondkalenders abbilden zu können.

Die Grundlage des Modells bilden die Begriffe *Partitionierung*, *Koordinate* und *Label*. Eine *Partitionierung* ist die Unterteilung eines (reellen) Wertebereichs in Partitionen bestimmter Länge. Die einzelnen Partitionen sind disjunkt und folgen unmittelbar aufeinander. Somit gehört jeder Wert des Wertebereichs genau einer Partition der Partitionierung an. Jede Partition besitzt einen ganzzahligen Index, der diese eindeutig identifiziert. Dieser Indexwert wird als (Partitions-) *Koordinate* bezeichnet. Es existiert zusätzlich eine (bijektive) Abbildungsfunktion, die für jeden Wert des Wertebereichs die Koordinate der Partition bestimmt, die den Wert enthält. *Labels* wiederum ermöglichen es jeder Partition (auf Basis ihrer Koordinate), eine zusätzliche Bezeichnung bzw. zusätzliche semantische Informationen zuzuweisen. Beispielsweise weisen einige Partitionen, wie die Tage innerhalb einer Woche, einen vordefinierten Namen (z.B. Montag, Dienstag) auf. Mittels der Labels kann den entsprechenden Partitionen bzw. Tagen die korrespondierende Bezeichnung für die Wochentage zugewiesen werden.

Bei Anwendung auf das im vorherigen Abschnitt 3.2.2 vorgestellte Time-Granularity Modell können Partitionen und Partitionierung als Granules und Granularitäten verstanden werden. Im Unterschied zum vorherigen Modell existieren jedoch keine "Löcher" in Granularitäten bzw. Granules, d.h. es gibt keine Abstände zwischen aufeinanderfolgenden Partitionen. Die Abbildung dieser praktisch relevanten Art von Granularitäten

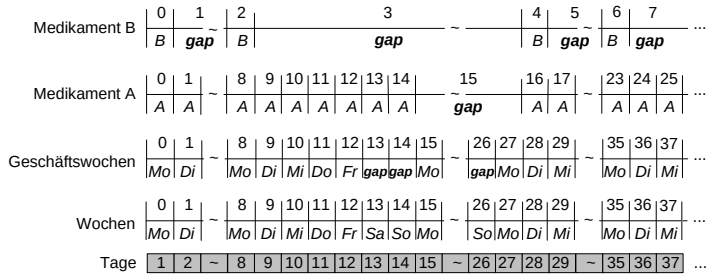


Abbildung 3.2: Darstellung von Granularitäten und periodischen Spezifikationen im Partitionen Modell

erfolgt in dem Modell grundsätzlich anders. Anstatt Löcher in der Partitionierung zuzulassen, werden die Partitionen, die semantisch "Löcher" entsprechen, lediglich als Löcher "markiert". Hierzu existiert ein vordefiniertes Label `gap`. Es zeigt an, dass Partitionen mit diesem Label bzw. die entsprechenden Werte (logisch) nicht in einer Granularität oder periodischen Spezifikation enthalten sein sollen. Diese Form der Modellierung von Löchern ist algorithmisch weitaus einfacher zu handhaben im Vergleich zu einer Modellierung mit Abständen zwischen Partitionen [66]. Beispielsweise kann die Gabe von Medikament A einer Chemotherapie als Partitionierung von 14 einzelnen Partitionen zu je einem Tag und einer einzelnen Partition mit 14 Tagen dargestellt werden (siehe Abbildung 3.2). Die ersten 14 Partitionen entsprechen Medikamentengaben und erhalten als Label z.B. den Medikamentennamen. Die 2. Partition mit Länge 14 erhält das Label `gap`, da diese dem Loch, d.h. dem Abstand zwischen der letzten Medikametengabe eines Zyklus und der ersten Gabe des nachfolgenden Zyklus, entspricht. Auf die gleiche Weise kann auch die Granularität *Geschäftswoche* abgebildet werden. Während den ersten fünf Partitionen (d.h. Tagen) entsprechende Labels (Wochentage) zugeordnet werden, erhalten die letzten beiden Partitionen (d.h. Samstag und Sonntag) jeweils das Label `gap`. Abbildung 3.2 zeigt die Darstellung der Beispiele Chemotherapie sowie die Granularitäten *Woche* und *Geschäftswoche* im Partitionen-Modell.

3.2.4 Bewertung

Die Erweiterung um eine explizite Spezifikationslogik bedeutet im Allgemeinen, dass in den entsprechenden Formalismen zwei neue Operatorenklassen unterstützt werden müssen. Zum einen bedarf es Operatoren, die auf Basis eines (periodischen) Auswahlschemas bestimmte Elemente bzw. Werte "markieren" können, etwa Angaben der Form "*Jeden 3. Zyklus*" oder "*Jeden 10. Wert*". Zum anderen werden Änderungsoperatoren benötigt, die basierend auf einer solchen Auswahl verschiedene Änderungen, wie Löschungen, Verschiebungen oder Einfügungen der markierten Elemente, vornehmen können.

Voraussetzung für eine Auswahloperation ist, dass das Zeitmodell die ausgewählten Elemente so "zugreifbar" macht, dass sie durch Änderungsoperationen manipuliert werden können. Im Zusammenhang mit diesem Aspekt lassen sich bei den vorgestellten Zeitmodellen folgende Beobachtungen machen:

- **Generalisierte Intervalle** entsprechen lediglich einer infiniten Sammlung von Intervallen. Lediglich über die Ordnung der Mengen können einfache semantische Zusammenhänge abgebildet werden, etwa die Zugehörigkeit eines bestimmten Intervalls bzw. Medikamentengabe zu einem bestimmten Zyklus. Möglichkeiten zum Markieren von Intervallen oder Intervallmengen bzw. optional diese Mengen einer bestimmten Auswahl zuzuordnen, sind in diesem Zeitmodell nicht vorhanden. Deshalb kann eine explizite Spezifikationslogik nicht unterstützt werden.
- Das **Time-Granularity Modell** ermöglicht es, auf Basis einer Indexmenge bzw. von Indexfunktionen, die Intervalle (Granules) zu identifizieren. Der Indexwert ermöglicht auch den Umgang mit den Intervallen von "außen", d.h. ein Nutzer kann über den Index auf ein Intervall bzw. den Wert einer Granularität (oder periodischen Spezifikation) zugreifen. Ein Problem ist jedoch, dass der Index die einzige Möglichkeit darstellt, ein Intervall zu bezeichnen. Es bestehen keine Möglichkeiten, ein Intervall zu markieren oder einer bestimmten Auswahl zuzuordnen. Eine explizite Spezifikationslogik kann mit diesem Zeitmodell somit nicht realisiert werden.
- Das **Partitionen-Modell** stellt Möglichkeiten zur Verfügung, Intervalle (Partitionen) sowohl über Indexwerte zu identifizieren als auch zusätzlich semantische Informationen (Labels) zuzuordnen. Die Verwendung von Labels führt in dem Modell zu einer *Trennung von Struktur und Semantik*. Während die Partitionierung die Struktur vorgibt, wird durch die Labels die Semantik der Partitionen bzw. der entsprechenden Wertemengen (informell) festgelegt. Somit können gleiche Partitionierung aufgrund unterschiedlicher Labelzuweisung unterschiedliche Semantik aufweisen (wie in Abbildung 3.2 für *Woche* und *Geschäftswoche* gezeigt). Es ist daher möglich, Intervalle nicht nur über Indexwerte zugreifbar zu machen, sondern auch über die Labels und deren Semantik. Die Labels können verwendet werden, um die Elemente einer (periodischen) Auswahl zu kennzeichnen. Das Modell erfüllt somit die wesentliche Anforderungen für eine explizite Spezifikationslogik.

Das Partitionen-Modell ist somit das einzige existierende Zeitmodell, das a priori eine explizite Spezifikationslogik zu unterstützen vermag. Mit entsprechenden Erweiterungen ist die Einführung von Auswahl- und Änderungsoperationen vorstellbar. Allerdings weist das Modell folgende Probleme im Hinblick auf die Beschreibung von Periodizitätsschemas auf semantisch hoher Ebene auf:

- Das Modell verwendet mit Partitionen semantisch sehr einfache Strukturen. Semantisch höherwertige Strukturen, wie Zyklen oder Zyklenpositionen, werden dagegen nicht explizit unterstützt. Solche Strukturen können zwar durch Verwendung von Labels implizit "nachgebildet" werden, allerdings erfordert dies erneut ein Um-

denken auf Seiten des Anwenders, um auf Basis der "Nachbildung" Angaben wie bei Chemotherapien zu beschreiben. Dies widerspricht jedoch der Anforderung nach *Verständlichkeit* (vgl. Abschnitt 3.1.1).

- Im Partitionen-Modell müssen die verschiedenen Medikamentengaben innerhalb eines Zyklus einer Chemotherapie jeweils als eigenständige periodische Spezifikation dargestellt werden. D.h. jede Medikamentengabe eines Zyklus verwendet jeweils eine eigene, zugrundeliegende Partitionierung. Dies widerspricht jedoch der Anforderung nach *Strukturierbarkeit* (vgl. Abschnitt 3.1.4).
- In dem Modell ist der vorliegende Beschreibungsformalismus sehr stark von einer mathematischen Sichtweise geprägt. So besteht entweder die Möglichkeit periodische Spezifikation in Form einer "Low-Level" Programmiersprache oder über eine (endliche) Aufzählung der Reihenfolge aller Partitionen mit ihren Längen zu beschreiben. Solche Beschreibungsformalismen sind für den normalen Anwender kaum zumutbar und widersprechen ebenfalls der Anforderung nach *Verständlichkeit* (vgl. Abschnitt 3.1.1).

Zusammenfassend betrachtet bildet das Partitionen-Modell einen "Low-Level"-Ansatz, wenn auch einem sehr ausdrucksstarkem. Für die Beschreibung periodischer Spezifikationen auf semantisch hoher Ebene erweist sich dieses Modell als weniger geeignet. Hierfür werden neue Modelle mit semantisch höherwertigen Strukturen und Beschreibungsmöglichkeiten benötigt.

Der vorliegende Abschnitt hat gezeigt, dass die Schwächen und Probleme vorliegender Beschreibungsansätze für periodische Spezifikationen zu großen Teilen in den zugrundeliegenden Zeitmodellen begründet sind. Um der Anforderung nach umfassender Beschreibung von Periodizitätsschemas begegnen zu können, wird ein neues Zeitmodell benötigt. Im folgenden Abschnitt werden mit dem *zyklenorientierten Modell* und seiner *algebraischen Notationsform* ein solches Zeitmodell und ein entsprechender Beschreibungsformalismus vorgestellt.

3.3 Vorstellung des zyklenorientierten Zeitmodells

In diesem Abschnitt wird das in dieser Arbeit entwickelte zyklenorientierte Zeitmodell zur Repräsentation von Periodizitätsschemas vorgestellt. Beim Entwurf dieses Zeitmodells werden die in Abschnitt 3.1 erörterten Anforderungen und Entwurfsziele berücksichtigt.

Die Grundstrukturen des zyklenorientierten Zeitmodells und dessen Eigenschaften werden im folgenden Abschnitt zunächst informell vorgestellt. Der Fokus der Betrachtungen liegt auf den Anforderungen nach Verständlichkeit und Strukturierbarkeit sowie der Unterstützung von Ausnahmen. Die Ziele Granularitätsfähigkeit, Auswertbarkeit und Korrektheit sowie eine formale Charakterisierung des Zeitmodells erfolgen in Kapitel 4.

3.3.1 Zyklus und Zykluspositionen

Die Basis des Zeitmodells bilden die Grundelemente *Zyklus* und *Zyklusposition.* Ein Zyklus bildet einen "semantischen Rahmen" bestimmter Länge, der bei entsprechender Interpretation (siehe später) einem Intervall in einem zugrundeliegenden Wertebereich entspricht. Zykluspositionen beschreiben die Menge an Werten, die durch einen einzelnen Zyklus im Wertebereich abgedeckt werden. Aktivitäten (z.B. Medikamentengaben) innerhalb eines Zyklus können spezifischen Zykluspositionen zugeordnet werden. Zykluspositionen werden relativ zum Startpunkt eines Zyklus indiziert. Bei entsprechender Interpretation entspricht jede Zyklusposition eines Zyklus einem konkreten zeitlichen Wert, an dem – bei entsprechender Zuordnung – eine konkrete Aktivität ausgeführt wird. Abbildung 3.3 zeigt einen Zyklus mit Länge n und die entsprechende Menge von Zykluspositionen $\{1, ..., n\}$.

Das Konzept von Zyklus und Zyklusposition findet man häufig bei periodischen Angaben der realen Welt wieder, etwa bei Medikationsanweisungen in der Medezin (z.B. "*Jede Woche am Montag, Mittwoch und Samstag*"). Der Zyklus entspricht hier einer einzelnen Woche und weist die Länge 7 auf. Die Wochentage "*Montag (1), Mittwoch (3) und Samstag* (6)" bilden die Zykluspositionen, an denen das Medikament verabreicht wird. Ein anderes Beispiel für a priori zyklenorientierte Angaben stellen Fahrpläne dar. In der Angabe "*Jede Stunde zur 30.Minute (immer zur halben Stunde)*" etwa bildet die Stunde (60 Minuten) den Zyklus und die 30.Minute die Zyklusposition.

Oftmals bedarf es einer Übertragung auf das Konzept Zyklus und Zykluspositionen, z.B. bei abstandsorientierten Angaben (z.B. *"Jeden 3.Tag erfolgt eine Medikamentengabe*") in denen lediglich der Abstand zwischen einzelnen Aktivitäten vorgegeben ist. Zur Übertragung auf die zyklenorientierte Sichtweise wird der angegebene Abstand als Zyklus verstanden (z.B. als Zyklus der Länge drei). In Abhängigkeit von der Semantik, d.h. "*jeden 3.Tag*" inklusive oder exklusive des Instanziierungspunktes ergibt sich die Zuordnung zu den Zykluspositionen eins oder drei. Diese Form der Überführung in die zyklenorientierte Sichtweise kann für beliebige abstandsorientierte Periodizitätsangaben vorgenommen werden. Abbildung 3.4 zeigt das Konzept Zyklus und Zyklusposition für das obige Beispiel der Wochentage sowie für die abstandsorientierte Angabe ("*Jeden 3.Tag beginnend mit dem Letzten*").

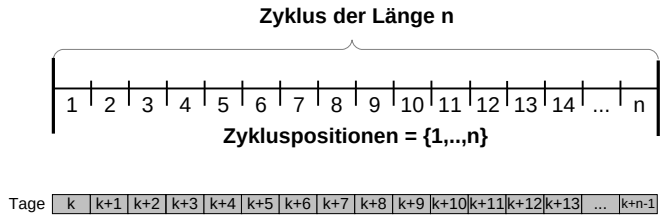


Abbildung 3.3: Grundstrukturen des zyklenorientierten Modells

(a) Zyklenorientierte Angabe "*Wochentage*" (b) Abstandsorientierte Angabe "*Jeden 3.Tag*"

Abbildung 3.4: Beispiels von Periodizitätsangaben und ihre Abbildung auf die Grundstrukturen des zyklenorientierten Modell

3.3.2 Labels

Jedem Zyklus und jeder Zyklusposition können *Labels* zur anwendungsabhängigen Bezeichnung zugeordnet werden. Die Menge an Labels entspricht einem kontrollierten Vokabular, das durch den Anwender vorgegeben wird. So könnten Labels für die Gabe einzelner Medikamente oder für Abfahrtszeiten von Zügen stehen.

Auf Basis von Labels erfolgt im Zeitmodell auch der Zugriff bzw. die Adressierung von Strukturen wie Zyklen und Zyklusspositionen, etwa um diese zu manipulieren oder zu ändern. Im Beispiel in Abbildung 3.5 kann etwa mit der Labelangabe `A` die Menge aller Medikamentengaben eines Zyklus bzw. die entsprechende Menge an Zyklusspositionen direkt adressiert werden. Neben der Bezeichnung und Adressierung dienen Labels auch der Verknüpfung bestimmter Aktivitäten (z.B. Medikamentengaben) mit einem konkreten Anwendungsdatenmodell. Das heißt, dass z.B. konkrete Medikamentengaben auf Basis von Labels mit den konkreten Medikamentendosierungen innerhalb der Anwendnung verknüpft werden können.

In Abbildung 3.5 ist ein Zyklus mit Labels dargestellt. Wie erwähnt, ist die Auswahl an Zyklusspositionen sowie die Zuordnungen der Labels anwendungsabhängig und hier nur exemplarisch zu verstehen.

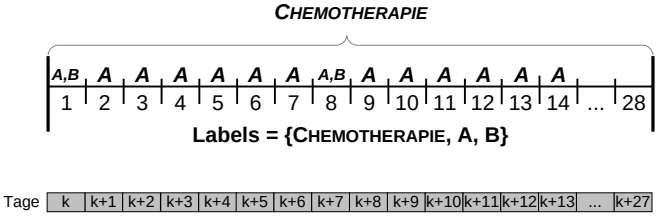


Abbildung 3.5: Verwendung von Labels zur Beschreibung eines Zyklus `Chemotherapie` der Länge 28 Tage sowie von Medikamentengaben `A` vom 1. bis zum 14. Tag sowie Gaben von Medikament `B` am 1. und 8. Tag

3.3.3 Strukturierte ("geschachtelte") Zyklen

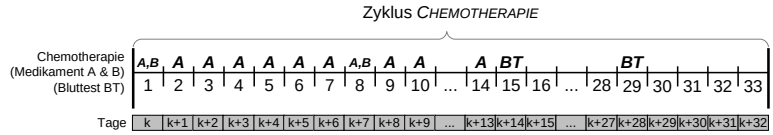
Einen Zyklus lediglich als einzelnes (zeitliches) Intervall bestimmter Länge zu betrachten reicht in der Praxis häufig nicht aus. Stattdessen werden im Allgemeinen zusätzliche Möglichkeiten zur Strukturierung von Zyklen benötigt. In unserem zyklenorientierten Zeitmodell stehen hierfür sogenannte *strukturierte Zyklen* zur Verfügung. Diese ermöglichen unterschiedliche Formen der Strukturierung, die nachfolgend vorgestellt werden.

3.3.3.1 Horizontale Strukturierung

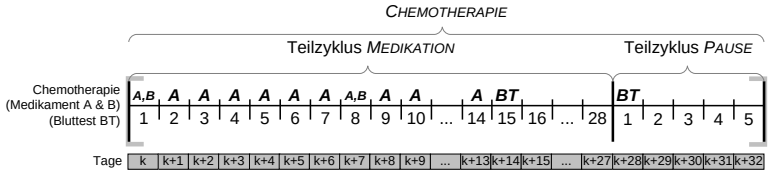
Die horizontale Strukturierung entspricht der Unterteilung eines Zyklus in mehrere *Teilzyklen*. Anstelle eines einzelnen "langen" Zyklus wird dieser durch eine (endliche) Menge kürzerer Teilzyklen modelliert. Diese Menge an Teilzyklen sowie ihre Reihenfolgen werden durch den strukturierten Zyklus repräsentiert. Formal entspricht jeder Teilzyklus erneut einem Zyklus.

Ein Beispiel für die Anwendung einer horizontalen Strukturierung ist eine Chemotherapie der Länge 28 Tage. Zwischen den einzelnen Therapiezyklen findet eine Pause von 5 Tagen statt. Bei Verwendung eines "langen" Zyklus der Länge 33 Tage (28 Tage + 5 Tage Pause) müsste eine Angabe wie "*jeden 1.Tag in der Pause einer Chemotherapie wird ein Bluttest durchgeführt*" manuell (durch Anwender) auf die korrespondierende Zyklusposition 29 umgerechnet werden. Bei komplexeren Angaben ist eine manuelle Umrechnung jedoch umständlich und fehleranfällig. Stattdessen empfiehlt es sich, eine horizontale Strukturierung des langen Zyklus in zwei Teilzyklen der Länge 28 Tage (`Medikation`) und 5 Tage (`Pause`) vorzunehmen. Im Falle der obigen Angabe kann so der Teilzyklus `Pause` und dessen 1. Zyklusposition (für die Zuordnung des Bluttests) unmittelbar adressiert werden. Eine manuelle Umrechnung des Anwenders entfällt. Die horizontale Strukturierung erleichtert den Umgang mit solchen Angaben. Das Beispiel ist in Abbildung 3.6 mit und ohne Strukturierung dargestellt. Der strukturierte Zyklus ist durch die graue eckige Klammer im Bild gekennzeichnet, d.h der strukturierte Zyklus besteht aus den beiden Teilzyklen `Medikation` und `Pause`.

Ein weiteres Beispiel für die Verwendung einer horizontalen Strukturierung bilden Öffnungszeiten wie "*Jeden Tag von 9:00 Uhr bis 12:00 Uhr und von 13:00 Uhr bis 20:00 Uhr*". Anstelle eines einzigen unstrukturierten Zyklus der Länge 24 Stunden (ein Tag bildet hier den Zyklus) bietet es sich an, die folgenden Teilzyklen zu definieren: `Geschlossen_Früh` (0:00-8:00 Uhr), `Vormittag` (8:00-12:00 Uhr), `Mittagspause` (12:00-13:00 Uhr), `Nachmittag` (13:00-20:00 Uhr) und `Geschlossen_ Abend` (20:00-24:00 Uhr). Eine solche Strukturierung ermöglicht den unmittelbaren Zugriff auf ganz bestimmte Zeiten bzw. Teilzyklen bezogen auf die Öffnungszeiten an einem Tag. Eine solche Strukturierung kann zur Angabe von periodischen Lieferungen verwendet werden, etwa "*Jeden Tag in der Mittagspause*" oder "*Jeden Tag in der ersten Stunde der vormittäglichen Öffnungszeit*". Ohne horizontale Strukturierung müsste für solche Beispiele stets eine manuelle Umrechnung auf die entsprechenden Zyklusspositionen erfolgen.



(a) Ohne horizontale Strukturierung



(b) Mit horizontaler Strukturierung

Abbildung 3.6: Beispiel für eine horizontale Strukturierung

3.3.3.2 Vertikale Strukturierung

Die vertikale Strukturierung entspricht der Einführung unterschiedlicher hierarchischer Ebenen, die sich alle auf die gleiche Basigranularität beziehen, d.h. die verwendete Granularität ist in allen Ebenen gleich (z.B. Tag oder Stunde)[1]. Ausgehend von einem Zyklus auf oberster Ebenen stellt jede tieferliegende Ebene eine verfeinerte horizontale Strukturierung dieses Ausgangszyklus dar. Die verschiedenen Ebenen ermöglichen unterschiedliche Sichten bzw. Abstraktionen auf die Struktur eines Zyklus in Abhängigkeit von Anwendung und Benutzer. Höhere Ebenen beschreiben lediglich eine "grobe" Sichtweise auf einen Zyklus, während tiefere Ebenen (stufenweise) eine detailliertere Sichtweise auf Zyklen bzw. deren Strukturen ermöglichen. Jede Ebene entspricht einer (endlichen) Menge an Teilzyklen, die in ihrer Gesamtheit den höheren Zyklus bilden. Diese Teilzyklen können erneut horizontale und vertikale Strukturierungen aufweisen. Die Labelzuordnungen von Zykluspositionen der tieferen Teilzyklen sind in der höheren Ebene zugreifbar. Abbildung 3.7 zeigt ein Beispiel für einen Zyklus mit vertikaler Strukturierung.

Der (äußere) strukturierte Zyklus `Chemotherapie` wird in dem Beispiel in der nächst tieferen Ebene in die beiden Teilzyklen `Medikation` und `Pause` unterteilt. Der Teilzyklus `Medikation` bildet erneut einem strukturierten Zyklus (innerer strukturierter Zyklus) mit den beiden Teilzyklen `Medikationsphase` und `Medikationsfrei`. Der Teil-

[1] Wie zu Beginn erläutert, wird im vorliegenden Abschnitt stets von einer einzelnen Granularitätsebene ausgegangen. Die vertikale Strukturierung darf hier nicht mit Granularitäten bzw. entsprechenden Granularitätsebenen verwechselt werden. In der vertikalen Strukturierung ist die zugrundeliegende Basisgranularität einer jeden Ebene stets identisch. Bei Granularitäten bzw. entsprechenden Hierarchien trifft dies jedoch nicht zu. Stattdessen bauen die Granularitätsebenen aufeinander auf. Beispielsweise basiert die Granularitätsebene *Tage* auf der Granularitätsebene *Stunden*. D.h. *eine Zyklusposition* der Ebene *Tage* setzt sich aus *24* Stunden *bzw. Zyklenpositionen* der tieferen Ebene *Stunden* zusammen. Der Umgang mit bzw. Abbildung von Granularitäten und Granularitätsebene wird in Abschnitt 3.5 besprochen.

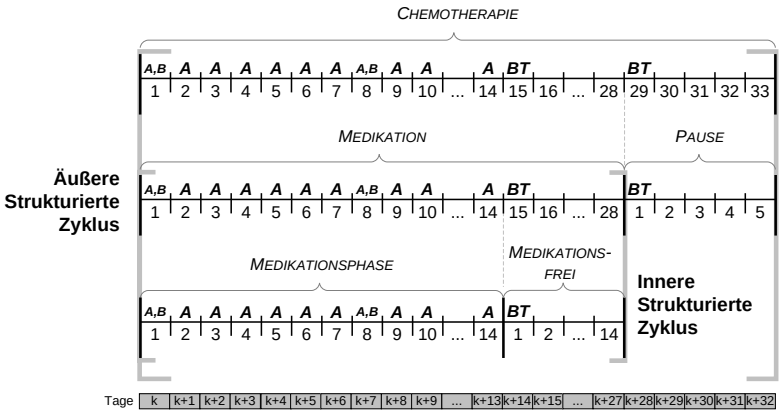


Abbildung 3.7: Beispiel für die vertikale Strukturierung eines Chemotherapiezyklus mit Medikamentengaben A und B sowie Bluttest *BT*

zyklus `Pause` wird nicht weiter verfeinert. Auf der obersten Ebene ist der Gesamtzyklus der Therapie ohne Struktur und lediglich mit der Summe an Medikamentengaben und Bluttests zu erkennen. Die Struktur der Therapie, d.h. die verschiedenen Teilzyklen, werden stufenweise in den einzelnen Ebenen beschrieben. So zeigt sich, dass die Bluttests stets am ersten Tag der Teilzyklen `Medikationsfrei` und `Pause` erfolgen.

Die Einführung unterschiedlicher hierarchischer Ebenen ermöglicht eine verständlichere Bezeichnung von Teilzyklen aus Sicht des Anwenders. Die horizontale Strukturierung auf lediglich einer einzelnen Ebene führt unter Umständen zu einer Vielzahl unterschiedlicher Teilzyklen. Bezogen auf das vorherige Beispiel der Öffnungszeiten weisen die einzelnen Teilzyklen unterschiedliche Labels auf, obwohl diese für gleiche bzw. vergleichbare semantische Zusammenhänge stehen. Die Labels `Geschlossen_Früh` und `Geschlossen_Abend` etwa bezeichnen beide Zeiten, an denen das Geschäft geschlossen ist. Je nach Komplexität und Anzahl tragen solche Benennungen nicht dazu bei, ein Periodizitätsschema für den Benutzer verständlich und nachvollziehbar zu gestalten. Eine vertikale Strukturierung ermöglicht es aber, dass gleiche semantische Zusammenhänge bzw. Zyklen auch gleiche Labels aufweisen können (z.B. `offen` und `geschlossen`).

Die Unterscheidung der Labels (z.B. von Öffnungszeiten wie oben) erfolgt über verschiedene Abstraktionsebenen. So können die Öffnungszeiten wie folgt strukturiert werden: Die oberste Ebene besteht aus einem Zyklus der Länge 24 (1 Tag). Auf der nächst tieferen Ebene kann ein Tag (bzw. Zyklus) in die drei Teilzyklen `Vormittag` (0:00 -12:00 Uhr), `Mittag` (12:00-13:00 Uhr) und `Nachmittag` (13:00-24:00) unterteilt werden. Eine weitere Ebene tiefer kann der Teilzyklus `Vormittag` in die beiden Teilzyklen `geschlossen` (0:00-8:00 Uhr) und `offen` (8:00-12:00 Uhr) unterteilt. Der Teilzyklus `Mittag` entspricht dem Teilzyklus `Pause` (12:00-13:00 Uhr) und der Teilzyklus `Nachmittag` wird in die beiden Teilzyklen `offen` (13:00-20:00 Uhr) und `geschlossen` (20:00-24:00 Uhr) zerlegt werden. Mit Hilfe von Labelausdrücken (siehe später) können die einzelnen Abstraktionsebenen

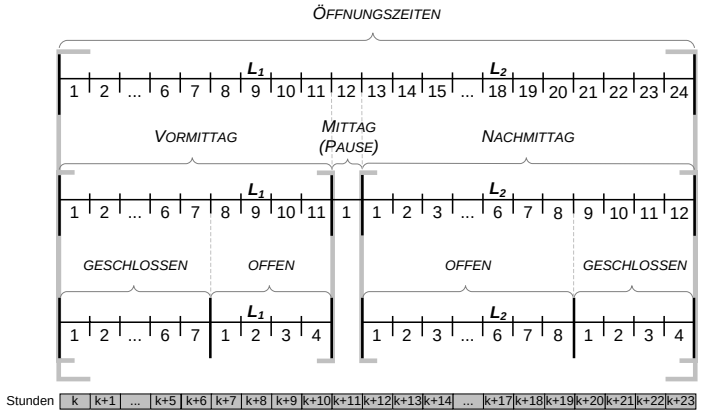


Abbildung 3.8: Vertikale Strukturierung am Beispiel von Öffnungszeiten

gezielt adressiert werden. Der Zugriff auf die Öffnungszeiten am Vormittag etwa kann navigierend entlang der Abstraktionseben erfolgen. Ausgehend von der obersten Ebene kann zunächst der Teilzyklus `Vormittag` der tieferen Ebene und dann der Teilzyklus `offen` der tiefsten Ebene adressiert werden. Der Zugriff auf die Nachmittagsöffnung erfolgt analog. Allerdings muss hier auf der zweiten Ebene der Teilzyklus `Nachmittag` adressiert werden. Ohne eine solche Strukturierung müssten solche Formen der Adressierung inflexibel durch entsprechende Benennungen von Labels realisiert werden. Eine vertikale Strukturierung ermöglicht flexiblere Möglichkeiten zur Adressierung und führt zu einer Reduktion der Anzahl unterschiedlicher Labels. In Abbildung 3.8 ist die vertikale Strukturierung für das Beispiel der Öffnungszeiten dargestellt. Die Labels $\texttt{L}_1$ und $\texttt{L}_2$ entsprechen (täglichen) Lieferungen zu den entsprechenden Zeiten.

3.3.4 Abbildung von Periodizitätsschemas

Im zyklenorientierten Modell wird ein Periodizitätsschema in Form einer unendlichen geordneten Menge von Zyklen abgebildet. Die Zyklen weisen eine absolute Ordnungsrelation auf, d.h. für jeden Zyklus kann ein eindeutiger Nachfolger in der Menge angegeben werden. Die Zyklen der Menge können hierbei unterschiedliche Längen, Labelzuweisungen oder Strukturierungen aufweisen.

Des Weiteren wird im zyklenorientierten Zeitmodell von einer Funktion ausgegangen, die in Verbindung mit einem Instanzierungspunkt, die Interpretation eines Periodizitätsschemas bezüglich eines diskreten Wertebereichs ermöglicht. Bei dieser Interpretation folgen die Zyklen ihrer Ordnungsrelation entsprechend unmittelbar aufeinander, d.h. Löcher bzw. Abstände zwischen Intervallen sind nicht zulässig.

In Abbildung 3.9 ist ein Beispiel für die Abbildung eines Periodizitätsschemas im zyklenorientierten Modell dargestellt. In der Abbildung wurde als Instanziierungszeitpunkt die 1 bei der entsprechenden Interpretation des Periodizitätsschemas angenommen.

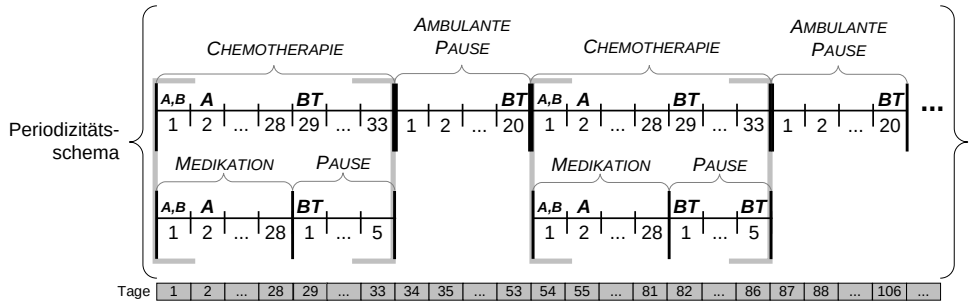


Abbildung 3.9: Abbildung von Periodizitätsschemas im zyklenorientierten Modell

Das dargestellte Beispiel beschreibt eine Chemotherapie, bei der nach jedem Chemotherapiezyklus ein ambulanter Pausenzyklus stattfindet (im Gegensatz zu dem stationären Aufenthalt bei "normalen" Pausenzyklen `Pause`). Das Periodizitätsschema entspricht somit einer (unendlichen) Menge sich abwechselnder Zyklen `Chemotherapie` und `ambulante_Pause`. Der Zyklus `Chemotherapie` ist ein strukturierter Zyklus bestehend aus den beiden Teilzyklen `Medikation` und (stationäre) `Pause`. Allerdings unterschieden sich die Zyklen `Chemotherapie` leicht. Im zweiten Zyklus findet ein zusätzlicher Bluttest am Ende der Pause statt. Ein solcher Unterschied kann beispielsweise das Ergebnis einer Ausnahme sein.

Abschließend werden in Abbildung 3.10 zum Vergleich mit den in Abschnitt 3.2 vorgestellten Zeitmodellen die Repräsentationen der selben Beispiele im zyklenorientierten Zeitmodell dargestellt. Für die Chemotherapie, die abstandsorientierte Angabe "*Jeden 3. Tag*" sowie die Granularitäten *Woche* und *Geschäftswoche* wurde jeweils ein unstrukturierter Zyklus angenommen sowie als Instanziierungszeitpunkt 1 gewählt.

Das zyklenorientierte Zeitmodell bildet den ersten Schritt zur Erfüllung der Anforderung nach Verständlichkeit. Die im nachfolgenden Abschnitt vorgestellte algebraische Notationsform stellt einen weiteren Schritt hin zu intuitiv verständlichen, logischen Beschreibungen von Periodizitätsschemas dar.

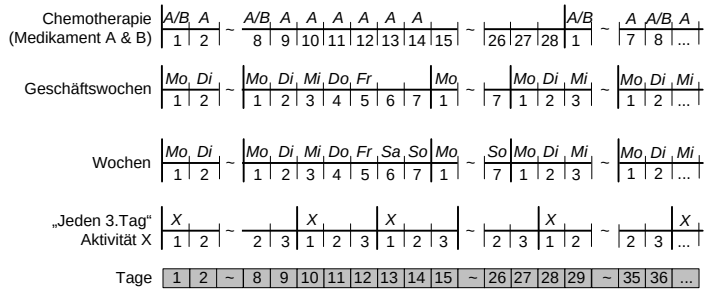


Abbildung 3.10: Periodizitätsschemas im zyklenorientierten Modell

3.4 Algebraische Notation von Periodizitätsschemas

Eine verständliche und nachvollziehbare Notationsform zur Beschreibung von Periodizitätsschema ist essentiel im Zusammenhang mit der Anforderung nach Verständlichkeit. Insbesondere die Unterstützung von Ausnahmen und die damit verbundene Forderung nach einer expliziten Spezifikationslogik ist wesentlich. In diesem Abschnitt wird, basierend auf dem zyklenorientierten Zeitmodell, eine algebraische Notation zur Beschreibung periodischer Spezifikationen bzw. entsprechender Periodizitätsschemas vorgestellt, die diesen Anforderungen begegnen kann.

Grundsätzlich basiert unsere algebraische Notation auf *Labelausdrücken* sowie den beiden Operationsklassen *Änderungsoperationen* und *Auswahloperationen.* Die Semantik der Operationen wird im Folgenden zunächst informell anhand von Beispielen illustriert. Eine formale Definition erfolgt in Kapitel 4.

3.4.1 Labelausdrücke

Labels dienen der anwendungsabhängigen Bezeichnung und Adressierung von Zyklen und Zyklenpositionen. In der Praxis reichen meist einfache "direkte Labelangaben" (d.h. die Angabe eines einzelnen Labels) zur Adressierung nicht aus. Stattdessen sollen Labelangaben auch logisch miteinander kombiniert bzw. verknüpft werden können. Im zyklenorientierten Modell stehen hierfür *Labelausdrücke* zur Verfügung. Labelausdrücke ermöglichen eine semantisch erweiterte Adressierung von (Teil-) Zyklen und Zyklenpositionen eines Periodizitätsschemas. Das Ergebnis der Adressierung bzw. eines Labelausdrucks ist eine (theoretisch unendliche) Menge von bestimmten Zyklen oder Zyklenpositionen eines entsprechenden Periodizitätsschemas. Die Grundlage von Labelausdrücken bilden "Verkettungen", die die einzelnen Labelangaben miteinander verknüpfen. Ein Labelausdruck wie "`Chemotherapie`$\triangledown$`A`" mit Verkettung $\triangledown$ adressiert alle Zykluspositionen mit Label `A` (d.h. alle Gaben von Medikament `A`) eines jeden Zyklus mit Label `Chemotherapie` (siehe später).

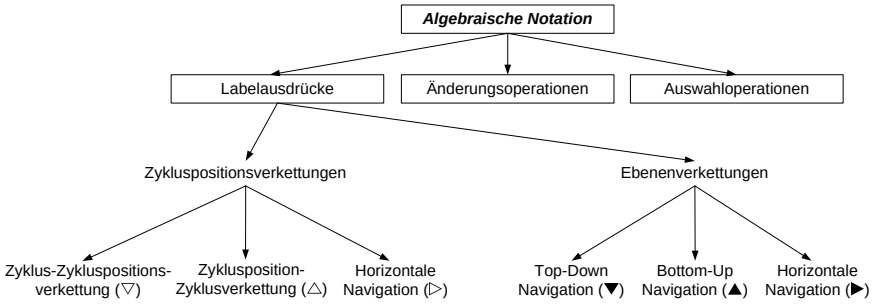


Abbildung 3.11: Übersicht zu Verkettungen in Labelausdrücke

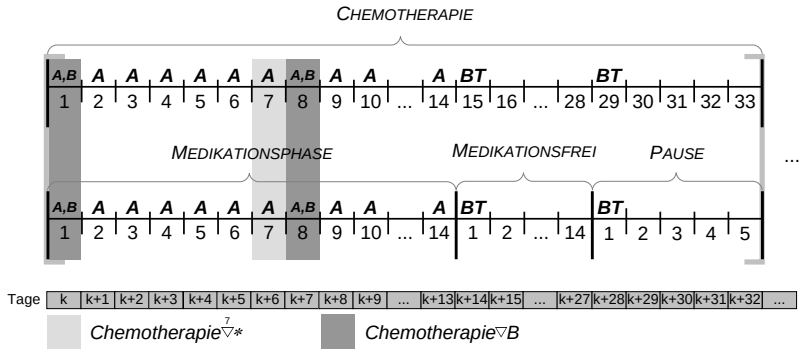


Abbildung 3.12: Beispiele einer Zyklus-Zykluspositionsverkettung

Generell können im zyklenorientierten Modell zwei Klassen von Verkettungen unterschieden werden: *Zykluspositionsverkettungen* und *Ebenenverkettungen*. Abbildung 3.11 zeigt eine Übersicht zu den Verkettungen in den einzelnen Klassen.

3.4.1.1 Zykluspositionsverkettungen

Die Klasse von Verkettungen ermöglicht eine unterschiedliche Adressierung von Zyklenpositionen innerhalb von (Teil-) Zyklen:

Zyklus-Zykluspositionsverkettung ($\overset{x}{\triangledown}$): Die Verkettung adressiert Zykluspositionen anhand ihrer absoluten Positionsnummer innerhalb von (Teil-) Zyklen. So liefert der Labelausdruck "`Chemotherapie`$\overset{7}{\triangledown}*$" die siebte Zyklusposition von jeden Zyklus `Chemotherapie` zurück. Die abschließende Wildcard ($*$) drückt aus, dass die adressierten Zykluspositionen eine beliebige Labelzuordnung aufweisen können. Im Gegensatz dazu adressiert der Ausdruck "`Chemotherapie`$\overset{7}{\triangledown}$`A`" nur die Zykluspositionen mit Position sieben in allen Zyklen `Chemotherapie`, die zusätzlich das Label `A` an dieser Position aufweisen.

Neben der Angabe einer absoluten Positionsnummer können auch spezielle Angaben verwendet werden, wie "*Erste*" bzw. "*Letzte*". So können zum Beispiel Angaben wie "*Erster Tag einer Chemotherapie*" bzw. "*Letzter Tag einer Chemotherapie*" mit den Labelausdrücken "`Chemotherapie`$\overset{first}{\triangledown}*$" bzw. "`Chemotherapie`$\overset{last}{\triangledown}*$" adressiert werden. Die Angabe einer Positionsnummer kann allerdings auch entfallen. Dann werden alle Zykluspositionen des Zyklus adressiert. So adressiert der Ausdruck "`Chemotherapie`$\triangledown*$ alle Zykluspositionen der Zyklen `Chemotherapie` bzw. der Ausdruck "`Chemotherapie`$\triangledown$`B`" alle Gaben von Medikament `B`. Abbildung 3.12 zeigt einige der vorherigen Beispiele. Auch wenn in dieser und in den folgenden Abbildungen lediglich ein einzelner Zyklus dargestellt ist, bezieht sich ein Labelausdruck stets auf alle Zyklen bzw. strukturierten Zyklen der (unendlichen) Menge eines Periodizitätsschemas.

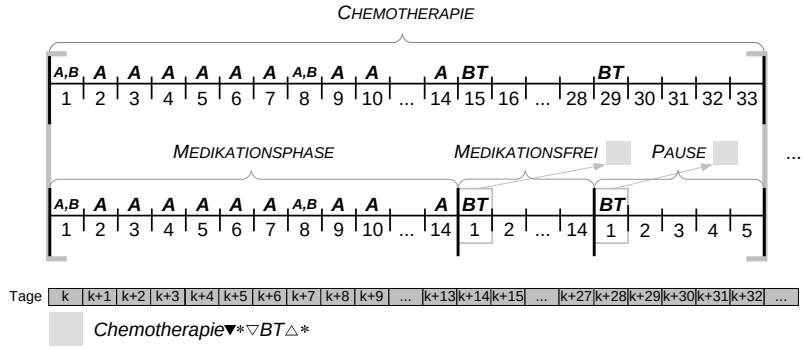


Abbildung 3.13: Beispiele einer Zykluspositions-Zyklusverkettung

Zyklusposition-Zyklusverkettung (△): Ausgehend von einer einzelnen Zyklusposition oder einer Menge von Zyklusspositionen adressiert diese Verkettung die zugehörigen (Teil-) Zyklen. Beispielsweise adressiert der Ausdruck "`Chemotherapie`▼ ∗ ▽`BT`" alle Bluttests (`BT`) innerhalb einer Chemotherapie unabhängig von dem konkreten Teilzyklus (vgl. Abschnitt 3.4.1.2). Die Ergänzung dieses Ausdrucks um die Verkettung △, d.h. "`Chemotherapie`▼ ∗ ▽`BT` △ ∗ adressiert alle Teilzyklen einer Chemotherapie in denen ein Bluttest vorkommt. In dem Beispiel aus Abbildung 3.13 werden auf diese Art und Weise die Teilzyklen `Medikationsfrei` und `Pause` adressiert. Wichtig ist, dass zunächst auf die Ebene navigiert wird (mit der Verkettung ▼), in der sich die Teilzyklen befinden. Ohne Navigation ▼, etwa im Ausdruck "`Chemotherapie`▽`BT` △ ∗, würden lediglich alle Zyklen `Chemotherapie` zurückgeliefert.

Horizontale Navigation ($\overset{\pm x}{\triangleright}$): Diese Verkettung ermöglicht die Adressierung von Vorgängern bzw. Nachfolgern von Zyklusspositionen. Das heißt, ausgehend von einer bestimmten Zyklusposition werden mit gegebenem Abstand x die nachfolgenden ($+x$) bzw. die vorhergehenden Zyklusspositionen ($-x$) eines (Teil-) Zyklus adressiert. Zum Beispiel werden mit dem Labelausdruck "`Chemotherapie`$\overset{2}{\triangledown}$∗ $\overset{+3}{\triangleright}$ ∗" ausgehend von der zweiten Zyklusposition, drei Zyklusspositionen nach rechts navigiert. Ein weiteres Beispiel ist der Ausdruck "`Chemotherapie`▽`BT` $\overset{+1}{\triangleright}$ ∗", der die Zyklusspositionen einen Tag nach einem Bluttest adressiert. Die Beispiele sind in Abbildung 3.14 dargestellt. Allerdings ist für den Parameter x der Verkettung auch der Wert Null zulässig. In diesem Fall bleibt die Ausgangsposition bestehen und es können mehrfache Belegungen von Labels einer Zyklusposition übeprüft werden. Der Ausdruck "`Chemotherapie`$\overset{8}{\triangledown}$∗ $\overset{0}{\triangleright}$ `A` $\overset{0}{\triangleright}$ `B` " liefert alle Zyklusspositionen mit Position acht zurück, die sowohl das Label `A` als auch das Label `B` aufweisen.

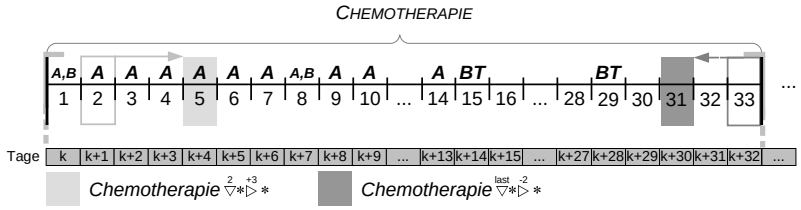


Abbildung 3.14: Beispiele einer horizontalen Navigation bei Zyklusspositionen

3.4.1.2 Ebenenverkettungen

Diese Klasse von Verkettungen ermöglicht die Adressierung von Zyklen und Teilzyklen entlang der vertikalen und horizontalen Strukturierung. Hierbei können die folgenden drei Grundformen unterschieden werden:

Top-Down Navigation ($\overset{x}{\blacktriangledown}$): Diese Verkettung navigiert – ausgehend von der höchsten Ebene – entlang der vertikalen Strukturierung nach *"unten"*. Ausgehend von einem Zyklus oder Teilzyklus werden mit der Verkettung bestimmte Teilzyklen der nächst tieferliegenden (vertikalen) Ebene adressiert. Unter Verwendung des Parameters x können Teilzyklen anhand ihrer absoluten Position bezüglich des höheren Zyklus adressiert werden. Der Ausdruck "`Chemotherapie`$\overset{+2}{\blacktriangledown}*$" adressiert den zweiten Teilzyklus der `Chemotherapie`, d.h. den Teilzyklus `Pause`. Die zulässigen Werte für den Parameter x werden durch die Anzahl an Teilzyklen einer konkreten vertikale Ebene bestimmt. Die Angabe des Parameters x ist jedoch nicht zwingend und kann entfallen. In einem solchen Fall werden alle Teilzyklen unabhängig von ihrer Position adressiert. So adressiert der Labelausdruck "`Chemotherapie▼Medikation▼Medikationsfrei`", ausgehend vom Zyklus `Chemotherapie` alle Teilzyklen mit Label `Medikation` und im zweiten Schritt alle Teilzyklen mit Label `Medikationsfrei`. Der Labelausdruck entspricht der Angabe "*alle Teilyzklen* `Medikationsfrei` *einer Chemotherapie*". Neben konkreten Labels können auch Wildcards (*) in einem Labelausdruck verwendet werden. In diesem Fall werden alle Teilzyklen der tieferliegenden Ebene adressiert. Der Labelausdruck "`Chemotherapie▼`$*$" adressiert alle Teilzyklen innerhalb einer Chemotherapie. Beide Beispiele sind in Abbildung 3.15 dargestellt. Die ausgefüllten Quadrate in der Abbildung symbolisieren das Ergebnis des Labelausdrucks für den dargestellten Zyklus. Die nichtausgefüllten, umrandeten Quadrate stellen die Zwischenergebnisse der einzelnen Verkettung dar.

Bottom-Up Navigation (▲): Diese Verkettung navigiert – ausgehend von den tiefsten Ebenen – entlang der vertikalen Strukturierung nach *"oben"* auf die nächst höherliegende Ebene. Die tiefsten Ebenen werden durch die Zyklen gebildet, die innerhalb eines strukturierten Zyklus nicht weiter verfeinert werden. Zum Beispiel adressiert der Labelausdruck "`Medikationsfrei▲`$*$" ausgehend vom Teilzyklus `Medikationsfrei` auf der "tiefsten Ebene" alle (Teil-) Zyklen der nächst höheren Ebene, d.h. den Teilzyklus `Medikation` im vorliegenden Beispiel. Dieses und ein weiteres Beispiel zeigt Abbildung 3.16.

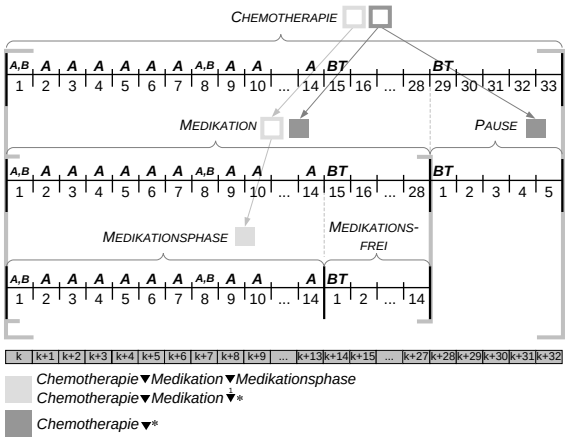


Abbildung 3.15: Beispiel einer Top-Down Navigation

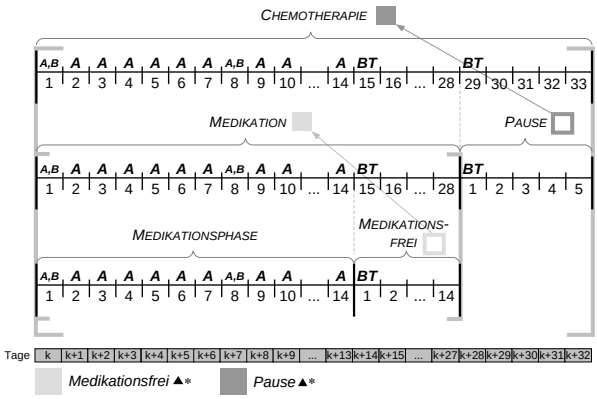


Abbildung 3.16: Beispiel einer Bottom-Up Navigation

Horizontale Navigation ($\overset{\pm x}{\blacktriangleright}$): Diese Verkettung navigiert entlang der horizontalen Strukturierung eines Zyklus auf einer (vertikalen) Ebene. In Abhängigkeit vom Vorzeichen des Parameters x werden entweder die Vorgänger ($-x$) oder Nachfolger ($+x$) adressiert. Der Ausdruck "`Chemotherapie▼Medikation▼Medikationsphase` $\overset{+1}{\blacktriangleright}$ $*$" adressiert den Nachfolgerzyklus des Teilzyklus `Medikationsphase`, d.h. den Teilzyklus `Medikationsfrei`. Die Wildcard ($*$) am Ende bedeutet hier, dass der Teilzyklus, auf den horizontal hin navigiert wird, ein beliebiges Label aufweisen darf.

Die zulässigen Werte für den Parameter x der Verkettung werden durch die Anzahl an Teilzyklen, in Abhängigkeit von der konkreten vertikalen Ebene, bestimmt. Ein horizontales "Hinausnavigieren" aus einem strukturierten Zyklus ist auf Ebene der Teilzyklen nicht möglich. Eine Einschränkung bildet jedoch die Anwendung der horizontalen Navigation auf der obersten Ebene. Lediglich in diesem Fall ist ein horizontales Navigieren innerhalb der Menge an Zyklen eines Periodizitätsschemas möglich. Der Ausdruck

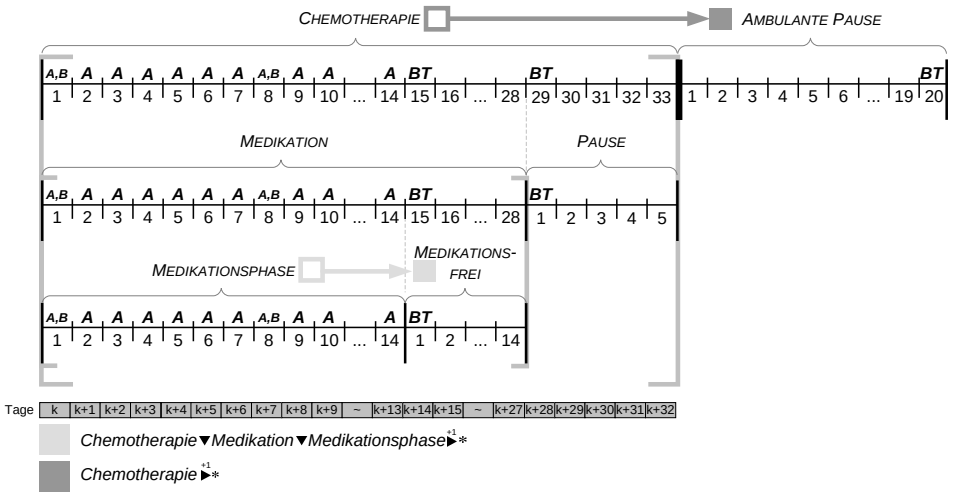


Abbildung 3.17: Beispiel einer horizontalen Navigation

"`Chemotherapie` $\overset{+1}{\blacktriangleright}$ $*$" adressiert somit die unmittelbaren Nachfolger aller strukturierten Zyklen `Chemotherapie` aus der (geordneten) Menge von Zyklen des entsprechenden Periodizitätsschemas. Beide vorherigen Beispiele sind in Abbildung 3.17 dargestellt.

Die vorgestellten Labelausdrücke des zyklenorientierten Zeitmodells bilden ohne Anspruch auf Vollständigkeit eine ausdrucksstarke Möglichkeit zur Adressierung von Strukturen, wie Zyklen, Teilzyklen, Zyklenpositionen sowie ihrer horizontalen und vertikalen Strukturierungen. Aufgrund der Kombinierbarkeit der Verkettungen können so auch komplexe Adressierungen vorgenommen werden. Allerdings gilt es hier, die Eingabe- und Ausgabetypen einer Verkettung zu berücksichtigen. Abschließend sind in Tabelle 3.1 die vorgestellten Verkettungen inklusive ihrer Ein- und Ausgabetypen sowie in Abbildung 3.18 eine BNF-Notation der Grammatik für Labelausdrücke.[2]

Verkettung		Eingabetyp (in)	Ausgabetyp (out)
Zyklus-Zyklusposition	in $\overset{x}{\triangledown}$ out	$(Teil-)\ Zyklus$	$Zyklusposition$
Zyklusposition-Zyklus	in $\triangle$ out	$Zyklusposition$	$(Teil-)\ Zyklus$
Horizontale Navigation	in $\overset{\pm x}{\triangleright}$ out	$Zyklusposition$	$Zyklusposition$
Top-Down Navigation	in $\overset{x}{\blacktriangledown}$ out	$(Teil-)\ Zyklus$	$(Teil-)\ Zyklus$
Bottom-Up Navigation	in $\blacktriangle$ out	$(Teil-)\ Zyklus$	$(Teil-)\ Zyklus$
Horizontale Navigation	in $\overset{\pm x}{\blacktriangleright}$ out	$(Teil-)\ Zyklus$	$(Teil-)\ Zyklus$

Tabelle 3.1: Übersicht zu Eingabe- und Ausgabetypen von Verkettungen

[2]Das Symbol $ZPOS$ steht in der BNF-Notation für die Zyklusposition.

<Labelausdruck> := <ZZ> | <ZP>

<ZZ> ::= <ZYKLUS> <zyk> (<ZYKLUS> | <ZZ> | <ZP>)
<ZP> ::= <ZYKLUS> <pos> (<ZPOS> | <PP> | <PZ> | <PP> <PZ>)
<PP> ::= <ZPOS> $\overset{+x}{\triangleright}$ (<ZPOS> | <PP>) | <ZPOS> $\overset{-x}{\triangleright}$ (<ZPOS> | <PP>)
<PZ> ::= <ZPOS> △ <ZYKLUS> | <ZPOS> △ <ZZ>

<zyk>::= ▼ | $\overset{x}{\blacktriangledown}$ | $\overset{x-y}{\blacktriangledown}$ | $\overset{first}{\blacktriangledown}$ | $\overset{last}{\blacktriangledown}$ | $\overset{+x}{\blacktriangleright}$ | $\overset{-x}{\blacktriangleright}$
<pos>::= ▽ | $\overset{x}{\triangledown}$ | $\overset{x-y}{\triangledown}$ | $\overset{first}{\triangledown}$ | $\overset{last}{\triangledown}$

<ZYKLUS> ::= <Bezeichner>
<ZPOS> ::= <Bezeichner>
<Bezeichner> ::= (<Buchstabe> | <Ziffer>) | {(<Buchstabe> | <Ziffer>)} | *
<Buchstabe> ::= A | B | C | D | ... | Z | a | b | ... | z
<Ziffer> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Abbildung 3.18: BNF-Notation für Labelausdrücke

3.4.2 Änderungsoperationen

Diese Operationsklasse dient der Änderung bzw. dem generellen Aufbau von Periodizitätsschemas. Unabhängig von der konkreten Semantik weisen alle Änderungsoperationen die folgende, vergleichbare Signatur auf:

$\otimes(\mathbf{PS}, \mathbf{ps}, \mathbf{p_1}, ..., \mathbf{p_n}) \mapsto \mathbf{PS}^*$

Input:	
PS	Periodizitätsschema
ps	Teilmenge von Zyklen, Teilzyklen oder Zykluspositionen des Periodizitätsschemas PS.
$p_1, ..., p_n$	Von der konkreten Operation abhängige Parameter
Output:	
PS^*	Geändertes Periodizitätsschemas

Jeder Änderungsoperation werden ein Periodizitätsschema (PS) sowie eine Teilmenge (ps) an Strukturen des Schemas übergeben. Lediglich diese Teilmenge wird durch die Änderungsoperation manipuliert. Ihre Angabe kann durch einen Labelausdruck erfolgen. Je nach Semantik können weitere Operationsparameter ($p_1, ..., p_n$) notwendig sein. Das Ergebnis der Operation ist ein Periodizitätsschema (PS^*), welches das Periodizitätsschema PS mit den gemäß der Operationsemantik geänderten Strukturen (ps) repräsentiert. Die Semantik der einzelnen Operationen wird nachfolgend vorgestellt.

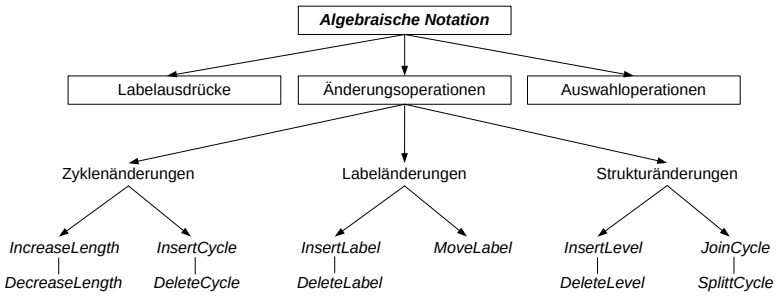


Abbildung 3.19: Übersicht zu den Änderungsoperationen

In Bezug auf die Grundelemente des zyklenorientierten Modells können Änderungen an (Teil-) Zyklen, Labelzuordnungen sowie an der horizontalen und vertikalen Strukturierung unterschieden werden. Abbildung 3.19 zeigt die im folgenden vorgestellten Änderungsoperationen in Übersicht.

3.4.2.1 Zyklenänderungen

Folgende Operationen stehen für Änderungen von Zyklen zur Verfügung:.

Verlängern/Verkürzen der Zykluslänge (*IncreaseLength*/*DecreaseLength*): Diese Operation ermöglicht das Verlängern bzw. Verkürzen von (Teil-) Zykluslängen. In Abbildung 3.20 ist dies anhand eines Zyklus bestehend aus den zwei Teilzyklen `A` und `B` dargestellt. Bei entsprechender Interpretation des Periodizitätsschemas führt die Verlängerung der Zykluslänge von Teilzyklus `A` zu einer Verschiebung von Teilzyklus `B` im zugrundeliegenden Wertebereich.

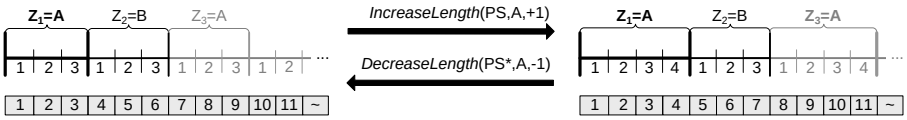


Abbildung 3.20: Verlängern (Verkürzen) von Zyklenlängen

Einfügen/Löschen von Zyklen (*InsertCycle*/*DeleteCycle*): Mit der Operation $InsertCycle(PS, ps, p_1)$ wird als unmittelbarer Nachfolger der (Teil-) Zyklen ps, der ebenfalls übergeben (Teil-) Zyklus p_1 eingefügt. Im Gegensatz dazu wird mit der Operation *DeleteCycle* der übergebene (Teil-) Zyklus gelöscht. In dem dargestellten Beispiel in Abbildung 3.21 wird zwischen den Teilzyklen `A` und `B` der Teilzyklus `C` eingefügt (bzw. gelöscht). Beispielsweise kann so nachträglich eine Pause zwischen zwei Medikationsteilzyklen `A` und `B` eingefügt bzw. gelöscht werden.

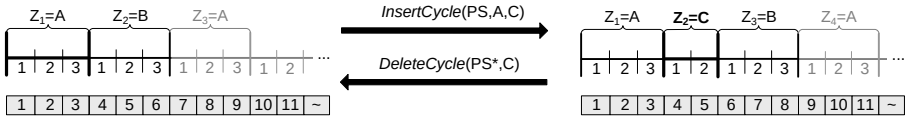


Abbildung 3.21: Einfügen (Löschen) von Teilzyklen

3.4.2.2 Labeländerungen

Diese Operationen umfassen Änderungen an der Zuordnung von Labels zu Zyklen und Zyklusposition. Es können die zwei folgenden Formen unterschieden werden.

Einfügen/Löschen von Labels (*InsertLabel*/*DeleteLabel*): Das Einfügen oder Löschen von Labels entspricht dem Zuweisen (Aufheben) von Labels zu Zyklen oder Zyklenpositionen. In Abbildung 3.22 etwa werden die Labels `M` in den Teilzyklen `B` gelöscht (bzw. an der Zyklusposition 2 eingefügt).

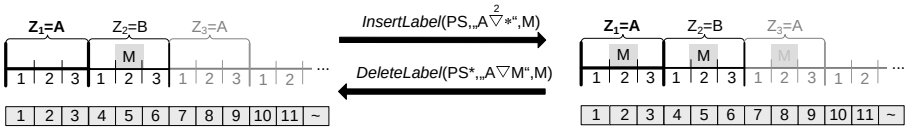


Abbildung 3.22: Einfügen (Löschen) von Labels

Verschieben von Labels (*MoveLabel*): Diese Operation dient dem Verschiebenen einzelner Labels. Allerdings ist das Verschieben eines Labels nur zwischen seinem Vorgängerlabel (Teilyzklus oder Zyklusposition mit gleichem Label) und seinem Nachfolgerlabel möglich. Ein Verschieben über das Vorgänger- oder Nachfolgerlabel hinaus ist im zyklenorientierten Modell nicht zulässig, da sonst der kausale Zusammenhang zwischen den einzelnen Labels, d.h. die zeitliche Reihenfolge ihres Auftretens, verloren gehen würde. Grundsätzlich ist es es anwendungsabhängig, wie schwerwiegend die Verletzung der Kausalität an dieser Stelle ist. Um jedoch "zufällige" bzw. unerwünschter Kausalitätsverletzungen zu vemeiden, besteht im zyklenorientierten Modell die entsprechende Einschränkungen. Zur Abbildung solcher Kausalitätsverletzungen müsste das betroffene Label im zyklenorientierten Modell gelöscht und an der gewünschten Stelle eingefügt werden.

Nicht berücksichtigt werden müssen hingegen die Zyklusgrenzen bei einer Verschiebung. Das Verschieben von Labels darf hier auch bis in den nachfolgenden (vorherigen) Zyklus hinein erfolgen. Anstatt eine Gabe eines Medikaments innerhalb einer Pause zu verabreichen, kann dies auch bis in den nachfolgenden Chemotherapiezyklus "hinein" verschoben werden. Abbildung 3.23 zeigt die Verschiebung des Labels `A` in Teilzyklus `B` über dessen Grenzen hinweg.

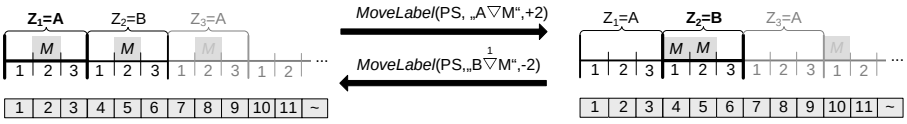


Abbildung 3.23: Verschieben von Labels

3.4.2.3 Änderung der horizontalen und vertikalen Strukturierung

Diese Operationen umfassen Änderungen an der horizontalen (vertikalen) Strukturierung entsprechender Zyklen.

Einfügen / Löschen von Ebenen (*InsertLevel*/*DeleteLevel*): Das Einfügen einer neuen vertikalen Ebene erfolgt unterhalb des übergebenen Ausgangszyklus. Der Ausgangszyklus wird unverändert auf die neue Ebene kopiert. Das Löschen einer Ebene betrifft die nächst tieferliegende vertikale Ebene des übergebenen (Teil-) Zyklus. Weitere Ebenen unterhalb der gelöschten Ebene werden ebenfalls gelöscht. Abbildung 3.24 zeigt dies für ein Beispiel.

Abbildung 3.24: Einfügen (Löschen) von vertikalen Ebenen

Zusammenfassen von / Aufteilen in Teilzyklen (*JoinCycle*/*SplittCycle*)): Diese Änderung ermöglicht es, einen Zyklus in mehrere Teilzyklen aufzuteilen bzw. bestehende Teilzyklen wieder zu einem einzelnen Zyklus zusammenzufassen. Die Gesamtlänge des Zyklus, der die Teilzyklen beinhaltet, verändert sich nicht. In Abbildung 3.25 etwa werden die Teilzyklen `A` und `B` wieder zu einem einzelnen Zyklus `AB` zusammengefasst (bzw. der Zyklus `AB` in die Teilzyklen `A` und `B` aufgeteilt).

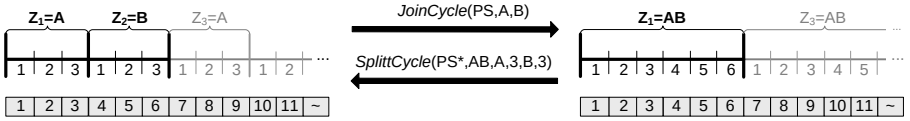


Abbildung 3.25: Zusammenfassen (Aufteilen) von Teilzyklen

Neben den vorgestellten Änderungsoperationen bilden die Auswahloperationen die zweite große Operationsklasse für die algebraische Beschreibung von Periodizitätsschemas. Diese werden im folgenden Abschnitt vorgestellt.

3.4.3 Auswahloperationen

Auswahloperationen ermöglichen es dem Anwender, basierend auf einem Auswahlschema, bestimmte Zyklen, Teilzyklen oder Zykluspositionen eines Periodizitätsschemas auszuwählen bzw. zu markieren. Die ausgewählten Strukturen können dann mit den vorgestellten Änderungsoperationen manipuliert werden. Unabhängig von ihrer konkreten Semantik weisen alle Auswahloperationen die folgende Signatur auf:

$\odot(\mathbf{PS}, \mathbf{ps}, \mathbf{AS}, \mathbf{p_1}, ..., \mathbf{p_n}) \mapsto \mathbf{PS}^*$	
Input:	
PS	Periodizitätsschema
ps	Auswahlmenge, d.h. Teilmenge von Zyklen, Teilzyklen oder Zykluspositionen des Periodizitätsschemas PS, aus der bestimmte Elemente ausgewählt werden sollen.
AS	Auswahlschema
$p_1, ..., p_n$	Von der konkreten Operation abhängige Parameter
Output:	
PS^*	Periodizitätsschema mit Auswahl

Bei jeder Auswahloperation werden ein Periodizitätsschema (PS), eine Auswahlmenge (ps) sowie ein Auswahlschema (AS) übergeben. Die Menge ps entspricht der (Teil-) Menge an Strukturen des Schemas PS, aus der bestimmte Elemente ausgewählt werden sollen. Welche dieser Elemente konkret von einer Auswahl betroffen sind, hängt vom Auswahlschema AS und der Semantik der Auswahloperation ab. Bei der Angabe "*Jeden 3.Teilzyklus Medikation* (*erfolgt am ersten Tag ein Bluttest*)" beispielsweise entsprechen die Chemotherapie dem Periodizitätsschema PS und die Auswahlmenge ps der Menge an Zyklen mit Label `Medikation`. Aus Letzteren soll mit dem Auswahlschema AS = "*Jeder 3. Teilzyklus*" ausgewählt werden soll.

Unabhängig von der konkreten Semantik einer Auswahloperation, erfolgt die Beschreibung der auszuwählenden Werte oder (Teil-) Zyklen durch ein entsprechendes Auswahlschema. Allerdings können diese Auswahlschemas unterschiedliche Ausprägungen aufweisen, die im Folgenden vorgestellt werden. Die unterschiedlichen Semantiken von Auswahloperationen werden im Anschluss daran vorgestellt.

3.4.3.1 Ausprägungen von Auswahlschemas

Im Allgemeinen kann zwischen *periodischen* und *nichtperiodischen Auswahlschemas* unterschieden werden. Während mit Ersteren Auswahlschemas periodisch bestimmte (Teil-) Zyklen oder Zykluspositionen eines Periodizitätsschemas ausgewählt werden, erfolgt dies bei nichtperiodischen Schemas lediglich einmalig. So stellt die Aussage "*Jeden 3.*

Zyklus" ein periodisches Auswahlschema dar und die Aussage "*Ausschließlich am 3. Zyklus*" ein nichtperiodisches Auswahlschema.

Im Zusammenhang mit geforderten Ableitbarkeit von Wiederholungsmustern (siehe Abschnitt 3.1.6) wurde bereits in Abschnitt 2.3.3 gezeigt, dass nichtperiodische Ausnahmen durch Wiederholungsmuster nicht darstellbar sind. Auch wenn das zyklenorientierte Zeitmodell grundsätzlich eine nichtperiodische Auswahl in Periodizitätsschema abbilden kann, könnten für solche Periodizitätsschemas ebenfalls keine Wiederholungsmuster abgeleitet werden. Um der Anforderung nach Ableitbarkeit von Wiederholungsmustern zu begegnen, werden im Folgenden lediglich periodische Auswahlschemas im Vordergrund stehen. Die Abbildung nichtperiodischer Auswahlschemas bzw. entsprechender Auswahlen wird in Abschnitt 3.5 behandelt.

Eine wesentliche Eigenschaft des zyklenorientierten Zeitmodells ist es, dass periodische Auswahlschemas analog zu herkömlichen Periodizitätsschemas beschrieben werden können. D.h. ein periodisches Auswahlschema wird, wie normale Periodizitätsschemas auch, durch Zyklen, Zykluspositionen und Labels beschrieben. Ebenso können Auswahlschemas horizontal und vertikal strukturiert sein und erneut Ausnahmen enthalten. Der wesentliche Unterschied zwischen Auswahlschema und Periodizitätsschema besteht im jeweils zugrundeliegenden Wertebereich. Die Interpretation von normalen Periodizitätsschemas erfolgt über einen diskreten Wertebereich etwa der, der natürlichen Zahlen (siehe Abschnitt 3.9). Die Interpretation von Auswahlschemas erfolgt analog, allerdings über die bei der Auswahloperation übergebenen Auswahlmenge *ps*. Formal bildet die Auswahlmenge ebenfalls einen (unendlichen) diskreten Wertebereich, in dem ein einzelner Wert einem Teilzyklus oder einer Zyklusposition entspricht. Im Folgenden werden verschiedene Ausprägungen von periodischen Auswahlschemas anhand von Beispielen vorgestellt.

Einfache Auswahlschemas: Angaben wie etwa "*Jeden 2.Medikationsteilzyklus einer Chemotherapie beginnend beim Ersten*" oder "*Jeder 10. Wert eines Schemas*" bilden Beispiele für einfache äquidistante Auswahlschemas. Beispielsweise entspricht die erste Angabe einem einfachen (äquidistanten) Auswahlschema mit Zykluslänge zwei und Zyklusposition eins. Dieses Beispiel sowie die Interpretation des Auswahlschemas über den Wertebereich ist in Abbildung 3.26 dargestellt. Der Wertebereich entspricht der Menge aller Medikationsteilzyklen (Label = `Med`) einer Chemotherapie. Das *Auswahllabel* `PA` (**P**eriodische **A**uswahl) bezeichnet die auszuwählenden Medikationsteilzyklen, d.h. "*Jeden 2. Medikationsteilzyklus beginnend beim Ersten*".

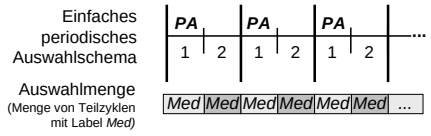


Abbildung 3.26: Beispiel für ein einfaches periodisches Auswahlschema

Die Labels des Auswahlschemas ermöglichen – nach erfolgter Auswahloperation – die Adressierung der von der Auswahl betroffenen Strukturen in dem entsprechenden Periodizitätsschema. D.h. die Labels des Auswahlschemas, aber auch dessen Struktur (siehe später), werden auf das Periodizitätsschema durch die Auswahloperation übertragen. Bezogen auf das Beispiel in Abbildung 3.26 bedeutet dies, dass jeder 2. Medikationsteilzyklus der Chemotherapie nach erfolgter Auswahloperation im Periodizitätsschema mit dem Label `PA` gekennzeichnet und somit adressierbar ist.

Die in Auswahlschemas verwendeten Labels sind anwendungsabhängig. Im oberen Beispiel wurde etwa das Label `PA` zur Bezeichnung der von der Auswahl betroffenen Medikationsteilzyklen verwendet. In der Praxis ist es häufig auch notwendig, die Gegenmenge zu bezeichnen, d.h. die nicht von der Auswahl betroffenen Strukturen. Dies erfolgt analog zu oben über ein spezielles Label, etwa ¬`PA`. Im Beispiel in Abbildung 3.26 könnten so die Zykluspositionen zwei mit einem solchen Label belegt werden, um so alle Medikationsteilzyklen, die nicht von der Auswahl betroffen sind zu bezeichnen.

Strukturierte Auswahlschemas: Die Betrachtung eines Auswahlschemas als lediglich einen unstrukturierten Zyklus mit konstanter Länge reicht häufig nicht aus. Stattdessen werden auch Auswahlschema benötigt, die wiederum horizontale und vertikale Strukturierungen aufweisen. Eine solche Strukurierung des Auswahlschemas führt – nach erfolgter Auswahloperation – zu einer Strukturierung bzw. Unterteilung der Gesamtmenge der von der Auswahl betroffenen Strukturen eines Periodizitätsschemas. In Abbildung 3.27 etwa ist ein Beispiel für ein strukturiertes Auswahlschema mit der Auswahlmenge "*Alle Gaben von Medikament A einer Chemotherapie*" dargestellt.

Nach erfolgter Auswahloperation bezeichnet der "oberste" Zyklus mit Label $\texttt{PA}_{\texttt{Gesamt}}$ die Menge aller ausgewählten Medikamentengaben. Durch die Strukturierung des Auswahlschemas wird die Gesamtmenge $\texttt{PA}_{\texttt{Gesamt}}$ zusätzlich in die zwei Teilmengen $\texttt{PA}_0$ und $\texttt{PA}_1$ unterteilt. Der Zugriff auf die von der Auswahl betroffenen Medikamentengaben, d.h. alle Elemente mit Label `PA` kann so zusätzlich bezüglich der Zugehörigkeit in einer der beiden Teilmengen $\texttt{PA}_0$ und $\texttt{PA}_1$ erfolgen. Ein solches Vorgehen kann beispielsweise notwendig werden, falls die Dosierungen der ausgewählten Medikamentengaben in den jeweiligen Teilzyklen bzw. Teilmengen $\texttt{PA}_0$ und $\texttt{PA}_1$ unterschiedlich sein sollen.

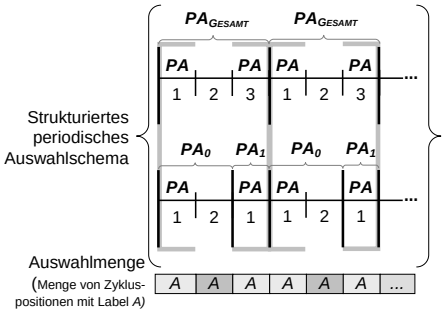


Abbildung 3.27: Beispiel für ein strukturiertes periodisches Auswahlschema

Ausnahmen in Auswahlschemas: Da (periodische) Auswahlschemas erneut einem Periodizitätsschema entsprechen, können diese ebenfalls Ausnahmen beliebiger Art und Weise aufweisen. Ein typisches Beispiel für ein Auswahlschema, das selber Ausnahmen beinhaltet, stellt die Definition von Schaltjahren im gregorianischen Kalender dar. Das Auswahlschema für Schaltjahre, d.h. für die Jahre bzw. Monate (Februar), die um einen Tag verlängert werden, bildet die Angabe "*Jedes 4.Jahr*". Die Ausnahme werden durch die beiden Aussagen "*Jedes 100.Jahr nicht*" und *"Jedes 400.Jahr*" gebildet[3]. In Abbildung 3.28 ist ein Auswahlschema mit Ausnahmen für eine Chemotherapie dargestellt. Im ursprünglichen Auswahlschema werden in jedem Auswahlzyklus des Auswahlschemas der erste und dritte Medikationsteilzyklus ausgewählt. Aufgrund einer Ausnahme wird in "*jedem 2. Auswahlzyklus*" lediglich der erste Medikationsteilzyklus ausgewählt.

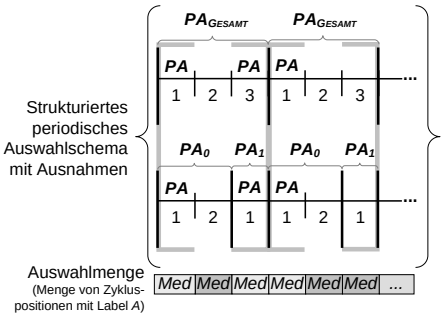


Abbildung 3.28: Beispiel für ein periodisches Auswahlschema mit Ausnahmen

3.4.3.2 Klassen von Auswahloperationen

Wir unterscheiden drei Auswahloperationen: *Zyklusauswahl, Zykluspositionsauswahl* und *Projektionsauswahl*. Abbildung 3.29 zeigt eine Übersicht.

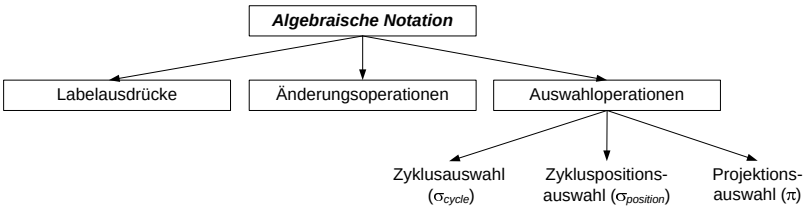


Abbildung 3.29: Übersicht zu Auswahloperationen

[3]Im gregorianischen Kalender sind alle die Jahre deren Jahreszahl durch vier teilbar sind Schaltjahre. Allerdings gibt es zwei Ausnahmen. Ist die Jahreszahl zusätzlich durch 100 teilbar ist das Jahr kein Schaltjahr. Ist die Jahreszahl jedoch zusätzlich auch durch 400 teilbar, dann wiederum ist das Jahr ein Schaltjahr.

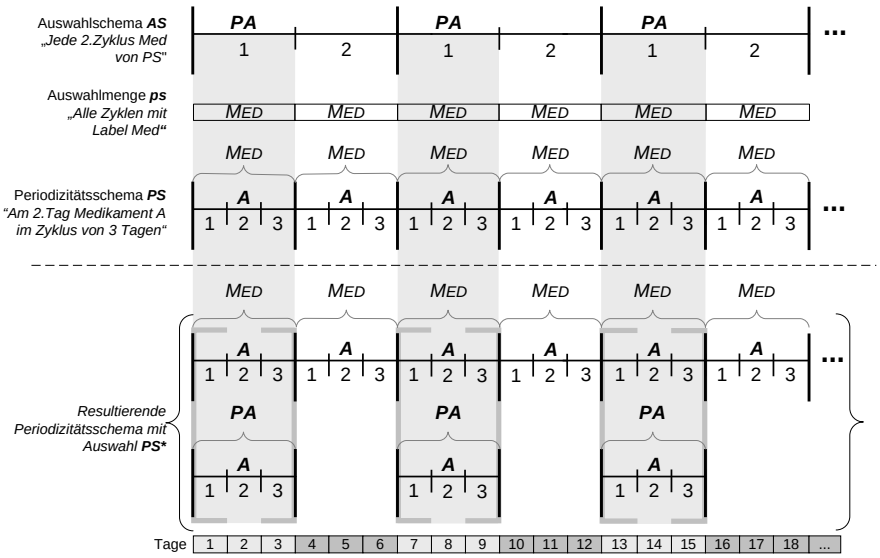


Abbildung 3.30: Zyklusauswahl mit einfachen Auswahlschema

Zyklusauswahl $\sigma_{cycle}(PS, ps, AS)$: Bei dieser Klasse von Auswahloperationen werden auf Basis eines Auswahlschemas bestimmte Zyklen oder Teilzyklen aus einem Periodizitätsschema ausgewählt. In der Auswahlmenge *ps* dürfen nur Strukturen vom Typ Zyklus oder Teilzyklus enthalten sein. Abbildung 3.30 zeigt eine Zyklusauswahl für das Periodizitätsschema *"Am 2. Tag eines Zyklus von drei Tagen erfolgt die Gabe von Medikament A"*. Das Auswahlschema (AS) "*Jeder 2.Zyklus*" ist im oberen Bereich der Abbildung dargestellt und das Periodizitätschema PS mit seinen Zyklen `Med` unmittelbar darunter. Die Auswahlmenge *ps* ist in diesem Beispiel identisch zum Schema PS, d.h. sie enthält alle Zyklen mit Label `Med`. Das Ergebnis der Auswahloperation, d.h. die Kennzeichnung der ausgewählten Zyklen im Periodizitätsschema ist im unteren Bereich dargestellt.

Unabhängig von der konkreten Klasse einer Auswahloperation werden zur Abbildung des Ergebnisses, d.h. zur Abbildung der ausgewählten Strukturen, neue vertikale Strukturierungen – *Auswahlebenen* – eingeführt. Für jedes von der Auswahl betroffene Element der Menge *ps* wird eine neue vertikale Ebene (Auswahlebene) direkt "unterhalb" der ursprünglichen Ebene bzw. seiner *Ausgangsebene* eingefügt. Die horizontale Strukturierung dieser Auswahlebene ist identisch zur Ausgangsebene. Die Labels des Auswahlschemas werden in Abhängigkeit der Klasse der Auswahloperation auf die (Teil-) Zyklen oder Zyklluspositionen der Auswahlebene übertragen. Im vorliegenden Fall wird in dem Beispiel das Auswahllabel `PA` als Label für den Zyklus der neuen Ebene verwendet. Des Weiteren werden die Labels der Zyklluspositionen der Ausgangsebene unverändert auf die Auswahlebene übertragen.

Der Zugriff bzw. die Adressierung der ausgewählten Zyklen erfolgt bei dieser Abbildung über Labelausdrücke. Im obigen Beispiel können mit dem Ausdruck "`Med▼PA`" die

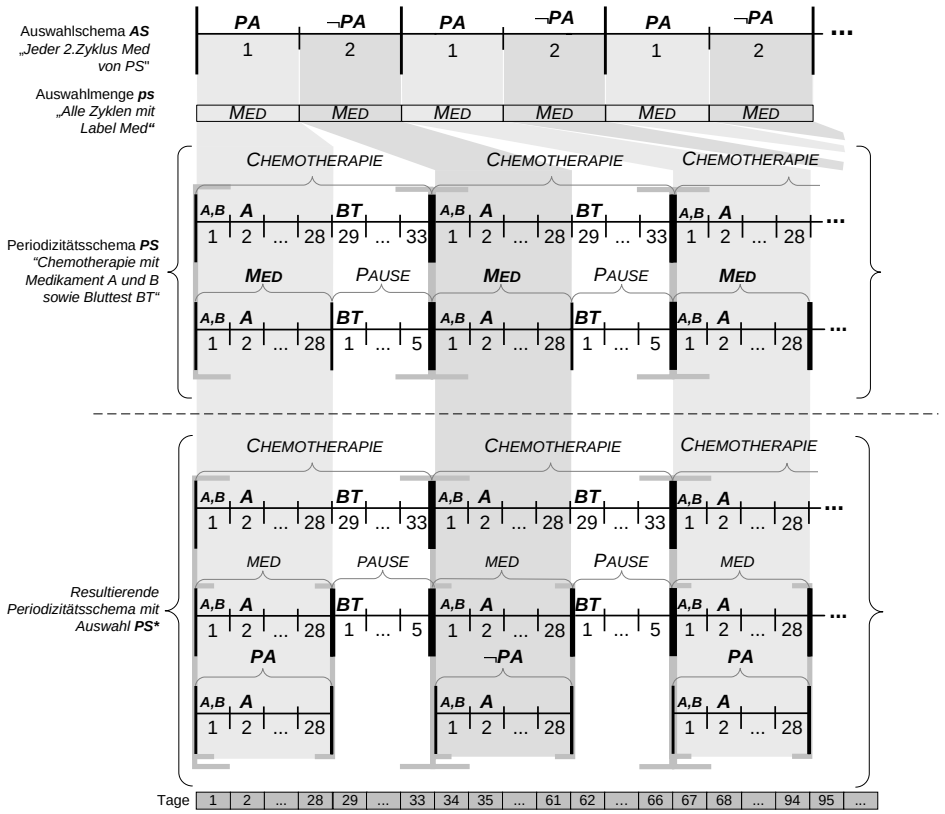


Abbildung 3.31: Zyklusauswahl bei strukturierten Periodizitätsschemas

ausgewählten Zyklen adressiert werden, etwa um diese zu manipulieren. Aufgrund der Verwendung neuer vertikaler Ebenen kann die Auswahl bei Bedarf auch ignoriert werden. Dies bedeutet, dass mit dem Label `Med` unabhängig von den bestehenden tieferen Auswahlebenen weiterhin alle Zyklen mit diesem Label ausgewählt werden können.

Abschließend betrachten wir ein weiteres Beispiel für eine Zyklusauswahl mit einfachem Auswahlschema bei strukturiertem Periodizitätsschema (siehe Abbildung 3.31). Beim verwendeten Periodizitätsschema PS handelt es sich um eine Chemotherapie mit den zwei Teilzyklen Medikation (`Med,` 28 Tage) und `Pause` (5 Tage) sowie Gaben von Medikament `A`, `B` und Bluttest `BT`. Im Beispiel wird lediglich für die Teilzyklen `Med` eine Auswahl vorgenommen. Das heißt, die Auswahlmenge umfasst lediglich die Menge aller Teilzyklen mit Label `Med`. Die Teilzyklen mit Label `Pause` sind nicht von der Auswahl betroffen. Im Ergebnisschema PS^* können mit dem Label `PA` alle von der Auswahl betroffenen Teilzyklen `Med` adressiert werden. Ebenfalls kann mit dem Label `¬PA` auch die komplementäre Menge adressiert werden; d.h. die Teilzyklen `Med`, die nicht von der Auswahl betroffenen sind.

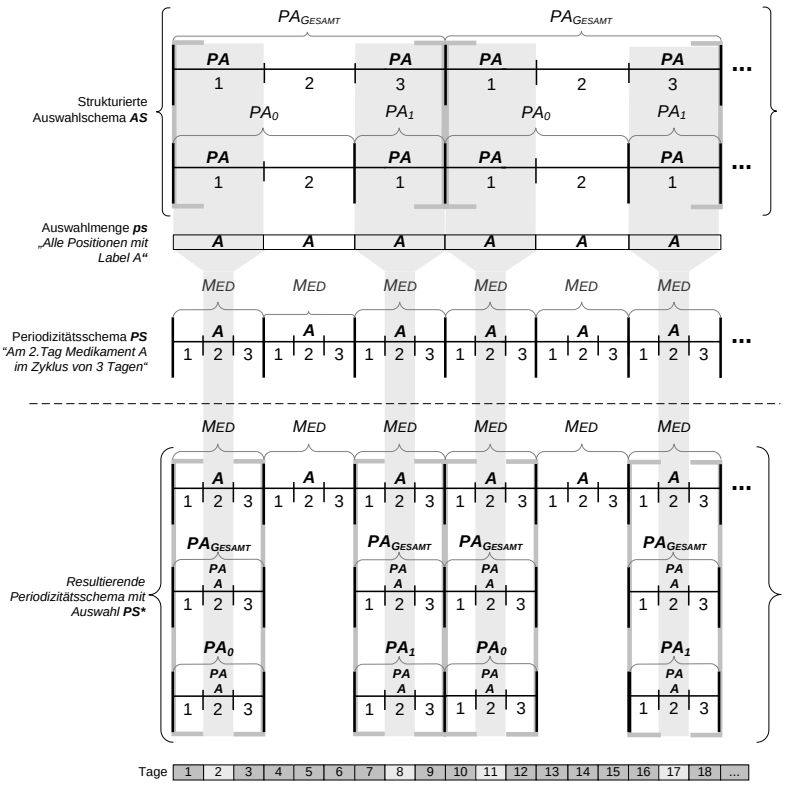


Abbildung 3.32: Zykluspositionsauswahl mit strukturiertem Auswahlschema

Zykluspositionsauswahl $\sigma_{position}(PS, ps, AS)$**:** Die Auswahloperation ermöglicht die Auswahl von Zykluspositionen. In der übergebenen Menge ps dürfen lediglich Strukturen vom Typ Zykluspositionen enthalten sein. In Abbildung 3.32 ist ein Beispiel einer Zykluspositionsauswahl mit Semantik “*Jede 1. und 3.Gabe von Medikament A (beginnend beim ersten)*” dargestellt. Beim verwendeten Auswahlschema handelt es sich um ein strukturiertes Auswahlschema.

Wie erwähnt, wird die Auswahl durch Einführung neuer vertikaler Ebenen für die von der Auswahl betroffenen Strukturen abgebildet. Auf die gleiche Weise wird auch die Strukturierung des Auswahlschemas im Periodizitätsschema abgebildet. Es werden für jede vertikale Ebene des Auswahlschemas eine neue Auswahlebene im Periodizitätsschema eingeführt und die Labels vom Auswahlschema entsprechend übertragen. Nach erfolgter Auswahloperation steht im Periodizitätsschema die ursprüngliche Struktur des Auswahlschemas zur Verfügung, zum Beispiel, dass navigierende Verkettungen bezüglich der Struktur des Auswahlschema, wie “$\texttt{Med}\blacktriangledown\texttt{PA}_{\texttt{Gesamt}}\blacktriangledown\texttt{PA}_0 \bigtriangledown \texttt{PA}$”, zur Adressierung bestimmter Zyklen bzw. Zykluspositionen verwendet werden können. Allerdings ist die dargestellte Form der Übertragung der Struktur des Auswahlschemas auf das Periodizitätsschema nicht vollständig. Einige Strukturinformationen des Auswahlschemas wer-

den bei der dargestellten Übertragung nicht in das Periodizitätsschema übernommen. Insbesondere betrifft dies die absoluten Positionsnummern von Teilzyklen des Auswahlschemas. So kann der zweite Teilzyklus $\texttt{PA}_1$ im Zyklus $\texttt{PA}_{\texttt{Gesamt}}$ des Auswahlschemas im Periodizitätsschema nicht mit dem Labelausdruck "$\texttt{PA}_{\texttt{Gesamt}}\overset{2}{\blacktriangledown}*$" adressiert werden. Diese Information über seine absolute Position im Auswahlschema liegt nach Übertragung der Struktur im Periodizitätsschema nicht mehr vor. Diese Einschränkung gilt sowohl für Zyklus- und Zykluspositionsauswahl als auch für die noch folgende Projektionsauswahl. Um entsprechende Informationen ebenfalls im Periodizitätsschema zugreifbar zu machen, werden neue Strukturen im zyklenorientierten Zeitmodell benötigt. In der vorliegenden Arbeit wird aus Gründen der Verständlichkeit des zyklus-orientierten Zeitmodells auf die Einführung solcher Strukturen verzichtet. Allerdings werden wir in Kapitel 4 zeigen, dass die kompakte Repräsentation des Zeitmodells – der HSP-Baum – diese Einschränkung des allgemeinen Modells nicht aufweist. Im HSP-Baum stehen die Strukturinformationen des Auswahlschemas uneingeschränkt nach der Überführung ins Periodizitätsschema zur Verfügung.

Projektionsauswahl $\pi(PS, p_1, PS_2, p_2)$ In den beiden vorherigen Auswahlklassen standen einzelne Periodizitätsschema im Vordergrund. Bei der Projektionauswahl hingegen werden bei einer Auswahl unterschiedliche Periodizitätsschemas (mit beliebiger Komplexität) zueinander in Beziehung gesetzt. Ein bekanntes Beispiel für eine Projektionsauswahl stellt die Beschreibung der Umstellung von Winter- auf Sommerzeit dar. In Deutschland erfolgt die Umstellung am letzten Sonntag im März (Winter- auf Sommerzeit) bzw. am letzten Sonntag im Oktober (Sommer- auf Winterzeit). In der Beschreibung werden die Granularität *Woche* und die Granularität *Monat* aufeinander abgebildet bzw. projiziert. Das heißt, nur die Sonntage (von allen möglichen Sonntagen), die gleichzeitig den letzten Sonntag im Monat März bzw. Oktober bilden, werden ausgewählt und sind von der Änderung der Tageslänge betroffen. Im Allgemeinen kann eine Projektion als eine "UND-Verknüpfung" von Zyklusposition zweier unabhängiger Periodizitätsschemas verstanden werden. Nur die Zyklusposition bzw. Labels eines Periodizitätsschemas, die gleichzeitig in einem anderen Periodizitätsschema mit bestimmten Positionen bzw. Labels korrespondieren, werden bei einer Projektionauswahl entsprechend markiert. Die Auswahl, welche Zyklusposition aufeinander projeziert werden sollen, erfolgt durch den Anwender.

Vergleichbar zu den beiden vorherigen Auswahloperationen, wird auch bei einer Projektionsauswahl ein "Auswahlschema" über einer Auswahlmenge interpretiert. Im Gegensatz zu diesen Auswahloperationen handelt es sich jedoch bei den Auswahlschema PS_2 um ein "echtes" Periodizitätsschema, das selber im Wertebereich "verankert" ist und einen Instanziierungspunkt aufweisen kann. Bei einer Projektionsauswahl erfolgt die Interpretation des Periodizitätsschemas PS_2 nicht über dem Wertebereich, sondern den Zyklusposition des Periodizitätsschema PS. Die Auswahlmenge, d.h. die Menge über der das Periodizitätsschema PS_2 interpretiert wird, entspricht immer der vollständigen Menge an Zyklusposition im Schema PS und wird nicht explizit übergeben. Die

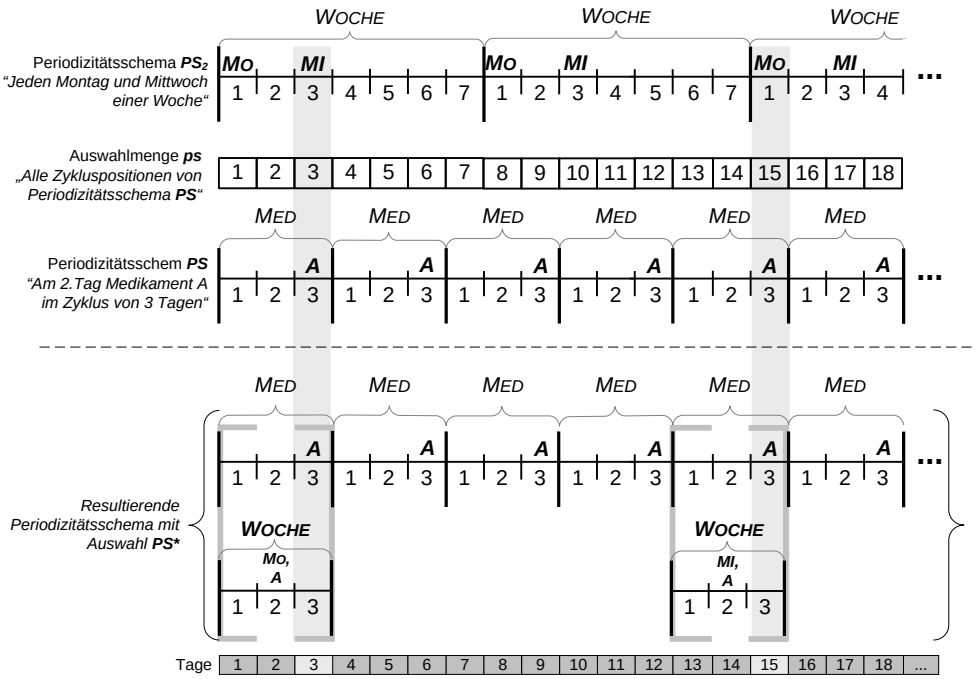


Abbildung 3.33: Beispiel für eine Projektionsauswahl

Projektionsmenge von PS beschreibt diejenige Menge an Zyklusspositionen, auf welche die Zyklusspositionen des Schemas PS_2 projeziert werden sollen. Der Einfachheithalber wird auf die Einführung einer Projektionsmenge für Schema PS_2 verzichtet. Stattdessen werden stets alle Zyklusspositionen von PS_2 bzw. deren Labels auf die Zyklusspositionen der Projektionsmenge von Schema PS projiziert. Die Projektionsmenge korrespondiert in seiner Semantik mit der Auswahlmenge bei den beiden anderen Auswahloperationen, d.h. sie enthält die Zyklusspositionen von Schema PS, die von der Auswahl bzw. Projektion betroffen sind. Die Projektionsmenge von PS wird mit dem Parameter p_1 übergeben, d.h. alle Zyklusspositionen von PS, die bei der Projektion berücksichtigt werden sollen. Mit Parameter p_2 wird der Instanziierungspunkt des Periodizitätsschemas PS_2 hinterlegt, da die Projektionsauswahl von diesem Instanziierungspunkt abhängig ist.

Abbildung 3.33 zeigt eine Projektionsauswahl zwischen dem Schema "*Jeden Montag und Mittwoch in einer Woche*" (PS_2) und einem Periodizitätsschema (PS) mit einem Zyklus der Länge 3 und Aktivität `A` am zweiten Tag. Im Beispiel werden alle Aktivitäten `A` im Schema PS ausgewählt, die gleichzeitig auch auf einen "Montag" (`Mo`) oder "Mittwoch" (`Mi`) im Schema PS_2 fallen. Die Projektionsmenge bzw. Parameter p_1 enthalten alle Positionen mit Label `A` von Schema PS. Das Ergebnis der Auswahl stellen die grau hinterlegten Bereiche dar. Für beide Periodizitätsschemas wird mit 1 der gleiche Instanziierungspunkt angenommen. Die Abbildung des Ergebnisses der Projektionsauswahl im

Periodizitätsschema PS erfolgt identisch zu den anderen beiden Auswahloperationen. Die Struktur des Periodizitätsschemas PS_2 wird durch neue vertikale Ebenen im Periodizitätsschema PS abgebildet und die entsprechenden Labels von PS_2 auf diese neuen Auswahlebenen übertragen. So kann nach erfolgter Auswahl etwa mit dem Labelausdruck "$\texttt{Med}\blacktriangledown\texttt{Woche}\triangledown\texttt{A} \overset{0}{\triangleright} \texttt{Mo}$" auf die Zykluspositionen zugegriffen werden, die sowohl eine Gabe von Medikament `A` aufweisen und auf einen Montag (`Mo`) fallen.

In Rückblick auf die in Kapitel 2 diskutierte Problemstellung von Abhängigkeiten zwischen periodischen Spezifikationen bzw. die daraus resultierenden Anforderungen hinsichtlich Vergleichbarkeit periodischer Spezifikationen erweist sich bei genauerer Betrachtung die Projektionsauswahl als eine Option diesem Problem zu begegnen. So ermöglicht es eine Projektionsauswahl bei der Konstruktion von Periodizitätsschemas a priori, bekannte Abhängigkeiten mit anderen Periodizitätsschema bzw. Spezifikationen zu berücksichtigen. Beispielsweise kann bei der Konstruktion des obigen Periodizitätsschemas berücksichtigt werden, dass die Gaben von Medikament `A` nicht auf einen Montag fallen bzw. die entsprechenden Gaben auf den Dienstag verschoben werden. Beispielsweise wird mit den Operationsausdruck $MoveLabel(PS^*, "\texttt{Med}\blacktriangledown\texttt{Woche}\triangledown\texttt{A} \overset{0}{\triangleright} \texttt{Mo}", +1)$ jede Gabe von Medikament `B`, die auf einen Montag fällt, um einen Tag (nach rechts) auf Dienstag verschoben. Allerdings ist eine Projektionsauswahl nicht dazu geeignet, den Anforderungen einer effizienten (nachträgliche) Vergleichbarkeit zwischen Periodizitätsschemas bzw. Spezifikationen zu begegnen. Hierfür werden andere Verfahren und Vorgehensweisen benötigt, wie sie in Kapitel 6 vorgestellt werden.

3.4.3.3 Auswahlerhaltende und nicht-auswahlerhaltende Operationen

Bei der Anwendung einer Auswahloperationen, werden aus zum Zeitpunkt der Anwendung vorliegende Periodizitätsschemas auf Basis des Auswahlschemas (Teil-) Zyklen oder Zykluspositionen ausgewählt. Beispielsweise werden bei einer Zykluspositionsauswahl $\sigma_{position}(PS, ps, AS)$ alle Positionen der Auswahlmenge ps auf Basis des Auswahlschemas AS ausgewählt. Wir nehmen an, dass das Beispiel die Semantik aufweist, dass "*alle Zykluspositionen mit Label A*" einer entsprechenden Auswahl unterzogen werden. Das heißt, die Auswahlmenge ps enthält alle Zykluspositionen von PS mit Label A und wurde durch einen entsprechende Labelausdruck gebildet.

Grundsätzlich ist es möglich, dass nach einer solchen Auswahloperation mit der Operation $InsertLabel$ für eine Zykluspositionen (die zuvor nicht in ps enthalten war) nachträglich das Label A hinzugefügt wird. Hierdurch wird jedoch die Semantik der bestehenden Auswahl "*Alle Zykluspositionen mit Label A*" nachträglich beeinflusst. Das heißt, einerseit weist die Auswahl die Semantik auf "*Alle Positionen mit Label A"* und andererseits wird nachträglich ein Position mit Label A hinzugefügt (die erstmal nicht in der Auswahl enthalten ist). Es ergibt sich die Fragestellung wie mit dieser neuen Zyklusposition mit Label A in Bezug auf die bestehende Auswahl "*alle Positionen mit Label A*"

umgegangen wird (siehe später). Operationen die nachträglich zu einer (direkten oder indirekten) Beeinflussung einer zuvor bestehenden Auswahl führen, werden als **nicht-auswahlerhaltende** Operation bezeichnet. Folglich werden alle anderen Operation als **auswahlerhaltende** Operationen bezeichnet. Im obigen Beispiel wäre dies etwa das Hinzufügen des Labels B (falls keine Auswahl für das Label B besteht).

Für den Umgang mit nicht-auswahlerhaltenden Operationen können generell zwei Strategien unterschieden werden:

1. *Zeitabhängige Auswahl*: Eine Auswahloperation bezieht sich stets nur auf die Situation wie sich diese zum Zeitpunkt der Anwendung der Operation vorlag. Nachträgliche nicht-auswahlerhaltende Operationen haben keine Auswirkungen auf bestehende Auswahlen. Für das obige Beispiel bedeutet dies, dass das Lalbel A einfach zu der Zyklusposition hinzugefügt wird, ohne diese Position in die bestehende Auswahl aufzunehmen.
2. *Zeitunabhängige Auswahl*: Eine Auswahloperation bezieht sich unabhängig vom Zeitpunkt der Anwendung stets auf die Gesamtsituation. Für das obige Beispiel bedeutet dies, dass das Label A zu der Zyklusposition hinzugefügt wird und die Position zusätzlich in die Auswahl mit aufgenommen wird.

Im zyklenorientierten Modell werden wir für den Umgang mit nicht-auswahlerhalten Operationen die erste Strategie, d.h. zeitabhängige Auswahl, verwenden. Bei dieser Strategien müssen nicht-auswahlerhaltende Operationen nicht explizit berücksichtigt (identifziert) werden, da die Auswahl hierdurch nicht beeinflusst bzw. manipuliert wird. Insbesondere die Identifikation nicht-auswahlerhaltenden Operationen, wie in der zweiten Strategie notwendig, erweist sich als problematisch. Die Identifikation ist abhängig von der verwendeten Operation, den verwendeten Parametern und den bereits bestehenden Auswahlen. Darüber hinaus, kann durch verdrehen der Reihenfolge, d.h. Anwendung der nicht-auswahlerhaltend Operation vor der entsprechenden Auswahloperation (mit zeitabhängiger Auswahl) analoge Ergebnisse erzielt werden.

3.4.4 Beispiele

Im Folgenden wird die algebraischen Notationsform beispielhaft für eine Chemotherapie dargestellt. Es handelt sich um eine Chemotherapie mit Medikationzyklus der Länge 28 Tage und nachfolgenden Pausenzyklus der Länge 5 Tage. Ausgangspunkt der Beschreibung bildet das Periodizitätsschema P_1 bestehend aus strukturierten Zyklen mit Label `Chemotherapie` sowie zwei Teilzyklen `Medikation` und `Pause` jeweils mit Länge 1. In den Schritten 1 und 2 werden die Zykluslängen angepasst.

Anpassen der Zykluslängen:
Schritt 1: $P_2 = IncreaseLength(P_1, \text{"}\texttt{Chemotherapie▼Medikation}\text{"}, +27)$
Schritt 2: $P_3 = IncreaseLength(P_2, \text{"}\texttt{Chemotherapie▼Pause}\text{"}, +4)$

Innerhalb des Medikationsteilzyklus finden vom 1.-14. Tag die Gaben von Medikament A (`Med_A`) und am 1. und 8. Tag die Gaben von Medikament B (`Med_B`) statt. Hierfür

müssen die jeweiligen Labels an den korrekten Positionen eingefügt werden (vgl. Schritte 3 und 4).

Einfügen von Labels für die Medikationen:

Schritt 3: $P_4 = InsertLabel(P_3, \text{"}\texttt{Chemotherapie▼Medikation}\overset{1-14}{\triangledown} * \text{"}, \texttt{Med_A})$

Schritt 4: $P_5 = InsertLabel(P_4, \text{"}\texttt{Chemotherapie▼Medikation}\overset{1,8}{\triangledown} * \text{"}, \texttt{Med_B})$

Während der Chemotherapie erfolgen in jeder zweiten Pause (beginnend bei der Zweiten) am letzten Tag ein zusätzlicher Bluttest (`BT`). Um diese (periodische) Ausnahme abzubilden, wird zunächst das Auswahlschema definiert. Ausgangspunkt bildet ein Periodizitätsschema AS mit Zyklen der Länge eins.

Auswahlschema beschreiben:

Schritt 5.1: $AS_2 = IncreaseLength(AS, " * ", +1)$ (Anpassen der Zykluslänge)

Schritt 5.2: $AS_3 = InsertLabel(AS_2, " * \overset{2}{\triangledown} * ", \texttt{PA})$ (Einfügen von Labels)

Nach Definition des Auswahlschemas wird im nächsten Schritt eine Zyklusauswahl vorgenommen und für die entsprechenden Pausezyklen der Bluttest am letzten Tag eingefügt (vgl. Schritt 6).

Periodische Zyklusauswahl vornehmen:

Schritt 6: $P_6 = \sigma_{cycle}(P_5, \text{"}\texttt{Chemotherapie▼Pause}, AS_3)$

Einfügen von Labels für den Bluttest:

Schritt 7: $P_7 = InsertLabel(P_6, \text{"}\texttt{Chemotherapie▼Pause▼PA}\overset{last}{\triangledown} * \text{"}, \texttt{BT})$

Auf die Medikamentengabe von Medikament `A` kann dann mit dem Labelausdruck "`Chemotherapie▽Med_A`" zugegriffen werden oder auf die Bluttests mit dem Ausdruck "`Chemotherapie▼Pause▽BT`".

Für das Periodizitätsschema P_7 gilt zusätzlich der Zusammenhang, dass alle Gaben von Medikament `B,` die auf einen Sonntag fallen, um einen Tag nach hinten (auf Montag) verschoben werden sollen. Hierfür wird das Periodizitätschema für "*Woche*" benötigt, das eine Woche auf Basis von Tagen beschreibt. Ausgangspunkt bildet das Periodizitätsschema G_1 bestehend aus Zyklen `Wochen` mit Zykluslänge = 1.

Periodizitätsschema G_{Woche} beschreiben:

Schritt 8.1: $G_2 = IncreaseLength(G_1, " * ", +6)$

Schritt 8.2: $G_{Woche} = InsertLabel(...InsertLabel(G_2, " * \overset{1}{\triangledown} * ", \texttt{Mo})...), " * \overset{7}{\triangledown} * ", \texttt{So})$

Im nächsten Schritt wird eine Projektionsauswahl "aller Sonntage" ("`Woche▽So`") auf die Gaben von Medikament `B` in Schema P_7 benötigt, um alle von der Verschiebung betroffenen Medikamentengaben zu kennzeichnen. Für das Schema G_{Woche} wird der Instanziierungspunkt 0 angenommen.

Projektionsauswahl vornehmen:

Schritt 9:$P_8 = \pi(P_7, \text{"}\texttt{Chemotherapie▼Medikation▽Med_B}\text{"}, G_{Woche}, \text{"}\texttt{Woche▽So}\text{"}, 0)$

Mit dem Labelausdruck "Chemotherapie▼Medikation▼Woche▽A $\overset{0}{\triangleright}$ So" kann nach erfolgte Projektionsauswahl auf alle Gaben von Medikament B zugegriffen werden, die auf einen Sonnatg fallen. Diese Medikamentengaben bzw. die entsprechenden Labels in P_8 müssen dann lediglich um einen Tag nach rechts auf den Montag verschoben werden.

Labels verschieben:

Schritt 10: $MoveLabel(P_8, \text{"Chemotherapie▼Medikation▼Woche▽A } \overset{0}{\triangleright} \text{ So"}, +1)$

3.5 Zusammenfassung

In diesem Kapitel haben wir Anforderungen an eine umfassende logische Beschreibung von praxisrelevanten Periodizitätsschemas diskutiert. Es zeigt sich, dass Periodizitätsschemas inklusive ihrer Ausnahmen auf semantisch hoher Ebene beschrieben werden müssen. Insbesondere gilt es, entsprechende Beschreibungen für Anwender sowohl intuitiv verständlich und nachvollziehbar zu gestalten, als auch, zumindest für Fachanwender, leicht anpassbar. Eine der wesentlichsten Anforderungen bildet hier die umfassende und generische Unterstützung von Ausnahmen in Periodizitätsschemas. Die in der Literatur vorherrschende implizite Spezifikationslogik erweist sich als keine geeignete Möglichkeit für eine generische Unterstützung von Ausnahmen. Ausdruckstärker und somit wünschenswerter ist eine explizite Spezifikationslogik. Diese Ergebnisse führten zu der Fragestellung, inwieweit vorliegende Ansätze den vorgestellten Anforderungen begegnen können. Um diese Frage zu beantworten, haben wir verschiedene Zeitmodelle aus der Literatur näher untersucht. Es zeigt sich jedoch, dass lediglich ein einziges Zeitmodell generell eine explizite Spezifikationslogik bei Ausnahmen gewährleisten könnte. Allerdings handelt es sich um ein Modell mit Low-Level Charakter, so dass den Anforderungen nach Verständlichkeit und Strukturierbarkeit nicht ausreichend begegnet werden konnten.

Um den Anforderungen einer umfassenden Beschreibung von Periodizitätsschemas begegnen zu können, werden ein neues Zeitmodell und Beschreibungsformalismus eingeführt. Bei dem vorgestellten zyklenorientierten Zeitmodell handelt es sich um ein High-Level-Modell, das der Anforderung nach Verständlichkeit und Strukturierbarkeit adäquat begegnen kann. So werden semantisch hohe Strukturen wie Zyklen und Zyklenpositionen unmittelbar durch das Zeitmodell unterstützt und müssen nicht durch den Benutzer manuell "nachgebaut" werden. Des Weiteren ermöglicht das Modell eine adäquate Strukturierung sowie einen gesamtheitlichen Umgang von komplexen Periodizitätsschemas mit ihrer oftmals großen Vielzahl an Einzelaktivitäten. Die Trennung von Struktur (vorgegeben durch die Menge an Zyklen, Teilzyklen und Zyklenpositionen) und Semantik (definiert über die Labels) im Zeitmodell, bildet die Grundlage der Unterstützung einer expliziten Spezifikationslogik. Hierdurch wird die wesentlichste Anforderung der umfassenden Unterstützung von Ausnahmen auf semantisch hoher Ebene hervorragend durch das Zeitmodell erfüllt. Neben einfachen Ausnahmen können auch komplexe nichtäquidistante Ausnahmen beliebiger Komplexität beschrieben werden. Im Gegensatz zu den bestehenden Ansätzen in der Literatur (siehe Abschnitt 3.2), erfüllt

das zyklenorientierte Zeitmodell die Anforderungen an eine umfassende und generische Unterstützung von Ausnahmen in Periodizitätsschemas.

Für die Anforderung Verständlichkeit können wir zeigen, dass die Grundoperationen der algebraischen Notationsform die Grundlage für eine intuitiv verständliche und nachvollziehbare Beschreibung von Periodizitätschemas ermöglichen. Die Konstruktionsvorschrift des Periodizitätsschemas liegt explizit und vollständig in Händen den Benutzers, was zu einer transparenten und leicht anpassbaren Spezifikation von Periodizitätsschemas führt. Dies fördert nicht nur die Benutzerfreundlichkeit sondern reduziert in gleichem Maße auch die Fehleranfälligkeit entsprechend beschriebener Periodizitätsschemas. Die vorgestellten Ausdrücke und Operationen bilden eine adäquate Grundmenge zur Beschreibung praxisrelevanter Periodizitätsschemas, die keinen Anspruch auf Vollständigkeit erhebt. Jedoch handelt es sich um gut erweiterbare Konzepte, die leicht um weitere Labelverkettungen oder Beschreibungsoperationen ergänzt werden können.

Zusammenfassend betrachtet standen in diesem Kapitel die Anforderungen nach Verständlichkeit, Strukturierbarkeit und Unterstützung von Ausnahmen im Vordergrund. Die nachfolgenden Kapitel 4 und 5 zeigen, wie aufbauend auf dem zyklenorientierten Zeitmodell, den weiteren bestehenden Anforderungen nach Auswertbarkeit, Korrektheit und Granularitätsfähigkeit begegnet werden kann.

4 Der HSP-Baum

Das zyklenorientierte Zeitmodell ermöglicht eine umfassende und anwenderorientierte Beschreibung komplexer Periodizitätsschemas. Dieses Zeitmodell basiert in seiner allgemeinen Theorie jedoch auf der Verwendung von (theoretisch) unendlichen Wertemengen. Für effizient berechenbare Algorithmen, etwa in Hinblick auf die Anforderungen nach Auswertbarkeit oder Korrektheit, ist die direkte Verwendung des logischen Zeitmodells für entsprechende Auswertungen nicht angebracht. Man benötigt vielmehr eine kompakte und berechenbare Repräsentationsform des zyklenorientierten Modells. Eine solche Repräsentation wird in diesem Kapitel mit dem HSP-Baum vorgestellt. Neben der Berechenbarkeit dient der HSP-Baum auch als Grundlage für eine präzise formale Beschreibung des zyklenorientierten Modells und der vorgestellten algebraischen Notationsform. Eine präzise formale Semantik und ein Regelwerk sind unerlässlich, wenn die Handhabung und Implementierung entsprechender Periodizitätsschemas effizient sein soll.

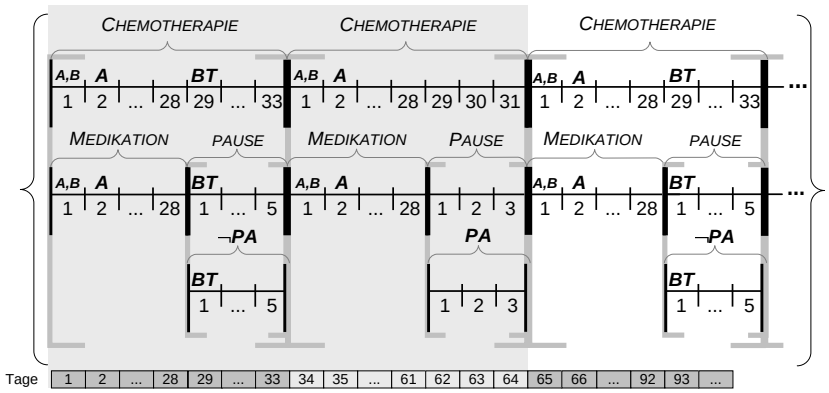
Abschnitt 4.1, stellt zunächst die Grundidee des HSP-Baums vor. Im Anschluss daran werden in Abschnitt 4.2 der HSP-Baum formal definiert und seine Elemente vorgestellt. Hierbei werden Strukturregeln definiert, die für die Korrektheit des HSP-Baumes unerlässlich sind. Die Realisierung der algebraischen Notationsform bzw. seiner Beschreibungsoperationen erfolgt in Abschnitt 4.3. Abschließend werden im Abschnitt 4.5 ein Ausblick gegeben und in Abschnitt 4.6 die Ergebnisse des Kapitels zusammengefasst.

4.1 Grundidee des HSP-Baums

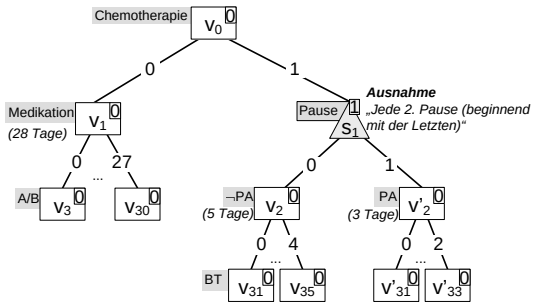
Im allgemeinen Modell wird jeder (strukturierte) Zyklus eines Periodizitätsschemas einzeln in seiner Struktur und Labelbelegung beschrieben. Dies führt u.a. dazu, dass ein Periodizitätsschema einer (theoretisch) unendlichen Menge solcher Strukturen entspricht. Die Mehrzahl dieser (strukturierten) Zyklen weisen jedoch gleiche Strukturen und Belegungen auf. Unterschiedliche Zyklen treten lediglich im Zusammenhang mit periodischen Ausnahmen auf (siehe Abschnitt 3.4.3). Dies ermöglicht eine kompaktere Repräsentationsform, die die Menge an Zyklen bzw. die aufgrund von Ausnahmen abweichenden Zyklen in endlicher Form repräsentiert. Eine solche Repräsentationsform bildet der im Rahmen dieser Arbeit entwickelte HSP-Baum. Seine Grundidee entspricht dem Ansatz zur Bildung von Wiederholungsmustern. Anstelle die vollständige Menge an identischen Zyklen zu repräsentieren, wird lediglich das Wiederholungsmuster der Zyklenreihenfolgen verwendet. Da in der vorliegenden Arbeit ausschließlich periodische Ausnahmen angenommen werden, ist die Bildung von Wiederholungsmustern für die Zyklenreihenfolgen auch immer möglich (siehe Abschnitt 3.4.3.1).

Ein HSP-Baum entspricht einer semantisch reichhaltigen Repräsentation eines Wiederholungsmusters. Hierbei werden neben Zyklusreihenfolgen auch bestehende Strukturierungen von Zyklen sowie Ausnahmen in kompakter und endlicher Form dargestellt. Ein "Durchlaufen" bzw. Traversieren eines (finiten) HSP-Baumes – in beliebiger Häufigkeit – generiert, das entsprechende (infinite) Periodizitätsschema des zyklenorientierten Modells. Ein solcher Traversierungsalgorithmus ermöglicht es darüber hinaus, die für die operationale Unterstützung benötigten Wiederholungsmuster (z.B. Bitvektoren) abzuleiten. Die Grundidee des HSP-Baums ist in Abbildung 4.1 anhand eines Beispiels dargestellt.

In Abbildung 4.1a ist ein Periodizitätsschema im zyklenorientierten Modell dargestellt. Es handelt sich um eine Chemotherapie, bestehend aus den beiden Teilzyklen `Medikation` (28 Tage) und `Pause`. Das Beispiel enthält eine Ausnahme, so dass "*Jede 2.Pause*" (Label =`PA`) anstatt fünf Tage lediglich drei Tage lang ist und kein Bluttest (`BT`)



(a) Darstellung im zyklenorientierten Modell



(b) Darstellung im HSP-Baum

Abbildung 4.1: Grundidee der Darstellung von Periodizitätsschemas durch HSP-Bäume

vorgenommen wird. Das Wiederholungsmuster der Zyklusreihenfolgen stellt der grau hinterlegte Bereich im Periodizitätsschema dar. Das heißt, die (theoretisch) unendliche Menge an Zyklen des Periodizitätsschemas kann basierend auf der endlichen Menge an Zyklen innerhalb des grauen Bereichs vollständig beschrieben werden. Durch einen HSP-Baum wird lediglich diese endliche Teilmenge an Zyklen und Reihenfolgen beschrieben und repräsentiert.

In Abbildung 4.1b ist der entsprechende HSP-Baum des Beispiels dargestellt. Die rechteckig dargestellten Knoten (H-Knoten) dienen der Abbildung der Strukturierung von Periodizitätsschemas. Im dargestellten Beispiel etwa besteht die Chemotherapie aus zwei aufeinander folgenden Teilzyklen `Medikation` und `Pause` (`PA` und ¬`PA`). Der dreieckig dargestellte Knoten s_1 (S-Knoten) beschreibt die periodische Ausnahmeregel („*Jede 2.Pause*“) und die nachfolgenden Knoten die entsprechenden Pausen. Das heißt, der Knoten s_1 repräsentiert die gesamte Menge an Pausen, wohingegen Knoten v_2' die von der Ausnahme betroffenen Pausen und Knoten v_2 die nicht von der Ausnahme betroffenen Pausen darstellt. Die Blätter des Baumes entsprechen den Zyklospositionen und die enthaltenen Labels (`A`/`B`) treffen Aussagen zu den entsprechenden Medikamentengaben. So besagt etwa das erste Blatt (Knoten v_3), dass am 1. Tag in der Medikation sowohl Medikament A (Label `A`) als auch Medikament B (Label `B`) verabreicht werden.

Die Grundidee des HSP-Baums weist auf den ersten Blick eine Verwandschaft bzw. eine Analogie zu formalen Sprachen bzw. Grammatiken mit entsprechenden Produktionsregeln auf [77]. So könnte ein HSP-Baum als ein Ableitungs- bzw. Parsebaum einer entsprechenden Grammatik verstanden werden. Bei entsprechender Traversierung des Parsebaums erhält man dann ein Wort, der durch die Grammatik festgelegten Sprache. Anders ausgedrückt, die bei der Traversierung des HSP-Baums resultierenden Wiederholungsmuster könnten als Wörter einer entsprechenden Sprache verstanden werden. Eine solche Betrachtungsweise würde es u.a. ermöglichen, dass wir für ein gegebenes Wiederholungsmuster (Wort) überpüfen könnten, ob es durch einen gegeben HSP-Baum (Sprache) erzeugt wird (bzw. enthalten ist). Im Folgenden werden wir diese Verwandschaft zu formalen Sprachen jedoch nicht weiter vertiefen bzw. berücksichtigen. So steht zum einen die Überprüfung ob ein Wiederholungsmuster durch einen bestimmten HSP-Baum erzeugt wurde (Wortproblem) nicht im Vordergrund der Arbeit. Zum Anderen dient der HSP-Baum selber als kompakte und berechenbare Repräsentationsform des zyklenorientierten Modells. Das heißt, die Festlegung der formalen Semantik und der konkreten Realisierung der algebraischen Notationsform des zyklenorientierten Zeitmodells erfolgt ausschließlich auf dem HSP-Baum und seinen Strukturen (und nicht auf einer entsprechenden Grammatik).

4.2 Definition des HSP-Baums

Die Grundlage für die Definition des HSP-Baums bilden sogenannte geordnete Bäume. Ein geordneter Baum ist ein Wurzelbaum, in dem für die Nachfolger eines jedes Knotens eine Ordnungsrelation definiert ist. Anschaulich kann die Ordnung als Nummerierung der Kanten zu den Nachfolgern verstanden werden. Die Nummerierung bzw. Ordnung gibt etwa an, in welcher Reihenfolge die Knoten bei unterschiedlichen Traversierungsverfahren (preorder, inorder, postorder) durchlaufen werden [68]. Ein HSP-Baum ist ein geordneter Baum, der um entsprechende Aspekte zur Abbildung des zyklenorientierten Zeitmodells erweitert wurde. Im Folgenden wird – wie im vorherigen Kapitel auch – von einer einzelnen Granularität ausgegangen. Das heißt, ein einzelner HSP-Baum entspricht einer einzelnen Granularitätsebene (z.B. Tage oder Stunden). Die Handhabung mehrerer Granularitätsebenen sowie die Berücksichtigung der Anforderung nach Granularitätsfähigkeit sind Gegenstand von Abschnitt 4.5.

Für die formale Repräsentation eines HSP-Baumes verwenden wir eine mengenbasierte Darstellung, d.h. ein HSP-Baum wird auf eine (finite) Menge von Knoten und Kanten unterschiedlichen Typs abgebildet. Formal ist ein HSP-Baum wie folgt definiert:

Definition 4.1 (HSP-Baum): *Ein HSP-Baum ist das Tupel*

$$T = (N, E, NT, ND, NL, SP)$$

mit Knotenmenge N und Kantenmenge E sowie den Abbildungen:

$NT : N \mapsto NodeTypes$	(Knotentyp)
$ND : N \mapsto \mathbb{N}$	(Schachtelungstiefe)
$NL : N \mapsto 2^{StructureSet \cup ActivitySet}$	(Labels)
$SP : N \mapsto \mathbb{Z} \times \mathbb{N} \times \mathbb{N} \times 2^{\mathbb{N}}$	(Auswahl- oder Projektionsparameter)

Die einzelnen Elemente des HSP-Baumes haben dabei die folgende Bedeutung:

- Die Knotenmenge N und die Kantenmenge $E \subseteq N \times N \times \mathbb{N}$ bilden einen geordneten Baum. Der zusätzliche Wert ($\mathbb{N}$) repräsentiert die Ordnungsrelation $(<, \mathbb{N})$ auf den Kanten des geordneten Baums, d.h. die Ordnungsrelation führt zu einer eindeutigen, linearen Nummerierung der Kanten für die Nachfolger eines jedes Knotens. In Abhängigkeit vom Knotentyp ($NodeTypes$, siehe unten) stellt ein einzelner Knoten einen konkreten zeitlichen Wert bzw. zeitlichen Intervall dar, oder er gehört zu einem Auswahlschema einer periodischen Ausnahme.
- $NT : N \mapsto NodeTypes$ ist eine Funktion, die jedem Knoten einen Typ aus der Menge $NodeTypes$ zuordnet. Die Menge $NodeTypes$ ist wie folgt definiert:

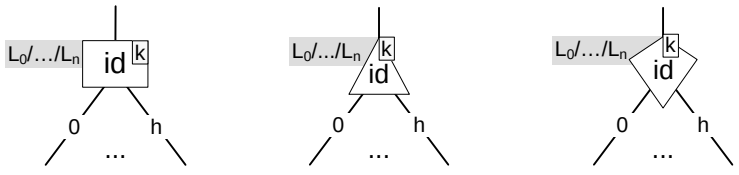
  $$NodeTypes := \{\texttt{HNODE}, \texttt{SNODE}, \texttt{PNODE}, \texttt{0NODE}\}$$

  Die Semantik der Knotentypen wird in den folgenden Abschnitten erläutert.

- $ND : N \mapsto \mathbb{N}$ ist eine Funktion, die jedem Knoten eine Schachtelungstiefe (nesting depth) zuordnet. In einem HSP-Baum können Periodizitätsschemas in beliebiger Tiefe geschachtelt vorliegen. Die Schachtelungstiefe ist ein Maß für die Tiefe dieser Schachtelung. Sie ermöglicht die Unterscheidung der unterschiedlichen Ebenen und ihrer Semantik. Die Semantik der Schachtelungstiefe wird in den folgenden Abschnitten noch genauer erläutert. Allerdings darf die Schachtelungstiefe nicht mit der "klassischen" Tiefe eines Baumes verwechselt werden. Es handelt sich hier um ein anderes semantisches Konzept.
- $NL : N \mapsto 2^{StructureSet \cup ActivitySet}$ ordnet jedem Knoten seine Menge an Labels zu. Die Menge von Labels entspricht einem kontrollierten Vokabular. Die Labelmenge *StructureSet* fasst die Labels zusammen, die zur Strukturierung von Periodizitätsschemas verwendet werden können. Die Labelmenge *ActivitySet* enthält alle Labels, die mit konkreten Aktivitäten (z.B. Medikamentengaben) in der realen Welt korrespondieren. Die Festlegung der beiden Labelmengen *StructureSet* und *ActivitySet* sind in Umfang und Gestalt im Wesentlichen anwendungsabhängig, wobei davon ausgegangen wird, dass $ActivitySet \cap StructuredSet = \emptyset$. Die unterschiedliche Semantik der Labelmengen bzw. damit verbundene Regeln werden später erläutert.
- $SP : N \mapsto \mathbb{Z} \times \mathbb{N} \times \mathbb{N} \times 2^{\mathbb{N}}$ mit $SP(n)$ ist eine Funktion, die jedem Knoten $n \in N$ mit $NT(n) \in \{\texttt{SNODE}, \texttt{PNODE}\}$ seine Auswahl- bzw. Projektionsparameter zuordnet. Die Parameter eines Knoten entsprechen dem Tupel (z, l_P, l_A, Pos), mit Instanziierungszeitpunkt $z \in \mathbb{Z}$, Musterlängen $l_P, l_A \in \mathbb{N}$ und der (nicht leeren) Menge an Musterpositionen $Pos := \{x | 0 \leq x < l_P\}$, $|Pos| > 0$. Die genaue Semantik der Auswahl- und Projektionsparameter wird später erläutert.

Zur übersichtlicheren Schreibweise werden im Folgenden alle Funktionen $F(x)$ durch F^x abgekürzt. In Abbildung 4.2 ist die im weiteren Verlauf verwendete graphische Darstellung der unterschiedlichen Knotentypen aufgezeigt.

Knoten der Typen `HNODE` (H-Knoten) und `ONODE` (Null-Knoten) werden durch Rechtecke[1], Knoten vom Typ `SNODE` (S-Knoten) durch Dreiecke und Knoten vom Typ `PNODE` (P-Knoten) durch Vierecke dargestellt. Die Labelmenge $L_0, ..., L_n$ eines Knotens (siehe



(a) H-Knoten (b) S-Knoten (c) P-Knoten

Abbildung 4.2: Grafische Darstellung der Knotentypen des HSP-Baums

[1]Knoten vom Typ `ONODE` weisen ihren Typ als Knotenbezeichnung auf.

unten) wird jeweils in dem grau hinterlegten Rechteck aufgeführt. Die Schachtelungstiefe (siehe unten) eines Knotens wird durch den Wert k dargestellt. Die Ordnungsrelation bzgl. der Nachfolger eines Knotens in einem geordneten Baum wird schließlich durch die nummerierten Kanten $(0, ..., h)$ dargestellt. Der Wert id bezeichnet die eindeutige Knoten-Id.

Die Semantik der unterschiedlichen Knotentypen wird im Folgenden erörtert und definiert. In den entsprechenden Definitionen gehen wir z.T. von unterschiedlichen Basisfunktionen eines geordneten Baums bzw. eines HSP-Baums $T = (N, E, ...)$ aus, die Tabelle 4.1 zeigt.

Name	Signatur	Beschreibung
$grad$	$N \mapsto \mathbb{N}$	$grad(n)$ bestimmt den Knotengrad des Knotens n, d.h. die Anzahl seiner Nachfolger.
$succ$	$N \times \mathbb{N} \mapsto N$	$succ(n, k) = m$ bestimmt den unmittelbaren Nachfolger m des Knotens n entlang Kante k von Baum T.
$succ$	$N \times 2^{\mathbb{N}} \mapsto 2^N$	$succ(n, K) = m$ mit $K \subseteq \{0, ..., grad(n)\}$ bestimmt die unmittelbaren Nachfolger m des Knotens n von Baum T entlang aller Kanten, die in der Menge K enthalten sind
$succ$	$N \mapsto 2^N$	$succ(n) = (m_0, ..., m_k)$ mit $m_0 := succ(n, 0), ..., m_k := succ(n, k)$ sowie $grad(n) = k - 1$ liefert eine Liste aller Nachfolger des Knotens n von Baum T. Die Ergebnisliste ist sortiert gemäß der Ordnungsrelation auf den Kanten.
$SUCC$	$N \mapsto 2^N$	$SUCC(n)$ bestimmte die Menge aller von Knoten n erreichbaren Nachfolger im Baum T.
$pred$	$N \mapsto N$	$pred(m) = n$ bestimmt den unmittelbaren Vorgänger n des Knotens m im Baum T.
$PRED$	$N \mapsto 2^N$	$PRED(m)$ bestimmte die Menge aller von Knoten m erreichbaren Vorgänger im Baum T.

Tabelle 4.1: Übersicht der verwendeten Basisfunktionen im HSP-Baum

4.2.1 H-Knoten

Die elementaren Komponenten des zyklenorientierten Modells, d.h. Zyklen, strukturierte Zyklen und Zyklusposition werden im HSP-Baum durch H-Knoten, d.h. Knoten vom Typ `HNODE` (**h**orizontal / **h**ierarchic node), abgebildet. Jeder (Teil-) Zyklus wird durch einen H-Knoten repräsentiert, wobei die Zykluslänge durch die Menge erreichbarer (nachfolgender) Blattknoten bestimmt wird. Jeder Blattknoten mit Typ `HNODE` stellt dem entsprechend eine Zyklusposition dar. Die Nummerierung der Zyklusposition ergibt sich aus dem Index der Kante, die zu dem entsprechenden Blattknoten führt (beginnend bei 0). In Abbildung 4.3 sind zwei einfache Beispiele für HSP-Bäume dargestellt.[2] Abbildung 4.3a zeigt den HSP-Baum für ein Periodizitätsschema einer Chemotherapie bestehend aus Zyklen der Länge 28 (Tage). In der Chemotherapie werden an den Tagen 1-14 ein Medikament A und am 1. und 8. Tag ein Medikament B verabreicht. Der Wurzelknoten v_0 repräsentiert den Zyklus `Chemotherapie`. Die Positionen $1, ..., 28$ des Zyklus werden durch die von Knoten v_0 erreichbaren (nachfolgenden) Blattknoten $v_1, ..., v_{28}$ entlang der Kantenindizes $0, ..., 27$ repräsentiert. Die Zykluslänge von 28 Tagen entspricht der Anzahl der Blattknoten. Die Medikamentengaben der Chemotherapie sind durch entsprechende Labelzuordnungen für die Blattknoten abgebildet, d.h. die Blattknoten $v_1, ..., v_{15}$ weisen das Label `A` und die Knoten v_1 und v_8 zusätzlich das Label `B` auf. In Abbildung 4.3b ist der HSP-Baum für das Periodizitätsschema der Granularität `Woche,` bestehend aus Zyklen der Länge 7 (Tage), dargestellt. Der Wurzelknoten v_0 repräsentiert eine einzelne Woche bzw. den Zyklus der Länge 7 (Tage). Die von Knoten v_0 erreichbaren Blattknoten $v_1, ..., v_7$ entsprechen den Zyklenpositionen $1, ..., 7$. Die Namen der Wochentage sind durch entsprechende Labelzuordnungen an den Blattknoten repräsentiert, wie etwa das Label `Mo` für Montag am Knoten v_1.

(a) Chemotherapie (28 Tage) (b) Granularität Woche (7 Tage)

Abbildung 4.3: Einfache Beispiele für HSP-Bäume

Die mengenbasierte, formale Repräsentation des HSP-Baumes aus Abbildung 4.3a ist wie folgt:

- $N = \{v_0, ..., v_{28}\}$
- $E = \{(v_0, v_1, 0), (v_0, v_2, 1), ..., (v_0, v_{28}, 27)\}$

mit den Eigenschaften:

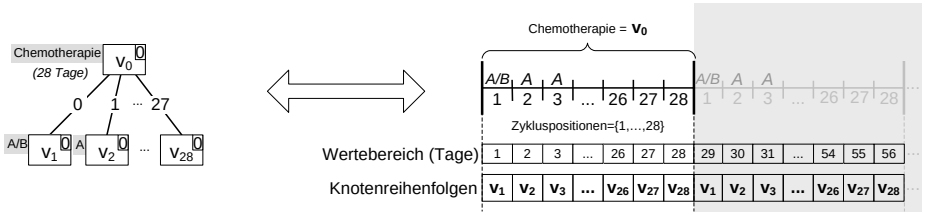
[2] Die Schachtelungstiefe ND eines Knotens wird im Folgenden zunächst vernachlässigt. Wir kommen später darauf zurück.

- $NT^{v_i} =$ `HNODE` für $i = 0, ..., 28$
- $ND^{v_i} = 0$ für $i = 0, ..., 28$
- $NL^{v_0} = \{$`Chemotherapie`$\}$,

  $NL^{v_1} = NL^{v_8} = \{$`A`, `B`$\}$, $NL^{v_i} = \{$`A`$\}$ für $\forall i \in \{1, ..., 28\}/\{1, 8\}$

  $StructureSet = \{$`Chemotherapie`$\}$; $ActivitySet = \{$`A`, `B`$\}$ (vgl. Def. 4.1)

Die Darstellung des durch den HSP-Baum in Abbildung 4.3a repräsentierten Periodizitätsschemas im zyklenorientierten Modell ist in Abbildung 4.4 dargestellt. Im allgemeinen Modell weisen alle (unendlich viele) Zyklen des Periodizitätsschemas die gleiche Länge und Labelbelegung auf. Die Repräsentation eines einzelnen Zyklus, wie im HSP-Baum ist somit ausreichend, um die gesamte (unendliche) Menge an Zyklen des Periodizitätsschemas zu repräsentieren. Durch ein – beliebig häufiges – Durchlaufen des HSP-Baumes kann die (theoretisch unendliche) Gesamtmenge der Zyklen des entsprechenden Periodizitätsschemas abgebildet werden. Allerdings ist das Durchlaufen des HSP-Baumes an dieser Stelle lediglich beispielhaft zu verstehen. Im Zusammenhang mit den nachfolgend erörterten Knotentypen erweist sich diese Traversierung als komplexer (siehe Kapitel 5).



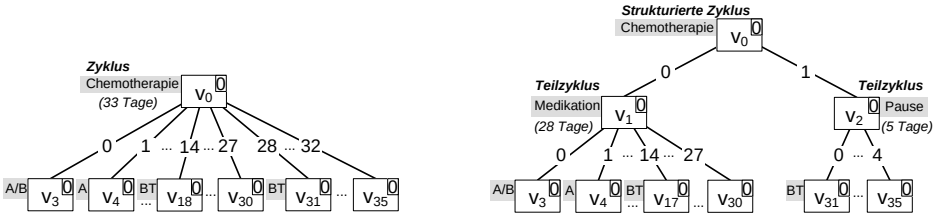
(a) HSP-Baum (b) Zyklenorientierte Modell

Abbildung 4.4: Darstellung eines Periodizitätsschemas im HSP-Baum und dessen entsprechende Darstellung im zyklenorientierten Modell

4.2.1.1 H-Knoten und strukturierte Zyklen

Die Baumstruktur eines HSP-Baum eignet sich sehr gut zur Abbildung von strukturierten Zyklen mit ihren horizontalen und vertikalen Strukturierungen.

Die horizontale Strukturierung eines (Teil-) Zyklus wird durch die Reihenfolge der Nachfolgerknoten des entsprechenden H-Knotens repräsentiert. Abbildung 4.5 zeigt einen HSP-Baum für eine `Chemotherapie` bestehend aus den Teilzyklen `Medikation` mit 28 Tagen und `Pause` mit 5 Tagen. Abbildung 4.5a stellt das Beispiel ohne (horizontale) Strukturierung als einen einzelnen Zyklus der Länge 33 (Tage) dar. In Abbildung 4.5b ist das Beispiel mit horizontaler Strukturierung dargestellt. Der Wurzelknoten v_0 repräsentiert den strukturierten Zyklus mit Label `Chemotherapie`. Dessen Nachfolgerknoten



(a) Ohne horizontale Strukturierung (b) Mit horizontaler Strukturierung

Abbildung 4.5: Beispiel für die Repräsentation einer horizontale Strukturierung im HSP-Baum

v_1 und v_2 repräsentieren die Unterteilung des "großen" Zyklus `Chemotherapie` in die zwei kleineren Teilzyklen `Medikation` und `Pause`.

Die vertikale Strukturierung eines strukturierten Zyklus mit ihren unterschiedlichen, hierarchischen Ebenen erfolgt im HSP-Baum ebenfalls auf Grundlage der entsprechenden Baumstruktur. Ausgehend von einem strukturierten (Teil-) Zyklus bzw. entsprechenden Knoten repräsentieren alle Nachfolgerknoten – sofern diese H-Knoten und keine Blattknoten sind – die vertikale Strukturierung mit ihren verschiedenen Ebenen. Abbildung 4.6 zeigt ein Beispiel für die Repräsentation der vertikalen Strukturierung im HSP-Baum. Der Knoten v_0 entspricht einem strukturierten Zyklus bestehend aus den beiden Teilzyklen `Medikation` (v_1) und `Pause` (v_2). Der Teilzyklus `Medikation` bzw. Knoten v_1 entspricht erneut einem strukturierten Zyklus bzw. Teilzyklus bestehend aus den beiden Teilzyklen `Medikationsphase` (v_{36}) und `Medikationsfrei` (v_{37}) jeweils mit Länge 14 (Tage). Kennzeichnend für den HSP-Baum ist, dass im Gegensatz zum allgemeinen Modell, die einzelnen Teilzyklen eines strukturierten Zyklus bzw. deren Zykluspositionen nicht in materialisierter Form repräsentiert werden. Stattdessen werden lediglich die Teilzyklen der tiefsten Strukturierungsebene im Detail im HSP-Baum repräsentiert. Für die (Teil-) Zyklen höherer Ebene gilt es, die Menge entsprechender Zykluspositionen von den im Baum tieferliegenden Zyklen abzuleiten. Genauer gesagt ergeben sich die Zykluspositionen eines (Teil-) Zyklus bzw. entsprechenden Knotens aus der Menge der von dem Knoten erreichbaren nachfolgenden Blattknoten. Beispielsweise ergeben sich im obigen Beispiel für den Teilzyklus `Medikation` bzw. Knoten v_1 die Zykluspositionen aus den erreichbaren Blattknoten $v_3, ..., v_{30}$.

Ein Baum, der nur aus H-Knoten besteht, bildet die elementarste Form eines HSP-Baums. Von diesem elementaren HSP-Baum leiten sich später alle weiteren Formen von HSP-Bäumen ab. In diesen nicht-elementaren Bäumen werden dann auch die weiteren Knotentypen $\{$`SNODE`, `PNODE`, `0NODE`$\}$ vorkommen bzw. benötigt werden. Hierbei wird die Schachtelungstiefe (siehe Def. 4.1) von Knoten eine wichtige Rolle spielen. Initial wird diese jedoch im elementaren HSP-Baum festgelegt, genauer gesagt, es wird festgelegt, dass die Schachtelungstiefe mit Null beginnt. Das bedeutet jedoch, dass später alle Knoten mit Schachtelungstiefe = Null, als die Knoten des elementaren Baums erkannt

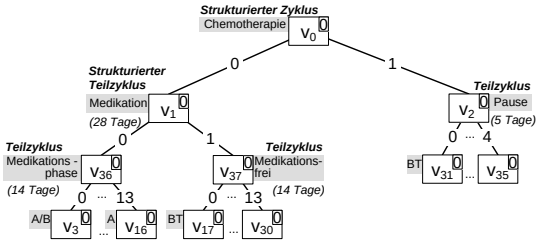


Abbildung 4.6: Beispiel einer vertikalen Strukturierung im HSP-Baum

werden können und stets mit (Teil-) Zyklen oder Zyklluspositionen korrespondieren. Formal ist die Initiierung der Schachtelungstiefe in der Strukturregel SR-1 definiert.

Strukturregel SR-1 (H-Knoten):
Sei $T = (N, E, ...)$ ein HSP-Baum, der nur H-Knoten enthält. Wir fordern stets, dass für die Schachtelungstiefe gilt: $\forall n \in N : ND^n = 0$.

4.2.1.2 H-Knoten und Labels

Die Funktion $NL : N \mapsto 2^{StructureSet \cup ActivitySet}$ ordnet jedem Knoten in einem HSP-Baum eine Menge an Labels zu. Beide Labelmengen *StructureSet* und *ActivitySet* können als anwendungsabhängiges, kontrolliertes Vokabular verstanden werden. Allerdings unterscheiden sich die beiden Mengen in der verwendeten Semantik.

Aus Anwendungssicht dienen Labels der Menge *StructureSet* der Bezeichnung von Zyklen, Teilzyklen oder Zyklluspositionen. Allerdings fordern wir stets, dass in einem beliebigen Pfad von der Wurzel aus ein Label niemals doppelt vorkommt. Solche Doppelbenennung würden später bei der Realisierung von Labelausdrücken zu Inkonsistenzen bzw. Unbestimmtheiten führen (siehe Abschnitt 4.3.1).

Die Menge *ActivitySet* enthält die Labels, die mit konkreten Aktivitäten (z.B. Medikamentengabe) der realen Welt korrespondieren und in ihrer Anwendung auch zeitliche Dauern aufweisen können. Allerdings fordern wir in der vorliegenden Arbeit stets, dass Labels aus der Menge *ActivitySet* lediglich Blattknoten zugeordnet sind. Der Hintergrund für diese Forderung besteht darin, dass solche Label in ihrer Semantik der einmaligen "Ausführung" einer bestimmten Aktivität in einem (Teil-) Zyklus entsprechen. Falls wir die Verwendung solcher Labels auch an Nicht-Blattknoten zulassen würden, führt dies automatisch zu einer Zuordnung zu einem (Teil-) Zyklus. Das heißt, eine entsprechende Labelzuordnung entspricht der Aussage "*die Aktivität findet einmalig irgendwann innerhalb des entsprechenden (Teil-) Zyklus statt*". Wann genau die Aktivität durchzuführen wäre, wird jedoch nicht festgelegt. Dies wiederum führt zu "Unschärfen", die lediglich auf Ebene der Zyklluspositionen ignoriert werden können. Zyklluspositionen stellen die tiefste darstellbare Ebene dar, wodurch sich die Unschärfe auf dieser Ebene verliert. In allen anderen Fällen bzw. höheren Ebenen entspricht die Zuordnung eines

Labels der Menge *ActivitySet* einer unscharfen Angabe bzw. führt zu einem unscharfen Periodizitätsschema. Wie bereits in Kapitel 2 erläutert, stehen diese Klassen von Periodizitätsschemas nicht im Fokus dieser Arbeit.

Die obigen Beschränkungen zur Verwendung von Labels im HSP-Baum sind in der folgenden Strukturregel formal definiert.

Strukturregel SR-2 (Labels):
Sei $T = (N, E, ...)$ ein HSP-Baum und $W = (v_0, ..., v_k)$ ein beliebiger Pfad im Baum T, d.h. für alle i und j aus $\{1, ..., k\}$ gilt, dass $v_i \neq v_j$, falls $i \neq j$. Wir fordern stets, dass gilt:

1. Labels aus der Menge *ActivitySet* dürfen nur bei Blattknoten vorkommen:
   $(\forall n \in N : \exists l \in NL^n : l \in ActivitySet) \Rightarrow n$ ist ein Blattknoten.
2. Labels dürfen in einem Pfad im Baum nicht mehrfach vorkommen:
   $\forall v_i, v_j \in W : NL^{v_i} \cap NL^{v_j} = \emptyset,\ i \neq j$

4.2.1.3 H-Knoten und Periodizitätsschemas

Ein Periodizitätsschema im zyklenorientierten Modell entspricht einer (theoretisch) unendlichen Menge an Zyklen bzw. strukturierten Zyklen (siehe Abschnitt 3.3.4). Das Wiederholungsmuster für die Reihenfolgen der Zyklen des Periodizitätsschemas wird durch den HSP-Baum repräsentiert. Der HSP-Baum in Abbildung 4.7 repräsentiert ein Periodizitätsschema, das aus den strukturierten Zyklus `Chemotherapie` und dem Zyklus `Ambulante Pause` besteht, die sich regelmäßig abwechseln (siehe Abbildung 3.9 in Abschnitt 3.3.4). Allerdings wird im HSP-Baum nicht unterschieden, ob ein Periodizitätsschema aus einem strukturierten Zyklus `Chemotherapie` besteht, der sich mit einem Zyklus `Ambulante Pause` abwechselt oder ob das Schema aus einem strukturierten Zyklus `vollständige Chemotherapie` besteht mit den beiden Teilzyklen `Chemotherapie` und `Ambulante Pause`. Im HSP-Baum wird ein Periodizitätsschema stets als ein "großer", strukturierter Zyklus verstanden bzw. dargestellt. Dieser durch den Wurzelknoten eines HSP-Baumes repräsentierte "große", strukturierte Zyklus stellt das Wiederholungsmuster des Periodizitätsschemas dar.

4.2.2 S-Knoten

Im HSP-Baum dienen Knoten vom Typ `SNODE` (**s**election **n**ode) bzw. S-Knoten der Abbildung des Ergebnisses der Auswahloperationen *Zyklusauswahl* und *Zyklusauswahl* (siehe Abschnitt 3.4.3). Zunächst erfolgen jedoch Vorbetrachtungen zur Abbildung von Auswahloperationen im HSP-Baum.

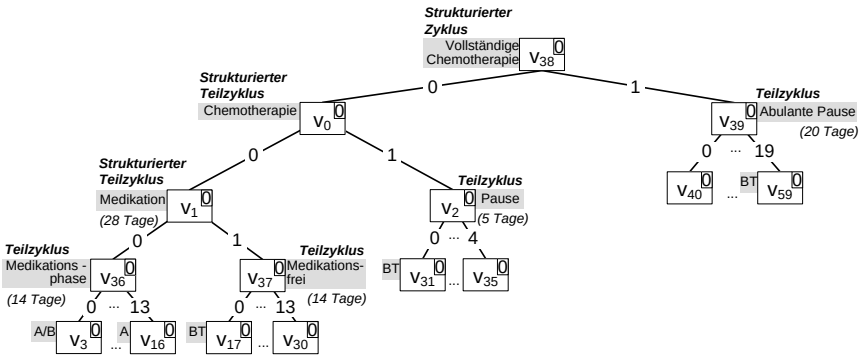


Abbildung 4.7: Beispiel eines HSP-Baum für ein Periodiziätsschemas mit mehrfach strukturierten Zyklen

4.2.2.1 Vorbetrachtungen zur Abbildung von Auswahloperationen im HSP-Baum

Die Auswahloperationen des zyklenorientierten Modells ermöglichen es, basierend auf einem periodischen Auswahlschema, bestimmte (Teil-) Zyklen oder Zykluspositionen eines Periodizitätsschemas zu kennzeichnen bzw. auszuwählen. Hierzu wird das periodische Auswahlschema als periodische Angabe verstanden, die über der Auswahlmenge des Periodizitätsschemas interpretiert wird, d.h. über der Menge an (Teil-) Zyklen oder Zykluspositionen, für die eine Auswahl vorgenommen wird. In Abhängigkeit von der Semantik einer Auswahloperation werden bei der Interpretation die Labelzuordnungen und Strukturen des Auswahlschemas auf die korrespondierenden Strukturen des Periodizitätsschemas übertragen. Auf Basis dieser zusätzlichen Labelzuordnung und Strukturen können später (Teil-) Zyklen oder Zyklenpositionen mit entsprechenden Labelausdrücken adressiert werden, etwa um diese (Teil-) Zyklen oder Zyklenpositionen zu manipulieren. Abbildung 4.8 zeigt ein Beispiel einer *Zyklusauswahl* für "*Jeder 2.Zyklus Med(ikation) einer Chemotherapie*". Die Auswahlmenge entspricht hier der Menge aller Zyklen mit Label `Med`. Im zyklenorientierten Modell basieren die bei einer Auswahloperation verwendete Auswahlmenge, das periodische Auswahlschema, und die Repräsentation der Ergebnisse auf (theoretisch) unendlichen Mengen bzw. Periodizitätsschemas. So werden zur Abbildung des Ergebnisses jedem der (theoretisch) unendlich vielen von einer Auswahloperation betroffenen Zyklen die entsprechenden Labels des Auswahlschemas zugeordnet sowie zusätzliche Strukturierungen eingeführt (siehe unterer Bereich in Abbildung 4.8). Im Zusammenhang mit dem HSP-Baum ergibt sich jedoch die Herausforderung, eine endliche (berechenbare) Repräsentation bei der Abbildung von Auswahloperationen zu finden.

Die Voraussetzung für die Abbildung einer Auswahloperation im HSP-Baum bildet die endliche Repräsentation der Auswahlmenge und des periodischen Auswahlschemas. Ein periodisches Auswahlschema wird im Allgemeinen als "normales" Periodizitätsschema verstanden, das über der (theoretisch) unendlichen Auswahlmenge interpretiert wird.

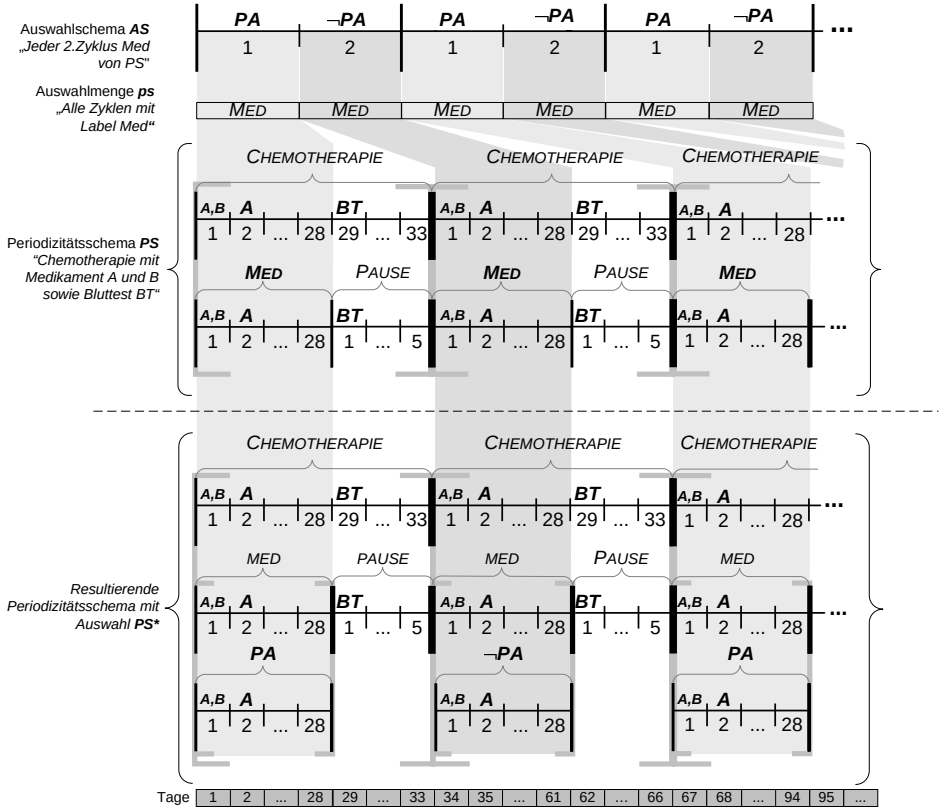


Abbildung 4.8: Beispiel einer Zyklusauswahl im zyklenorientierten Modell

(a) Äquidistantes unstrukturiertes Auswahlschema

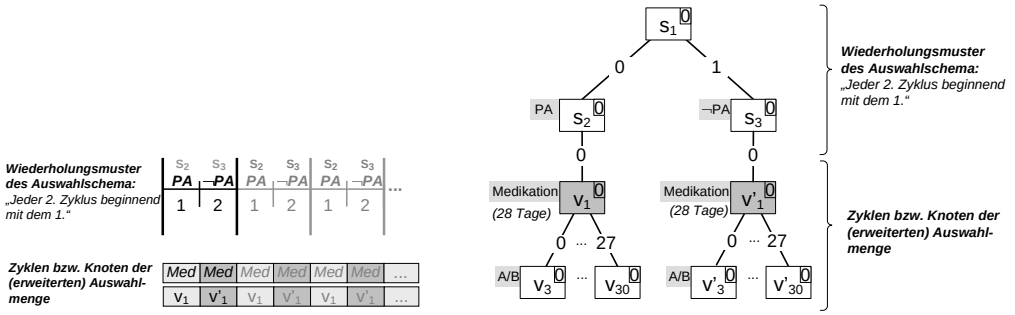
(b) Periodizitätsschema mit Auswahlmenge $\{v_1\}$

Abbildung 4.9: Beispiel für die endliche Repräsentation einer Auswahlmenge und (periodischen) Auswahlschemas

Dies bedeutet jedoch, dass ein periodisches Auswahlschema ebenfalls als (endlicher) HSP-Baums repräsentiert werden kann. Beispielsweise ist der HSP-Baum für das einfache Auswahlschemas "*Jeder 2. Zyklus (Med einer Chemotherapie) beginnend bei dem Ersten*" in Abbildung 4.9a dargestellt. Die Labels `PA` bzw. ¬`PA` bezeichnen die von der Auswahl betroffenen bzw. nicht betroffenen Teilzyklen. Im zyklenorientierten Modell wird die Auswahlmenge lediglich als eine "beliebige" Menge von Zyklen bzw. Zyklenpositionen verstanden. Im HSP-Baum hingegen basiert die Angabe der Auswahlmenge stets auf der Verwendung von Labelausdrücken. Wie wir später noch im Detail zeigen werden, führen Labelausdrücke im HSP-Baum stets zu endlichen Mengen von H-Knoten. Eine solche endliche Menge von H-Knoten repräsentiert die (theoretisch) unendliche Auswahlmenge. In Abbildung 4.9b ist der HSP-Baum des Periodizitätsschemas PS des Beispiels aus Abbildung 4.8 dargestellt. Für die Auswahl "*Jeder 2. Zyklus Medikation einer Chemotherapie beginnend bei dem Ersten*" wird die Auswahlmenge (d.h. alle Knoten mit Label `Medikation`*)* durch den einzelnen Knoten v_1 repräsentiert.

Die Grundidee der endlichen Abbildung von Auswahloperationen besteht darin, anstelle des resultierenden Periodizitätsschemas PS^* (siehe unterer Bereich in Abbildung 4.8) lediglich dessen "Konstruktion" bestehend aus Auswahlschema und -menge (siehe oberer Bereich in Abbildung 4.8) zu repräsentieren. Genauer gesagt, basierend auf dem Auswahlschema wird lediglich das "*Wiederholungsmuster des Auswahlschemas über der Auswahlmenge*" repräsentiert (siehe Abbildung 4.10a). Dieses Wiederholungsmuster beschreibt die Reihenfolge von ausgewählten und nicht ausgewählten Zyklen bzw. Zyklenpositionen aus der Auswahlmenge. Im obigen Beispiel führt das Auswahlschema zu einem Wiederholungsmuster mit Zykluslänge = 2, bei dem sich ein ausgewählter Zyklus `Med(ikation)/PA` regelmäßig mit einem nicht ausgewählten Zyklus `Med(ikation)/`¬`PA` abwechselt.

Das Wiederholungsmuster des Auswahlschemas über der Auswahlmenge kann als "normales" Periodizitätsschema verstanden werden und somit ebenfalls als HSP-Baum repräsentiert werden. Die Repräsentation als HSP-Baum sowie die (unendliche) zyklenorientierte Repräsentation des Wiederholungsmusters ist in Abbildung 4.10 dargestellt. Allerdings muss bei Bildung des Wiederholungsmusters die Auswahlmenge an die Zy-



(a) Zyklenorientierte Darstellung (b) HSP-Baumdarstellung

Abbildung 4.10: Wiederholungsmuster des Auswahlschemas über der Auswahlmenge

kluslänge des Auswahlschemas angepasst bzw. bezüglich dieser erweitert werden. Hierzu werden, ausgehend von dem Zyklus der Auswahlmenge, weitere (unmittelbare) Nachfolgerzyklen aus der (theoretisch) unendlichen Auswahlmenge "materialisiert". Im vorliegenden Beispiel muss bezüglich der Zykluslänge = 2 des Auswahlschemas, die Auswahlmenge mit dem einzelnen Zyklus bzw. Knoten v_1 lediglich um einen weiteren Knoten erweitert werden. In Abbildung 4.10b entspricht der Knoten v'_1 dem materialisierten, unmittelbaren Nachfolgerzyklus von Knoten v_1. Somit repräsentieren sowohl Knoten v_1 als auch Knoten v'_1 jeweils einen Zyklus `Medikation`. Während v_1 dem ersten Zyklus bzw. dem ersten Vorkommen des Zyklus `Medikation` entspricht, wird durch v'_1 das zweite Vorkommen des Zyklus repräsentiert.

Durch eine Auswahloperation wird der HSP-Baum des Wiederholungsmusters des Auswahlschemas[3] als Teilbaum in den HSP-Baum des von der Auswahl betroffenen Periodizitätsschemas eingefügt. Beispielsweise ist der resultierende HSP-Baum der einfachen Auswahl "*Jeder 2. Zyklus (Medikation einer Chemotherapie) beginnend bei der Ersten*" in Abbildung 4.11 dargestellt. Der eingefügte Teilbaum ersetzt den bisherigen Knoten bzw. Zyklus `Medikation`. Durch Einfügen des Teilbaums entsteht im HSP-Baum des Schemas PS eine sogenannte "Auswahlebene", die die Struktur des periodischen Auswahlschemas repräsentiert (siehe später). Jede Auswahlebene wird stets durch einen S-Knoten eingeleitet bzw. durch diesen explizit gekennzeichnet.[4] Da es sich im vorliegenden Beispiel lediglich um ein einfaches äquidistantes Auswahlschema ohne Strukturierung handelt, besteht die Auswahlebene lediglich aus dem einzelnen S-Knoten s_1. Die Zykluslänge (= 2) des Auswahlschemas ergibt sich durch die Anzahl der unmittelbaren Nachfolger des S-Knotens s_1. Der Knoten s_1 umfasst alle Zyklen `Medikation`, unabhängig davon ob es sich um ausgewählte oder nicht ausgewählte Zyklen handelt. Die unmittelbaren Nachfolgerknoten der Auswahlebene werden als "*Blätter der Auswahlebene*" bezeichnet und repräsentieren die Auswahlmenge, d.h. die Menge an Zyklen

[3]Bei dem Einfügen wird eine angepasste Variante des ursprünglichen HSP-Baums verwendet, was die Realisierung von Labelausdrücken erleichtert (siehe später).
[4]Auf die Semantik der Schachtelungstiefe wird in den nachfolgenden Abschnitten eingegangen.

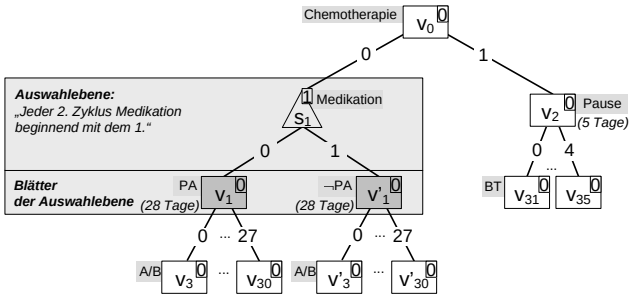


Abbildung 4.11: Beispiel für die Abbildung einer Auswahl im HSP-Baum

bzw. Zyklenpositionen, die von der Auswahloperation betroffen sind. Im Beispiel stellen die Knoten v_1 und v'_1 die Blätter der Auswahlebene dar. Der Knoten v_1 (`PA`) repräsentiert die Menge an ausgewählten Teilzyklen (Zyklusposition im Auswahlschema = 0) bzw. der Knoten v'_1 (¬`PA`) die Menge nicht ausgewählter Teilzyklen (Zyklusposition im Auswahlschema = 1).

Die Auswahlebene repräsentiert, zusammen mit ihren Blättern, im HSP-Baum das Wiederholungsmuster des Auswahlschemas über der Auswahlmenge. Erst durch eine Auswertung des resultierenden HSP-Baums einer Auswahloperation ergibt sich das Periodizitätsschema PS^*. Im dargestellten Beispiel ergibt sich bei entsprechender Auswertung des HSP-Baums die Zyklenreihenfolge `Medikation`/`PA` $(s_1/v_1) \rightarrow$ `Pause`(v_2) $\rightarrow$ `Medikation`/¬`PA` $(s_1/v'_1) \rightarrow$ `Pause`(v_2). Der erste Teilzyklus `Medikation` wird durch Knoten v_1 repräsentiert und die nachfolgende Pause durch Knoten v_2. Die zweite Wiederholung bzw. das zweite Vorkommen des Zyklus `Medikation` wird durch Knoten v'_1 repräsentiert und die nachfolgende Pause erneut durch Knoten v_2 (und anschließend wieder beginnend bei s_1 bzw. v_1 und v_2). Die resultierende Zyklenreihenfolge einer solchen Auswertung entspricht dem Wiederholungsmuster des Periodizitätsschema PS^* einer Chemotherapie mit Auswahl (siehe Abbildung 4.8 unterer Bereich).

Beim dargestellten Beispiel handelt es sich um den einfachsten Fall, d.h. eine Auswahloperation mit äquidistanten, unstrukturiertem Auswahlschema und einer Auswahlmenge, die durch einen einzelnen H-Knoten beschrieben werden kann. Die weiteren Fälle mit strukturiertem Auswahlschema bzw. mehrelementiger Auswahlmenge werden in den folgenden Abschnitten 4.2.2.2 bzw. 4.2.2.3 vorgestellt.

4.2.2.2 Strukturierte Auswahlschemas, Auswahlebenen und Schachtelungstiefe

Jeder S-Knoten im HSP-Baum stellt den Beginn einer Auswahlebene dar bzw. leitet eine solche Ebene ein. Semantisch wird durch eine Auswahlebene und ihren nachfolgenden Knoten das Wiederholungsmuster der Interpretation des Auswahlschemas über der Auswahlmenge repräsentiert. Die Struktur des Auswahlschemas wird durch die Auswahlebene abgebildet. Im einfachsten Fall, d.h. bei Vorliegen eines äquidistanten Auswahl-

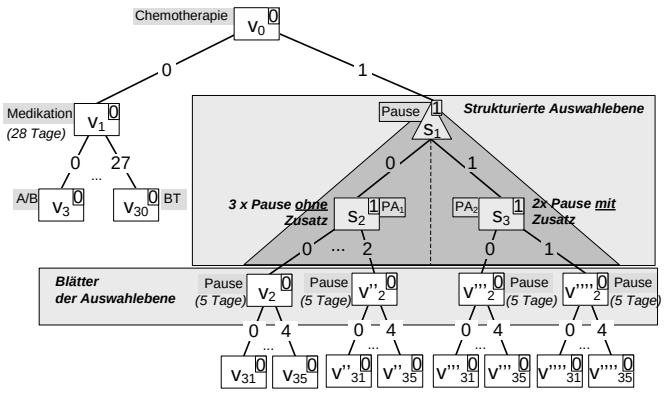


Abbildung 4.12: Abbildung einer Zyklusauswahl im HSP-Baum

schemas ohne Strukturierung, entspricht die Auswahlebene einem einzelnen S-Knoten. Ein Beispiel zeigt Abbildung 4.11. In den anderen Fällen, d.h. bei Vorliegen von strukturierten Auswahlschemas bzw. Auswahlschemas mit Ausnahmen, wird die Struktur des Auswahlschemas auf die Auswahlebene übertragen.

Die Abbildung strukturierter Auswahlschemas bzw. die Übertragung der Struktur auf die entsprechende Auswahlebene erfolgt im HSP-Baum analog zu Periodizitätsschemas, d.h. auf Basis von H-Knoten. Zur Unterscheidung der H-Knoten des Auswahlschemas von denen des Periodizitätsschemas dient die Schachtelungstiefe (ND) der Knoten (siehe Def. 4.1). Der S-Knoten der Auswahlebene sowie die zugehörigen nachfolgenden H-Knoten zur Abbildung der horizontalen und vertikalen Strukturierung des Auswahlschemas weisen die gleiche Schachtelungstiefe auf. Diese Schachtelungstiefe unterscheidet sich von der Schachtelungstiefe der Knoten des Periodizitätsschemas. Abbildung 4.12 zeigt ein Beispiel einer Auswahl mit strukturierten Auswahlschema für den Teilzyklus "*Pause*" einer Chemotherapie. Die Auswahlebene wird in diesem Beispiel durch das grau hinterlegte Dreieck gekennzeichnet. In der Chemotherapie wird in den ersten drei Pausen nach dem Medikationsteilzyklus kein Zusatzmedikament verabreicht, aber in den nachfolgenden zwei Pausen (Pausen 4 und 5). Das Auswahlschema ist somit horizontal in die zwei Teilzyklen `Pause_ohne_Zusatz` ($\texttt{PA}_1$) (Zykluslänge = 3) und `Pause_mit_Zusatz` ($\texttt{PA}_2$)" (Zykluslänge = 2) unterteilt. Nach erfolgter Auswahl kann mit den Labels $\texttt{PA}_1$ und $\texttt{PA}_2$ auf die entsprechenden Teilmengen von Pausen zugegriffen werden.

Der S-Knoten s_1 mit Schachtelungstiefe $ND^{s_1} = 1$ kennzeichnet den Beginn der Auswahlebene für den Teilzyklus `Pause`. Das verwendete Auswahlschema bzw. die resultierende Auswahlebene ist horizontal in die beiden Teilauswahlen `Pause_ohne_Zusatz` (Länge = 3, Knoten s_2) und `Pause_mit_Zusatz` (Länge = 2, Knoten s_3) strukturiert. Beide Knoten, s_2 und s_3, weisen die gleiche Schachtelungstiefe auf wie der Knoten s_1 ($ND(s_1) = ND(s_2) = ND(s_3) = 1$), was eine Unterscheidbarkeit von den "normalen" H-Knoten des Periodizitätsschemas (Schachtelungstiefe = 0) ermöglicht. Die Blätter der Auswahlebene, d.h. Knoten $v_2, ..., v_2''$ entsprechen den Teilzyklen der Teil-

auswahl `Pausen_ohne_Zusatz` und die Knoten v_2'''', v_2'''' den Teilzyklen der Teilauswahl `Pausen_mit_Zusatz`.

Die oben am Beispiel erläuterten Strukturen *Auswahlebene* und *Blätter der Auswahlebene* zur Abbildung von periodischen Auswahloperationen im HSP-Baum werden im Folgenden in ihrer Struktur und Semantik formal charakterisiert.

Auswahlebene: Im Allgemeinen entspricht jede Auswahlebene einem Teilbaum innerhalb eines HSP-Baums. Im Fall eines einfachen Auswahlschemas ohne Strukturierung oder Ausnahmen, entspricht ein einzelner S-Knoten, der kleinsten bzw. minimalen Auswahlebene. In allen anderen Fällen, d.h. bei Vorliegen komplexer Auswahlschemas mit entsprechenden Strukturierungen, wird die Schachtelungstiefe der Knoten benötigt. Diese Schachtelungstiefe ermöglicht eine eindeutige Erkennung und Abgrenzung von Auswahlebenen im HSP-Baum. Der konkrete Wert der Schachtelungstiefe spielt hierbei weniger eine Rolle. Vielmehr ist zum korrekten Abgrenzen der Auswahlebenen das relative Verhältnis der Schachtelungstiefen der Knoten untereinander wichtig.

Jede Auswahlebene wird durch einen S-Knoten mit einer bestimmten Schachtelungstiefe n eingeleitet. Einer Auswahlebene werden dann alle (nachfolgenden) H-Knoten zugeordnet bzw. sind Bestandteil der Auswahlebene, die die gleiche Schachtelungstiefe wie der einleitende S-Knoten aufweisen. Entsprechende H-Knoten repräsentieren die ursprüngliche Struktur des Auswahlschemas einer Auswahloperation bzw. der resultierenden Auswahlebene. In einer Auswahlebene können auch weitere Auswahlebenen enthalten sein. Dies ist der Fall, wenn eine Auswahlebene bzw. ein entsprechendes Auswahlschema selber wiederum Ausnahmen beinhaltet (siehe Abschnitt 3.4.3.1). Dies führt im HSP-Baum zu "geschachtelten" Auswahlebenen, die erneut durch einen S-Knoten eingeleitet werden. Dieser S-Knoten bzw. alle der geschachtelten Auswahlebene zugeordnete Knoten weisen dann eine größere Schachtelungstiefe als die höhere Auswahlebene, d.h. die Auswahlebene in der die geschachtelte Auswahlebene enthalten ist. Anhand der größeren Schachtelungstiefe kann die geschachtelte Auswahlebene (bzw. ihre Knoten) eindeutig innerhalb der höheren Auswahlebenen identifiziert bzw. von dieser abgegrenzt werden.

Nicht zu einer Auswahlebenen zugerechnet werden alle S-Knoten, die die gleiche Schachtelungstiefe wie der Wurzelknoten der Auswahlebene aufweisen. Diese S-Knoten bilden Wurzelknoten für eigene, nicht geschachtelte Auswahlebenen (siehe später). Ebenso enthält eine Auswahlebene keine Projektionsknoten (P-Knoten). Diese Knoten dienen der Abbildung der Projektionsauswahl (siehe Abschnitt 3.4.3.2) und sind semantisch innerhalb einer Auswahlebene nicht sinnvoll.

Formal ist eine Auswahlebene im HSP-Baum, wie folgt definiert:

Definition 4.2 (Auswahlebene): *Sei $T = (N, E, ...)$ ein HSP-Baum und $n \in N$ ein S-Knoten. Der von Knoten n erreichbare HSP-Teilbaum $T' = (N', E', ...)$ wird als Auswahlebene σ^n bezeichnet, falls gilt:*
$\forall n' \in N', n' \neq n : NT^{n'} \neq \texttt{PNODE}$ *und*

$$NT^{n'} \in \{\texttt{HNODE}, \texttt{ONODE}\} \quad \Rightarrow \quad ND^{n'} = ND^n$$
$$NT^{n'} = \texttt{SNODE} \quad \Rightarrow \quad (ND^{n'} = ND^n + 1) \wedge \sigma^n \textit{ ist eine Auswahlebebene}$$

Eine Auswahlebene wird formal als ein vom einleitenden S-Knoten erreichbarer Teilbaum verstanden. Das heißt, der einleitende S-Knoten der Auswahlebene bildet die Wurzel und alle nach unten erreichbaren Knoten bzw. Nachfolgerknoten, für die die angegebenen Bedingungen gelten, die Knoten des Teilbaums. Im Falle einer geschachtelten Auswahlebene wird verlangt, dass diese eine um den Wert Eins höhere Schachtelungstiefe aufweist, als die Schachtelungstiefe der höheren Ebene. Aufgrund der rekursiven Definition wird zusätzlich verlangt, dass alle geschachtelten Auswahlebenen selber wieder Auswahlebenen darstellen. Das heißt, es müssen erneut alle Bedingungen, die im Allgemeinen an eine Auswahlebene gestellt werden, gelten.

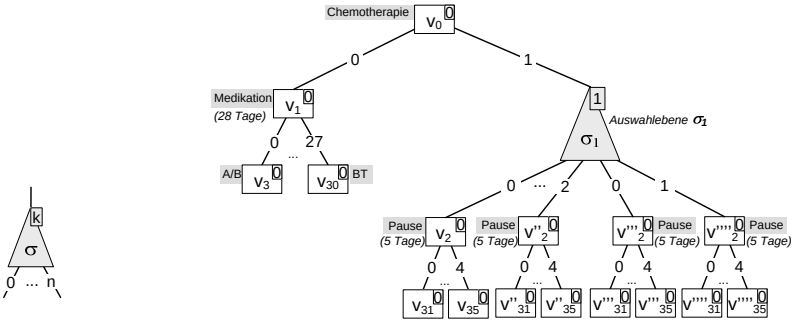
Zur Sicherstellung der korrekten Abbildung von Auswahlschemas bzw. Auswahloperationen durch Auswahlebenen im HSP-Baum müssen noch weitere zusätzliche Bedingungen sichergestellt werden. In der folgenden Strukturregel SR-3 sind diese Bedingungen festgelegt.

Strukturregel SR-3 (Auswahlebene):
Sei $T = (N, E, ..)$ ein HSP-Baum und Σ die Menge aller Auswahlebenen von T. Wir fordern stets, dass gilt:

1. Vollständigkeit:
   $\nexists n \in N : NT^n = \texttt{SNODE} \wedge \nexists \sigma^n \in \Sigma$
2. Abgeschlossenheit:
   $\forall \sigma^n = (N', E', ...) \in \Sigma$: $\forall n' \in N' : \forall k \in succ^{n'} : k \in N' \vee k \in \sigma^n_{leaf}$
3. Jeder Auswahlebene ist ein Tupel an Auswahlparametern zugeordnet:
   $\forall \sigma^n \in \Sigma : SP^n \neq \emptyset$

Die Strukturregel stellt als Erstes sicher, dass alle in einem HSP-Baum vorkommenden S-Knoten stets zu einer korrekten Auswahlebene führen bzw. eine solche einleiten. Die Strukturregel besagt zum Zweiten, dass jeder Knoten einer Auswahlebene stets einen Nachfolger aufweist. Der Nachfolgerknoten kann entweder erneut zu der Auswahlebene gehören oder zu der Menge der "Blätter der Auswahlebene"[5]. Knoten innerhalb der Auswahlebene, die keinen Nachfolger aufweisen, sind nicht zulässig. Semantisch wür-

[5]Die "Blätter der Auswahlebene" werden nachfolgend noch im Detail eingeführt. Die verwendete Funktion σ^n_{leaf} ermittelt die Blätter einer Auswahlebene.



(a) Vereinfachte Darstellung

(b) Beispiel einer vereinfachten Darstellung

Abbildung 4.13: Vereinfachte Darstellung von Auswahlebenen

de dies einer Auswahl ohne zugeordneten Zyklus bzw. Zyklusposition entsprechen, was nicht sinnvoll ist. Vereinfacht ausgedrückt, die Knoten einer Auswahlebene können niemals Blattknoten des HSP-Baums T darstellen. Des Weiteren wird in der Strukturregel festgelegt, dass jeder Auswahlebene ein Tupel an Auswahlparametern zugeordnet ist. Die Auswahlparameter sind später für die Auswertung eines HSP-Baums wesentlich. Im Folgenden können diese Parameter jedoch zunächst vernachlässigt werden. Eine genaue Betrachtung der Semantik der Auswahlparameter erfolgt erst im Abschnitt 4.2.5 bzw. im Detail in Kapitel 5.

Allerdings können strukturierte Auswahlschemas unter Umständen zu sehr umfangreichen Auswahlebenen führen. Aus Gründen der Übersicht verwenden wir im Folgenden eine vereinfachte Darstellung von Auswahlebenen, die Details bei der Struktur der Auswahlebene ausblendet (siehe Abbildung 4.13a). Die vereinfachte Darstellung des Beispiels aus Abbildung 4.12 ist in Abbildung 4.13b dargestellt.

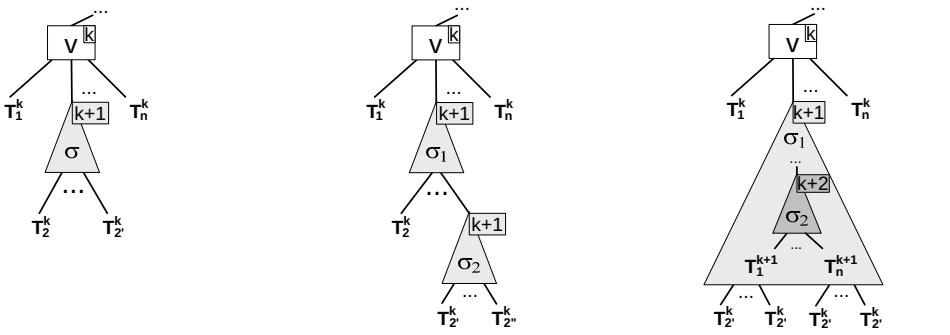
In den bisherigen Beispielen lag lediglich eine einzelne Auswahlebene im HSP-Baum vor, was der einmaligen Anwendung einer Auswahloperation entspricht. Allerdings können entsprechende Operationen auf einem Periodizitätsschema beliebig oft stattfinden. Infolgedessen können auch beliebig viele Auswahlebenen in einem HSP-Baum enthalten sein. Eine Auswahl besonderer Grundformen von Auswahlebenen ist in Abbildung 4.14 dargestellt.

Einfache Auswahlebene: Eine einfache Auswahlebene entspricht einer einmaligen Auswahloperation in einem Periodizitätsschema. Alle vorherigen Beispiele gehören zu dieser Grundform, die in Abbildung 4.14a dargestellt ist.

Gekoppelte Auswahlebene: Eine gekoppelte Auswahlebene beschreibt die mehrmalige Anwendung von Auswahloperationen, wobei für ein Blatt bzw. mehrere Blätter der ersten Auswahlebene σ_1 erneut eine Auswahl vorgenommen wird. Im (schematischen) Beispiel in Abbildung 4.14b bildet die Auswahlebene σ_1 die erste Auswahlebene. Für

ein Blatt der Ebene σ_1 wurde eine weitere Auswahl vorgenommen, die zu der Auswahlebene σ_2 führte. Ein Beispiel hierfür stellt die erste Auswahl "*In jedem 2. Zyklus* `Medikation` *einer Chemotherapie (beginnend beim 1.) findet ein Bluttest statt*" und die zweite Auswahl "*in jedem 3. Zyklus* `Medikation` *einer Chemotherapie (beginnend beim letzten) findet die Gabe eines Zusatzmedikaments statt".* Auf Basis der Auswahloperation kann somit der Zyklus ausgewählt bzw. adressiert werden, der sowohl einen Bluttest als auch eine Gabe des Zusatzmedikaments enthält, etwa um sicherzustellen, dass das Zusatzmedikament erst nach dem Bluttest gegeben wird.

Geschachtelte Auswahlebene: Eine geschachtelte Auswahlebene entspricht einer Auswahl innerhalb einer Auswahlebene. Dies ist der Fall, wenn das Auswahlschema selber wieder Ausnahmen beinhaltet, wie Abbildung 4.14c zeigt. Kennzeichnend ist hierbei, dass lediglich Auswahlebenen bei einer Schachtelung verwendet werden können. Ein typisches Beispiel stellt die Definition von Schaltjahren dar. Die Definition "*Jedes 4. Jahr*" und die beiden Ausnahmen "*Jedes 100. Jahr nicht*" aber "*Jedes 400. Jahr*" kann in Form einer geschachtelten Angabe beschrieben werden.



(a) Einfache Auswahl (b) Gekoppelte Auswahl (c) Geschachtelte Auswahl

Abbildung 4.14: Grundformen von Auswahlebenen

Blätter der Auswahlebene: Eine Auswahlebene stellt semantisch lediglich die Struktur des Auswahlschemas einer Auswahloperation dar. Erst durch die direkten Nachfolgerknoten einer Auswahlebene wird die Auswahlmenge repräsentiert, d.h. die Menge an (Teil-) Zyklen oder Zyklenspositionen, die von einer Auswahl betroffen sind (siehe Abschnitt 4.2.2.1). Diese unmittelbaren Nachfolgerknoten einer Auswahlebene werden im Folgenden als *"Blätter der Auswahlebene"* bezeichnet. Im Beispiel in Abbildung 4.12 entsprechen die Knoten $v_2, ..., v_2''''$ den *Blättern der Auswahlebene* σ^{s_1}. Formal sind die Blätter der Auswahlebene wie folgt definiert:[6]

[6]Wie erwähnt, werden in der folgenden Definition und in den folgenden Definitionen auf verschiedene Basisfunktionen in Bäumen Bezug genommen. In Tabelle 4.1 hatten wir diese Funktion in ihrer Struktur und Semantik bereits aufgeführt.

Definition 4.3 (Blätter der Auswahlebene): *Sei $T = (N, E, ...)$ ein HSP-Baum und $\sigma^n = (N', E', ...)$ eine Auswahlebene von T. Als Blätter der Auswahlebene σ^n wird die Knotenmenge $B \subseteq N$ und $B \cap N' = \emptyset$ bezeichnet, für die gilt:*

$$\forall b \in B : \exists n' \in N' : pred^b = n'$$

Die Funktion $\sigma_{leaf} : N \mapsto 2^N$ mit σ^n_{leaf} bestimmt die Blätter der Auswahlebene σ^n.

Als Blätter einer Auswahlebene σ^n werden alle Knoten bezeichnet, deren unmittelbare Vorgänger (Funktion $pred^n$ liefert den Vorgänger eines Knotens n) der Auswahlebene σ^n angehören. Der konkrete Typ dieser Knoten spielt hier jedoch keine Rolle. So können neben H-Knoten weitere Auswahlebenen oder Projektionsebenen folgen (siehe "gekoppelte Auswahl" unten). Allerdings müssen in Abhängigkeit vom Knotentyp bestimmte Regeln für die Schachtelungstiefe gelten, die die folgende Strukturregel festlegt:

Strukturregel SR-4 (Blätter der Auswahlebene):
Sei $T = (N, E, ..)$ ein HSP-Baum und Σ die Menge aller Auswahlebenen von T. Sei weiterhin $\sigma^n \in \Sigma$ eine Auswahlebene für die gilt: $\exists \sigma^k \in \Sigma : n \in \sigma^k$.
Wir fordern stets, dass für jedes Blatt $b \in \sigma^n_{leaf}$ der Auswahlebene σ^n gilt:

$$\begin{aligned} NT^b \in \{\texttt{HNODE}, \texttt{ONODE}\} \quad &\Rightarrow \quad ND^b = ND^k - 1 \\ NT^b \in \{\texttt{SNODE}, \texttt{PNODE}\} \quad &\Rightarrow \quad ND^b = ND^k \end{aligned}$$

In der obigen Strukturregel wenden wir einen formalen "Trick" an, um die Schachtelungstiefe für alle vorkommenden Fälle bei Auswahlebenen abzudecken (siehe Abbildung 4.14). Das heißt, wir gehen von einer Auswahlebene σ^k aus, die die Auswahlebene σ^n beinhaltet. Auf dem ersten Blick entspricht dies dem Fall der geschachtelten Auswahlebene (siehe Abbildung 4.14c). Allerdings können die beiden Knoten k und n gleich sein. In diesem Fall, wird durch die obige Regel auch der Fall der einfachen Auswahlebene abgedeckt. Mit der obigen Strukturregel wird Folgendes festgelegt:

- Beim Übergang von einer Auswahlebene auf seine Blätter muss die Schachtelungstiefe entsprechend verringert werden. Diese Verringerung muss jedoch nicht immer um den Faktor 1 erfolgen. Im Fall von geschachtelten Auswahlebenen kann dieser Faktor größer ausfallen. Allerdings existiert stets eine Auswahlebene, in der der Knoten b ein Blatt darstellt und der Faktor lediglich eine Verringerung um den Faktor 1 aufweist. In Abbildung 4.15 ist ein entsprechendes Beispiel dargestellt. Der Faktor beträgt hier den Wert 2 bei Auswahlebene s_2 und den Knoten v_2''' und v_2''''. Bezüglich der Ebene s_1 weist der Faktor jedoch wieder den Wert 1 auf.
- Falls der Blattknoten bzw. der Knoten b ein S-Knoten oder P-Knoten ist, muss die Schachtelungstiefe gleich bleiben. Es handelt sich hier um den Fall einer gekoppelten Auswahl- oder Projektionsebene. Diese Ebene wird zu der vorherigen Ebene auf gleicher Schachtelungstiefe verstanden. Alle anderen Fälle, d.h. einfache und geschachtelte Auswahlebenen sind bereits im ersten Fall (siehe oben) enthalten.

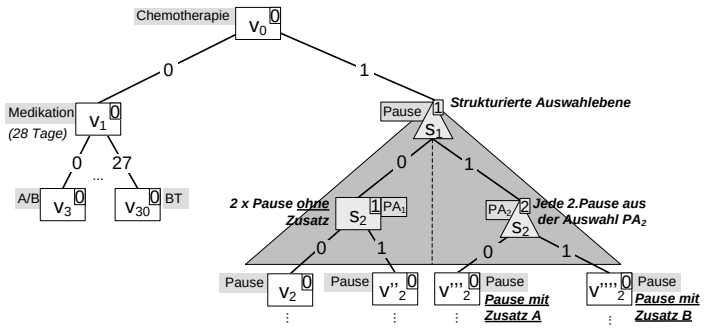
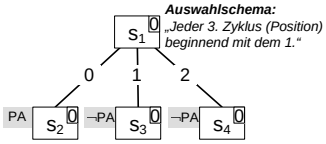


Abbildung 4.15: Beispiel eines HSP-Baums mit Faktor $\neq 1$ beim Wechsel der Schachtelungstiefe

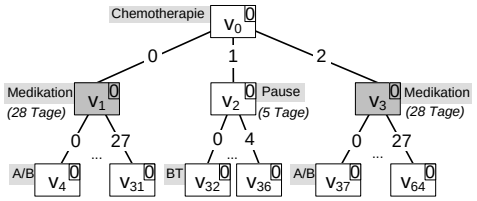
4.2.2.3 Abbildung von mehrelementigen Auswahlmengen

In den vorherigen Betrachtungen zu Zyklus- und Zykluspositionsauswahloperationen wurde stets von dem Fall ausgegangen, dass die Auswahlmenge einer Auswahloperation im HSP-Baum durch einen einzelnen Knoten repräsentiert wird. Im Folgenden steht der Fall im Fokus, dass die Auswahlmenge einer (endlichen) Menge von Knoten entspricht. Ein solches Beispiel stellt etwa eine Chemotherapie dar, bestehend aus drei Teilzyklen, wobei der erste und letzte Zyklus einer Medikationsphase (`Medikation`) entspricht und der mittlere einem Zyklus `Pause`. Das Beispiel ist in Abbildung 4.16 dargestellt. Der HSP-Baum des Auswahlschemas ist in Abbildung 4.16a dargestellt. Der HSP-Baum des Periodizitätsschemas mit der entsprechenden Auswahlmenge ist in Abbildung 4.16b dargestellt. In diesem Beispiel führt eine Zyklusauswahl mit der Semantik "*In jedem 3. Zyklus* `Medikation` *(erfolgt eine unterschiedliche Dosierung von Medikamenten)*" zu einer Auswahlmenge bestehend aus den beiden Zyklen `Medikation` bzw. im HSP-Baum zu den beiden korrespondierenden Knoten v_1 und v_3.

Vergleichbar zum vorherigen Abschnitt wird auch im Fall mengenwertiger Auswahlmengen für jeden der Knoten aus der Menge eine Auswahlebene basierend auf dem



(a) Äquidistantes Auswahlschema



(b) Periodizitätsschema mit Auswahlmenge $\{v_1, v_3\}$

Abbildung 4.16: Beispiel einer Auswahloperation mit mengenwertiger Auswahlmenge

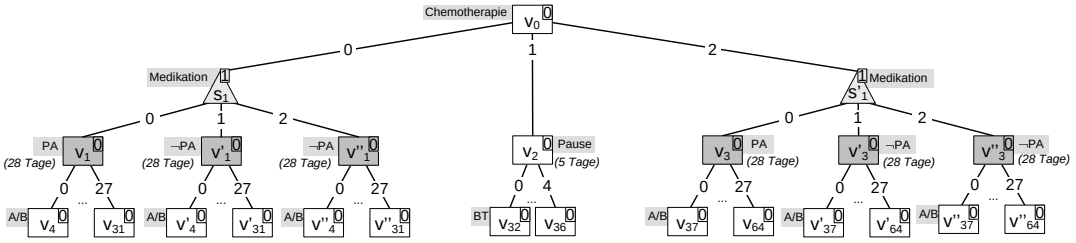


Abbildung 4.17: Auswahlebenen im HSP-Baum bei mengenwertiger Auswahlmenge

Auswahlschema eingefügt. In Abbildung 4.17 ist für das obige Beispiel für jeden Knoten der Auswahlmenge eine entsprechende Auswahlebene eingefügt worden.

Allerdings reicht das Einfügen der Auswahlebenen alleine nicht aus. Dies würde nämlich dazu führen, dass bei der Auswertung des Baumes die eingefügten Auswahlebenen unabhängig voneinander behandelt werden. Das heißt bei dem ersten Vorkommen des Zyklus `Medikation` würde der Knoten s_1 mit Kantenindex = 0 ausgewählt. Analog würde bei dem ersten Vorkommen bzw. Wiederholung des zweiten Zyklus `Medikation` bzw. Knoten s'_1 ebenfalls der Knoten mit Kantenindex 0 ausgewählt. Allerdings wäre dieses Verhalten falsch. Da es sich bei dem zweiten Zyklus bereits um das zweite Vorkommen des Zyklus `Medikation` handelt, muss bei dem ersten Vorkommen von Knoten s'_1 bereits der Knoten mit Kantenindex = 1 ausgewählt werden. Beim anschließenden, erneuten Vorkommen des Zyklus `Medikation` (Knoten s_1) entspricht dies dann bereits dem dritten Vorkommen des Zyklus `Medikation` und es muss der Knoten mit Kantenindex = 3 ausgewählt werden und so weiter. Dieser Zusammenhang ist in Abbildung 4.18 anhand des Wiederholungsmusters des Auswahlschemas über der Auswahlmenge dargestellt. Im unteren Bereich ist die korrekte, vollständige Reihenfolge der Blätter der Auswahlebenen bei entsprechender Auswertung des HSP-Baums dargestellt.

Um die korrekte Reihenfolge der Blätter sicherzustellen, müssen die beiden Auswahlebenen s_1 und s'_1 miteinander "verknüpft" bzw. abhängig voneinander ausgewertet werden. Im HSP-Baum wird dies durch die Auswahlparameter (siehe Definition 4.1, Funktion $SP()$) sichergestellt, die jeder Auswahlebene zugewiesen sein müssen. So wird zwar für jeden Knoten einer Auswahlmenge eine separate Auswahlebene im HSP-Baum erzeugt, über die Auswahlparameter können diese jedoch miteinander verknüpft werden. Bei der Auswertung werden die so verknüpften Auswahlebenen virtuell dann wie eine einzelne Auswahlebene ausgewertet, was zu der oben dargestellten Auswertung von *abhängigen Auswahlebenen* führt. Die Semantik, Auswertung und Erzeugung der Auswahlparameter werden erst in Kapitel 5 vorgestellt.

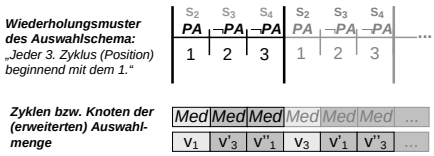


Abbildung 4.18: Zyklenorientierte Darstellung des Wiederholungsmusters des Auswahlschemas über der Auswahlmenge

4.2.3 P-Knoten

Im HSP-Baum dienen Knoten vom Typ `PNODE` bzw. P-Knoten (**p**rojection **n**ode) der Abbildung des Ergebnis der Auswahloperation *Projektionsauswahl.*

4.2.3.1 Vorbetrachtungen zur Abbildung der Projektionsauswahl im HSP-Baum

Eine Projektionsauswahl ermöglicht es, zwei voneinander unabhängige Periodizitätsschemas miteinander in Beziehung zu setzen. Nur die Zykluspositionen bzw. Labels eines Periodizitätsschemas PS, die gleichzeitig in einem anderen Periodizitätsschema PS_2 mit bestimmten Positionen bzw. Labels korrespondieren, werden bei einer Projektionsauswahl ausgewählt. Das Periodizitätsschema PS_2 kann hierbei als Auswahlschema verstanden werden, das über der Auswahlmenge interpretiert wird. Im Unterschied zu den beiden anderen Auswahloperationen entspricht bei einer Projektionsauswahl die Auswahlmenge stets der vollständigen Menge an Zykluspositionen von Schema PS. In der folgenden Abbildung 4.19 ist eine Projektionsauswahl zwischen dem Schema “*Jeden Montag und Mittwoch in einer Woche*” (PS_2) und einem Periodizitätsschema (PS) mit einem Zyklus der Länge 3 und Aktivität A am zweiten Tag dargestellt. In dem Beispiel werden alle Aktivitäten `A` im Schema PS ausgewählt, die gleichzeitig auch auf einen Montag (`Mo`) oder Mittwoch (`Mi`) im Schema PS_2 fallen. Somit enthält die Projektionsmenge von PS alle Zykluspositionen mit Label `A`. Für beide Periodizitätsschemas wurde der gleiche Instanziierungspunkt = 1 angenommen.

Die Abbildung der Projektionsauswahl erfolgt im Wesentlichen analog zu den bereits beschriebenen Auswahloperationen. Auch hier wird das Auswahlschema bzw. das Periodizitätsschema PS_2 als Teilbaum in den HSP-Baum des Periodizitätsschema PS eingefügt. Beispielsweise ist der resultierende HSP-Baum für das obige Beispiel einer Projektionsauswahl in Abbildung 4.20 dargestellt. In Abbildung 4.20a ist der HSP-Baum des Periodizitätsschema PS mit der Projektionsmenge v_1 dargestellt. Analog zeigt Abbildung 4.20b den HSP-Baum des Periodizitätsschema PS_2. Das resultierende Periodizitätsschema PS^* bzw. der entsprechende HSP-Baum mit Projektionsebenen ist in Abbildung 4.20c dargestellt.

Durch das Einfügen des Teilbaums entsteht in dem resultierenden HSP-Baum eine sogenannte “Projektionsebene”, die die Struktur des Periodizitätsschemas PS_2 repräsentiert.

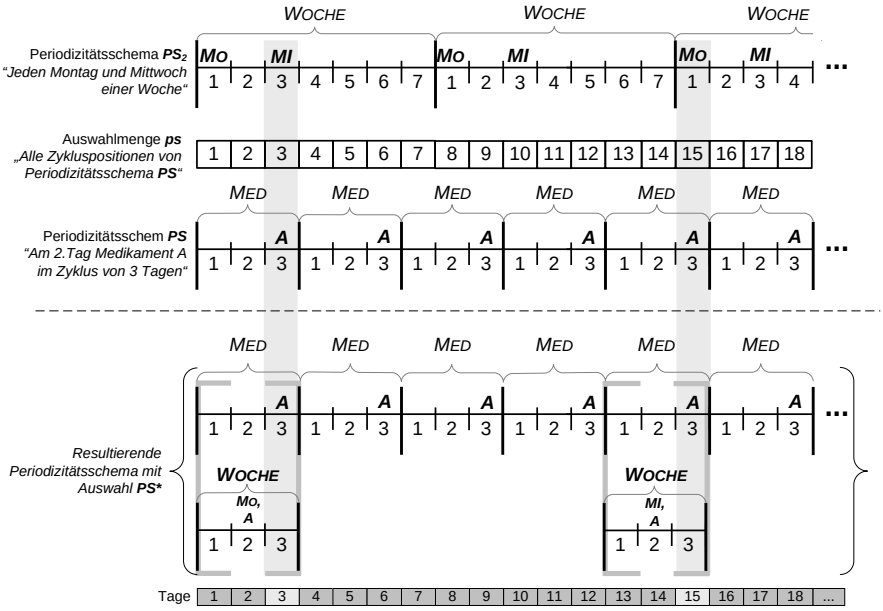
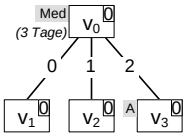
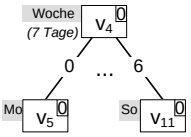


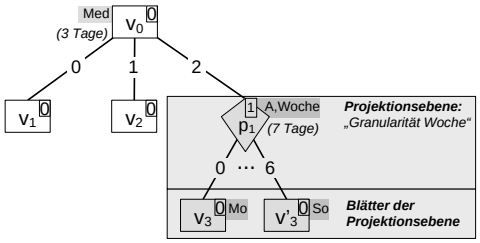
Abbildung 4.19: Beispiel einer Projektionsauswahl im zyklenorientierten Modell



(a) Periodizitätsschema PS mit Projektionsmenge $\{v_3\}$



(b) Periodizitätsschema PS_2



(c) Resultierender HSP-Baum mit Projektionsebene p_1

Abbildung 4.20: Beispiel für die Abbildung einer Projektionsauswahl im HSP-Baum

Jede Projektionsebene wird stets durch einen P-Knoten eingeleitet bzw. durch diesen explizit gekennzeichnet. Da es sich im vorliegenden Beispiel lediglich um ein einfaches, äquidistantes Periodizitätsschema PS_2 handelt, besteht die Projektionsebene lediglich aus dem einzelnen P-Knoten p_1.

Dieser Form der Abbildung der Projektionsauswahl im HSP-Baum liegt ein fundamentaler Zusammenhang bei dem Verhalten zweier Periodizitätsschemas bzw. deren Wiederholungsmuster zugrunde. Dieser Zusammenhang besagt, dass bei einer Projektion jede Zyklusposition des Wiederholungsmusters von Schema PS genau einmal mit jeder Zyklusposition des Wiederholungsmusters von Schema PS_2 zusammenfällt (bzw. maximal einmal in einem Ausnahmefall). Einfacher ausgedrückt, jede einzelne Zyklusposition von PS "durchläuft" das gesamte Periodizitätsschema PS_2 stets in einer bestimmten Reihenfolge, ohne hierbei Zyklusspositionen von PS_2 mehrfach zu "berühren". Diese Reihenfolge von "Berührungen" kann auf Basis des fundamentalen Zusammenhangs berechnet werden, d.h. es kann für jedes Vorkommen einer Zyklusposition von Schema PS angegeben werden, mit welcher Zyklusposition im Schema PS_2 diese korrespondiert. Allerdings wird im HSP-Baum durch eine Projektionsebene nicht das Wiederholungsmuster der "Berührungen" repräsentiert. Stattdessen wird das ursprüngliche Periodizitätsschema PS_2 eingefügt[7]. Die Reihenfolge der Blätter der resultierenden Projektionsebene entspricht somit nicht der Reihenfolge der "Berührungen" mit der entsprechenden Zyklusposition im Schema PS. Erst bei der Auswertung des HSP-Baums bzw. der Projektionsebene wird berechnet, mit welchem Blatt der entsprechenden Projektionsebene das Vorkommen einer Zyklusposition korrespondiert. In Bezug auf das obige Beispiel bedeutet dies, dass die Blätter der Projektionsebene in der folgenden Reihenfolge bezüglich der Kantenindizes durchlaufen werden: $2-5-1-4-0-3-6$. Das bedeutet, dass das erste Vorkommen von Zyklusposition A in Schema PS mit der Zyklusposition 1 (Kantenindex = 0) im Schema PS_2 korrespondiert. Das zweite Vorkommen von A mit der Position 4 (Kantenindex = 3) in PS_2 und so weiter. Auf Basis einer solchen Auswertung kann das resultierende Periodizitätsschema PS^* abgeleitet werden (siehe unterer Bereich in Abbildung 4.19). Eine detailierte Vorstellung und Beschreibung des fundamentalen Zusammenhangs erfolgt später in Kapitel 5.

4.2.3.2 Projektionsebene

Jeder P-Knoten im HSP-Baum stellt den Beginn einer Projektionsebene dar bzw. leitet eine solche Ebene ein. Die Struktur des Periodizitätsschema PS_2 wird durch die Pro-

[7] Grundsätzlich kann ein Wiederholungsmuster für die Reihenfolge der "Berührungen" einer Zyklusposition aus Schema PS mit den Zyklusspositionen in Schema PS_2 angegeben bzw. als HSP-Baum repräsentiert werden. Die Basis hierfür würde das Schema PS_2 bilden. Die Reihenfolge der Blätter der resultierenden Projektionsebene würde dann mit der korrekten Reihenfolge der "Berührungen" übereinstimmen. Allerdings würde bei der Erzeugung eines solchen Baums bzw. Projektionsebene die ursprüngliche Struktur des Periodizitätsschemas PS_2 verloren gehen. In Hinblick auf Labelausdrücke bzw. Positionsangaben würde sich der Verlust der Struktur des Schemas PS_2 jedoch als hinderlich erweisen, weswegen im HSP-Baum ein andere Abbildungsform verwendet wird.

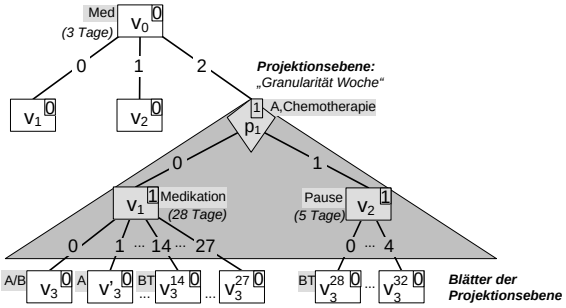


Abbildung 4.21: Beispiel einer Projektion mit strukturierten Periodizitätsschema PS_2

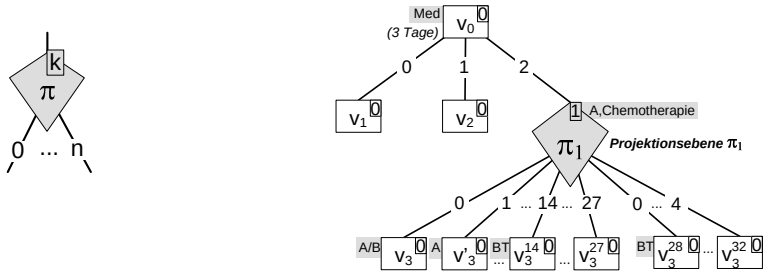
jektionsebene im HSP-Baum von Schema PS abgebildet. Die Abbildung strukturierter Periodizitätsschemas bzw. die Übertragung der Struktur auf die entsprechende Projektionsebene erfolgt analog wie bei Auswahlebenen, d.h. auf Basis von H-Knoten und der Schachtelungstiefe ND (siehe Abschnitt 4.2.2.2). In Abbildung 4.21 ist ein Beispiel für eine Projektion mit strukturiertem Projektionsschema dargestellt. In dem Beispiel handelt es sich, um eine Projektion des Periodizitätsschemas PS aus Abbildung 4.20a auf das Periodizitätsschema einer Chemotherapie. Formal ist eine Projektionsebene wie folgt definiert:

Definition 4.4 (Projektionsebene): *Sei $T = (N, E, ...)$ ein HSP-Baum und $n \in N$ ein P-Knoten. Der von Knoten n erreichbare HSP-Teilbaum $T' = (N', E', ...)$ wird als Projektionsebene π^n bezeichnet, falls $\forall n' \in N', n' \neq n$ gilt:*

$$
\begin{array}{lcl}
NT^{n'} \in \{\texttt{HNODE}, \texttt{ONODE}\} & \Rightarrow & ND^{n'} = ND^n \\
NT^{n'} = \texttt{SNODE} & \Rightarrow & (ND^{n'} = ND^n + 1) \wedge \sigma^n \textit{ ist eine Auswahlebebene} \\
NT^{n'} = \texttt{PNODE} & \Rightarrow & (ND^{n'} = ND^n + 1) \wedge \pi^n \textit{ ist eine Projektionsebene}
\end{array}
$$

Im Unterschied zu Auswahlebenen können Projektionsebenen sowohl Auswahlebenen als auch weitere Projektionsebenen beinhalten. Da die bei der Projektion verwendeten Schemas “klassischen” Periodizitätsschemas entsprechen, können diese ebenfalls bereits Projektions- bzw. Auswahlebenen und entsprechende Strukturierungen beinhalten.

Zur Sicherstellung der korrekten Abbildung von Projektionsauswahlen im HSP-Baum fordern wir, vergleichbar zu Auswahlebenen, dass gilt:



(a) Vereinfachte Darstellung (b) Beispiel einer vereinfachten Darstellung

Abbildung 4.22: Vereinfachte Darstellung von Projektionsebenen

Strukturregel SR-5 (Projektionsebene):
Sei $T = (N, E, ..)$ ein HSP-Baum und Π die Menge aller Projektionsebenen von T. Wir fordern stets, dass gilt:

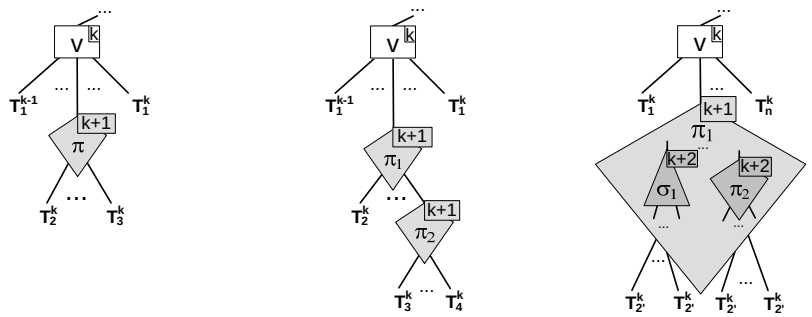
1. Vollständigkeit:
   $\nexists n \in N : NT^n = \texttt{PNODE} \wedge \nexists \pi^n \in \Pi.$
2. Abgeschlossenheit:
   $\forall \pi^n = (N', E', ...) \in \Pi$: $\forall n' \in N' : \forall k \in succ^{n'} : k \in N' \vee k \in \pi^n_{leaf}$
3. Jeder Projektionsebene ist ein Tupel an Projektionsparametern zugeordnet:
   $\forall \sigma^n \in \Sigma : SP^n \neq \emptyset$

In der Strukturregel werden die analogen Bedingungen an Projektionsebenen gestellt, wie sie auch für Auswahlebenen gelten müssen. Wir verzichten daher auf eine nähere Erläuterung und verweisen stattdessen auf die Erläuterung zu Strukturregel SR-3 auf Seite 101, die leicht übertragen werden können. Ebenfalls analog zu Auswahlebenen führen wir für Projektionsebenen ebenfalls eine vereinfachte Darstellung ein, die Abbildung 4.22 zeigt.

Eine Projektionsauswahl kann mehrfach auf einem Periodizitätsschema angewendet werden. Infolgedessen können auch beliebig viele Projektionsebenen (mit zum Teil unterschiedlichen Instanziierungspunkten) in einem HSP-Baum enthalten sein. Grundformen von Projektionsebenen zeigt Abbildung 4.23, die in ihrer Struktur mit den Grundformen von Auswahlebenen vergleichbar sind (siehe Abbildung 103).

Einfache Projektionsebene: Eine einfache Projektion entspricht einer einmaligen Anwendung einer Projektionsoperation in einem Periodizitätsschema. Alle vorherigen Beispiele gehören zu dieser Grundform, die Abbildung 112 zeigt.

Gekoppelte Projektionsebene: Eine gekoppelte Projektion beschreibt die mehrma-



(a) Einfache Projektion (b) Gekoppelte Projektion (c) Geschachtelte Projektion und Auswahl

Abbildung 4.23: Verschiedene Grundformen von Projektionsebenen

lige Anwendung einer Projektion, wobei für ein bzw. mehrere Blätter der ersten Projektionsebene π_1 erneut eine Projektion vorgenommen wird. Schematisch zeigt dies Abbildung 112. Ein konkretes Beispiel hierfür bildet etwa die Abstimmung zweier voneinander unabhängiger (periodischer) Therapien, die zusätzlich auf den (periodischen) Dienstplan eines Arztes abgestimmt werden. Mit der ersten Projektion π_1 werden die beiden Therapien miteinander in Beziehung gesetzt und etwa identische Medikamentengabe am gleichen Tag in ihrer Dosierung angepasst. Mit der zweiten Projektion π_2 können diese Therapien bzw. bestimmte Medikamentengaben dann mit dem Dienstplan des Arztes abgestimmt werden.

Geschachtelte Projektionsebene: Eine geschachtelte Projektion entspricht einer Auswahl und/oder Projektion innerhalb einer Projektionsebene, wie in Abbildung 112 schematisch gezeigt. Ein Beispiel hierfür bildet etwa eine Projektion zwischen der Therapieangabe "*Jeder 3.Tag*" und dem Periodizitätsschema der Granularität "Monate" (auf Basis von Tagen). So können etwa alle Tage der Therapie adressiert werden, die mit den dienstfreien Tagen des Arztes an "*jedem letzten Mittwoch im Monat"* zusammenfallen. Das Periodizitätsschema der Granularität "Monate*"* bzw. der entsprechende HSP-Baum kann sowohl eine Auswahlebene als auch Projektionsebene enthalten. Aufgrund der Schaltjahresproblematik ergibt sich eine Auswahlebene für den Monat "Februar" (siehe Abschnitt 4.2.2.2). Zusätzlich besteht durch die Umschaltung der Sommer-/Winterzeit eine Projektionsebene in Bezug auf die letzten Sonntage im März und Oktober.

4.2.3.3 Blätter der Projektionsebene

Analog zu Auswahlebenen werden die Nachfolgerknoten einer Projektionsebene als *Blätter der Projektionsebene* bezeichnet. Diese entsprechen der Menge an Positionen des Ausgangsschemas, die von der Projektion betroffen sind. In der Abbildung 4.21 bilden die Knoten $v_3, ..., v_3^{32}$ die Blätter der Projektionsebene p_1. Formal sind diese wie folgt

definiert:

Definition 4.5 (Blätter der Projektionsebene): *Sei $T = (N, E, ...)$ ein HSP-Baum und $\pi^n = (N', E', ...)$ eine Projektionsebene von T. Als Blätter der Projektionsebene π^n wird die Knotenmenge $B \subseteq N$ und $B \cap N' = \emptyset$ bezeichnet, für die gilt:*

$$\forall b \in B : \exists n' \in N' : pred^b = n'$$

Die Funktion $\pi_{leaf} : N \mapsto N^*$ mit π^n_{leaf} bestimmt die Blätter der Projektionsebene π^n.

Wie bei Auswahlebenen sind mit dem konkreten Knotentyp eines Blattes einer Projektionsebene bestimmte Bedingungen für die Schachtelungstiefe verknüpft. Diese werden in der nachfolgenden Strukturregel festgelegt.

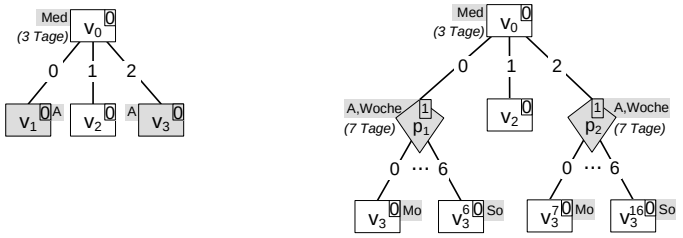
Strukturregel SR-6 (Blätter der Projektionsebene):
Sei $T = (N, E, ..)$ ein HSP-Baum, Π die Menge aller Projektionsebenen und Σ die Menge aller Auswahlebenen von T. Sei weiterhin $\pi^n \in \Sigma$ eine Projektionsebene für die gilt: $\exists \pi^k \in \pi : n \in \pi^k$. Wir fordern stets, dass für jedes Blatt $b \in \pi^n_{leaf}$ der Auswahlebene π^n gilt:

$$\begin{array}{lcl} NT^b \in \{\texttt{HNODE}, \texttt{ONODE}\} & \Rightarrow & ND^b = ND^k - 1 \\ NT^b \in \{\texttt{SNODE}, \texttt{PNODE}\} & \Rightarrow & ND^b = ND^k \end{array}$$

Auch die obige Strukturregel für die Blätter der Projektionsebene ist analog zur Strukturregel SR-4 auf Seite 104 für die Blätter von Auswahlebenen. Entsprechende Erläuterungen zu dieser können leicht auf die vorliegende Strukturregel übertragen werden, sodass an dieser Stelle auf weitere Erläuterungen verzichtet wird.

4.2.3.4 Abbildung bei mehrelementiger Auswahlmenge

Im Gegensatz zu den vorherigen Auswahloperationen (siehe Abschnitt 4.2.2.3) muss bei einer Projektionsauswahl nicht zwischen Projektionsmenge mit einem einzelnen Knoten und einer Menge von Knoten unterschieden werden. Beispielsweise ist in der Abbildung 4.24 das gleiche Beispiel wie in Abschnitt 4.2.2.3 dargestellt. Es handelt sich um eine Chemotherapie, bestehend aus drei Teilzyklen, wobei der erste und letzte Zyklus einer Medikationsphase (`Medikation`) entspricht und der mittlere einem Zyklus `Pause`. Die Projektion erfolgt hier auf der ersten Gabe von Medikament `A` im Zyklus `Medikation` und führt zu der Projektionsmenge $\{v_1, v_3\}$. Für jeden der Knoten wird durch eine Projektionsauswahl eine entsprechende Projektionsebene eingefügt. Bei einer Auswertung können allerdings beide Projektionsebenen unabhängig voneinander ausgewertet werden. Wie wir später noch zeigen werden, ist alleine die Zyklusposition der Knoten der Projektionsmenge im Ausgangschema ausreichend für die richtige Auswertung der jeweils eingefügten Projektionsebene. Im vorliegenden Beispiel ergibt sich für die Blätter der Ebene π_1 die (Kantenindex) Reihenfolge $0-3-6-2-5-1-4$ und für die Ebene π_2 die Reihenfolge $2-5-1-4-0-3-6$.



(a) Ausgangsschema mit Projektionsmenge $\{v_1, v_3\}$

(b) Periodizitätsschema mit mehrfachen Projektionsebenen

Abbildung 4.24: Projektionsauswahl mit mengenwertiger Projektionsmenge

Abschließend zu Projektionsebenen gilt es noch festzuhalten, dass im formalen Modell eine Projektionsebene an beliebiger Stelle bzw. Baumebene im HSP-Baum vorkommen darf. Aus semantischer Sicht muss jedoch beachtet werden, dass eine Projektion nur zwischen solchen Periodizitätsschemas als semantisch sinnvoll erachtet werden kann, bei denen die Zykluspositionen die gleiche Granularität aufweisen. Beispielsweise ist die Projektionsauswahl der Angabe "*Jeden 3.Tag*" mit der Angabe "*Jede 3.Stunde"* semantisch nicht sinnvoll, da die Zykluspositionen einmal einem einzelnen Tag und einmal einer einzelnen Stunde entsprechen. Allerdings wird in der später vorgestellten Algebra diese semantische Korrektheit nicht explizit überprüft. Die Sicherstellung der semantischen Korrektheit wird an dieser Stelle dem Anwender überlassen und kann nicht im HSP-Baum überprüft werden. Wie wir später in Abschnitt 4.5 noch zeigen werden, ist einer der Vorteile eines expliziten Granularitätshierarchiemodell, dass so auch die semantische Korrektheit bei Projektionen überprüft werden kann.

4.2.4 0-Knoten

Die Knoten vom Typ `ONODE` bzw. 0-Knoten bilden das Null-Element des HSP-Baums. Ein solches Nullelement wird benötigt, um das (periodische) Löschen von Zyklen aus einer Auswahlmenge abzubilden. Beispielsweise ist der HSP-Baum des Beispiels "*Jede 3. Pause einer Chemotherapie (beginnend bei der letzten) entfällt*" in Abbildung 4.25 dargestellt. Bei dem Löschen einer "*jeden 3. Pause*" kann der entsprechende Knoten der Auswahlebene s_3, d.h. der Knoten mit Kantenindex = 2, nicht einfach gelöscht werden. Ein solches Löschen würde lediglich die Semantik der Auswahlebene bzw. des Auswahlschemas von "*Jede 3.Pause"* auf "*Jede 2.Pause"* verändern. Stattdessen muss der Zweig mit Kantenindex=2 bestehen bleiben und es darf lediglich der Zyklus bei der Auswertung nicht berücksichtigt werden. Diese "Nullzyklen" d.h. die Menge der gelöschten Pausen, wird durch einen 0-Knoten repräsentiert. Bei der Auswertung von HSP-Bäumen werden 0-Knoten nicht in ein entsprechendes Wiederholungsmuster übertragen. Im obigen Beispiel etwa ergibt sich die folgende Reihenfolge von Zyklen $Med(v_1) - Pause(v_2) - Med(v_1) - Pause(v_4) - Med(v_1) - -Med(v_1) - Pause(v_2)$....

Diese Reihenfolge kann ohne Verwendung von 0-Knoten lediglich durch eine explizite Modellierung der Zyklenreihenfolge als H-Knoten repräsentiert werden.

Durch ihre Semantik dürfen 0-Knoten nicht beliebig im HSP-Baum vorkommen. Stattdessen sind diese nur unterhalb von Auswahl- oder Projektionsebenen zulässig, d.h. innerhalb der Blätter solcher Ebenen. In den anderen Fällen, d.h. etwa unterhalb von einem H-Knoten wird kein 0-Knoten benötigt, da der Originalknoten einfach gelöscht werden könnte. Wir fordern daher stets, dass Folgendes gilt:

Strukturregel SR-7 (0-Knoten):
Sei $T = (N, E, ..)$ ein HSP-Baum und θ die Menge aller 0-Knoten für die gilt: $\nexists n \in N : NT^n = \texttt{ONODE} \wedge \texttt{n} \notin \theta$. Des Weiteren sei Π die Menge aller Projektionsebenen und Σ die Menge aller Auswahlebenen von T. Wir fordern stets, dass gilt: $\forall n \in \theta$:

1. 0-Knoten kommen nur in den Blätter von Auswahl- oder Projektionsebenen vor: $\exists m \in N : n \in \pi^m_{leaf} \vee n \in \sigma^m_{leaf}$
2. 0-Knoten haben keinen Nachfolger: $SUCC^n = \emptyset$

Diese Strukturregel besagt, dass 0-Knoten nicht "frei" innerhalb einer Auswahl- bzw. Projektionsebene vorkommen können. Stattdessen muss innerhalb der entsprechenden Auswahl- oder Projektionsebene eine weitere geschachtelte Auswahl- oder Projektionsebene bestehen. Dies hängt damit zusammen, dass wenn eine Auswahlebene eine solche weitere Ebene nicht aufweisen würde, dass zugehörige Auswahlschema lediglich aus H-Knoten bestanden hat. Ein 0-Knoten ist in dem entsprechenden Baum des Auswahlschemas jedoch nicht notwendig, da die Semantik von 0-Knoten durch Löschen des entsprechenden H-Knotens abgebildet werden könnte.

Allerdings können aufgrund der obigen Regel 0-Knoten auf Ebene der Zyklupositionen auftreten. Die Auswertung eines solchen HSP-Baums würde semantisch dazu führen, dass der entsprechende Zyklus der Zyklusposition in seiner Länge verkürzt wird. Semantisch entspricht dies somit dem Verkürzen von Zyklen. Die Auswertung des Baumes würde somit ebenfalls zu semantisch korrekten Ergebnissen führen. Allerdings weist die

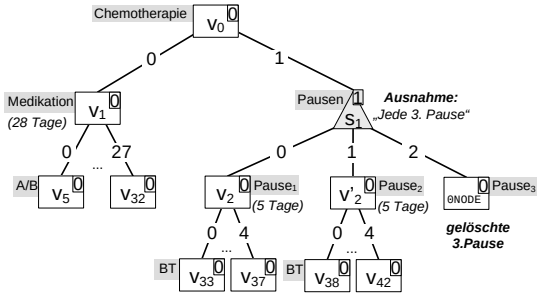


Abbildung 4.25: Beispiel eines HSP-Baums mit 0-Knoten

später noch vorgestellte Zyklusalgebra keine Operation auf, die ein Löschen einer Auswahl von Zykluspositionen ermöglicht. Aus diesem Grund ist der Fall, dass 0-Knoten auf Ebene von Zykluspositionen auftauchen, nur theoretisch innerhalb eines HSP-Baums möglich.

4.2.5 Auswahl- und Projektionsparameter

Die grundsätzliche Semantik von Auswahl- und Projektionsparametern steht im engen Zusammenhang mit nicht-auswahlerhaltenden Operationsreihenfolgen der algebraischen Notationsform des zyklenorientierten Zeitmodells (siehe Kapitel 3, Abschnitt 3.4.3). Bei nicht-auswahlerhaltenden Operationsreihenfolgen wird eine Auswahl nachträglich durch entsprechende Operationen beeinflusst. Beispielsweise tritt dies auf, wenn nach einer erfolgten Projektionsauswahl zwischen Schema P_1 und P_2 die Länge eines Zyklus in Schema P_1 verändert wird. Aufgrund der Zyklusänderung verschiebt sich jedoch auch das Verhältnis zwischen den Schemas P_1 und P_2 bezüglich einer Projektion. Das heißt, die vor der Änderung der Zykluslänge erfolgte Projektion führt zu anderen Ergebnissen (gemeinsamen Positionen zwischen P_1 und P_2) als eine Projektion, die nach der Zykluslängenänderung stattgefunden hätte. Wie im vorherigen Kapitel beschrieben, wird im zyklenorientierten Zeitmodell in solchen Fällen stets das Ergebnis der Projektion zum Zeitpunkt der Projektionsauswahl beibehalten. Eine "nachträgliche" Änderung der Projektion aufgrund einer nicht-auswahlerhaltenden Operationsreihenfolge wird nicht vorgenommen (siehe Abschnitt 3.4.3.3). Ein solches Verhalten des formalen Modells führt jedoch dazu, dass die vorliegende Situation in einem HSP-Baum zum Zeitpunkt der Auswertung eigentlich irrelevant ist. Stattdessen gilt es im HSP-Baum die Situation zum Zeitpunkt der Auswahloperation "einzufrieren". Zum Zeitpunkt der Auswertung muss dann auf die "eingefrorene" Situation, d.h. die Situation zum Zeitpunkt der Auswahloperation, Bezug genommen werden. Für das vorherige Beispiel übertragen auf einen HSP-Baum bedeutet dies, dass die Auswertung so erfolgen muss, als ob die nachträgliche Änderung der Zykluslänge und damit verbundene Beeinflussung der Projektionsebene nicht stattgefunden hat.

Die Auswahl- bzw. Projektionsparameter, die jeder entsprechenden Ebene im HSP-Baum zugeordnet sind, dienen dazu, die Situation zum Zeitpunkt der Auswahloperation "einzufrieren" bzw. festzuhalten. Bei der Auswertung entsprechender Ebenen werden dann nur die Werte der Parameter berücksichtigt und nicht die zum Auswertungszeitpunkt vorliegende Situation im HSP-Baum. Mit der Funktion $SP() \mapsto SelParam := (z, l_P, l_A, Pos)$ (siehe Definition 4.1) können die entsprechenden Parameter für jede Auswahlebenen bzw. für den einleitenden S-Knoten oder P-Knoten bestimmt werden. Wie wir später zeigen werden, sind diese vier Parameter ausreichend, um die zum Zeitpunkt der Auswahloperation vorliegende Situation im HSP-Baum für jede Ebene zu beschreiben. Zum Verständnis der genauen Semantik der Parameter sind jedoch tiefergehende theoretische Betrachtungen notwendig. Die detaillierte Vorstellung ihrer Semantik und Eigenschaften erfolgt in Kapitel 5.3.

4.2.6 Korrektheit

Die bisherigen Strukturregeln haben für jeden einzelnen Knotentypen die korrekte, grundsätzliche Verwendung dieser Strukturen im HSP-Baum formal festgelegt. Auf Basis der in den vorherigen Abschnitten eingeführten Strukturregeln ist die Korrektheit eines HSP-Baums wie folgt definiert:

Definition 4.6 (Korrektheit): *Ein HSP-Baum T wird als korrekter HSP-Baum bezeichnet, wenn die Strukturregeln SR-1 – SR-7 gelten.*

Diese Definition von Korrektheit bildet die Grundlage für die im Folgenden noch vorgestellten Algorithmen zur Auswertung bzw. Operationen zur Bildung von HSP-Bäumen, d.h. diese setzen korrekte HSP-Bäume voraus und führen wieder zu korrekten HSP-Bäumen.

4.3 Realisierung der algebraischen Notationsform

Eine verständliche und nachvollziehbare Notationsform zur Beschreibung von Periodizitätsschema ist essentiell im Zusammenhang mit der Anforderung nach Verständlichkeit. Insbesondere die Unterstützung von Ausnahmen und die damit verbundene Forderung nach einer expliziten Spezifikationslogik ist wesentlich. In Kapitel 3 wurde das zyklenorientierte Zeitmodell sowie eine algebraische Notationsform eingeführt, die den Anforderungen nach einer umfassenden Beschreibung von Periodizitätsschemas begegnen konnten. Wie gezeigt, bilden die eingeführten Ausdrücke und algebraischen Operationen eine adäquate Grundmenge zur Beschreibung praxisrelevanter Periodizitätsschemas.

In Kapitel 3 wurde die algebraische Notationsform lediglich informell eingeführt. Die im vorliegenden Kapitel eingeführten HSP-Bäume ermöglichen jedoch eine präzise formale Beschreibung der algebraischen Notationsform. Im Folgenden werden in Abschnitt 4.3.1 zunächst ausgewählte Labelausdrücke bzw. Verkettungsoperationen formal vorgestellt. Im Anschluss daran erfolgt in Abschnitt 4.3.2 bzw. Abschnitt 4.3.3 die formalen Definition der Änderungs- bzw. Auswahloperationen auf Basis von HSP-Bäumen.

4.3.1 Labelausdrücke

Labelausdrücke ermöglichen eine semantisch erweiterte Adressierung von (Teil-) Zyklen und Zyklenpositionen eines Periodizitätsschemas. Das Ergebnis der Adressierung bzw. eines Labelausdrucks ist eine (theoretisch unendliche) Menge von Zyklen oder Zyklenpositionen. Die Grundlage von Labelausdrücken bilden "Verkettungsfunktionen", die die einzelnen Labelangaben miteinander verknüpfen. Abbildung 4.26 zeigt die in Kapitel 3 eingeführten Formen von Verkettungen. Die Realisierung von Labelausdrücken im HSP-Baum basiert, analog zum logischen Modell, auf Verkettungsfunktionen. Während im logischen Modell entsprechende Verkettungen als Ergebnis spezifische Mengen von (Teil-)

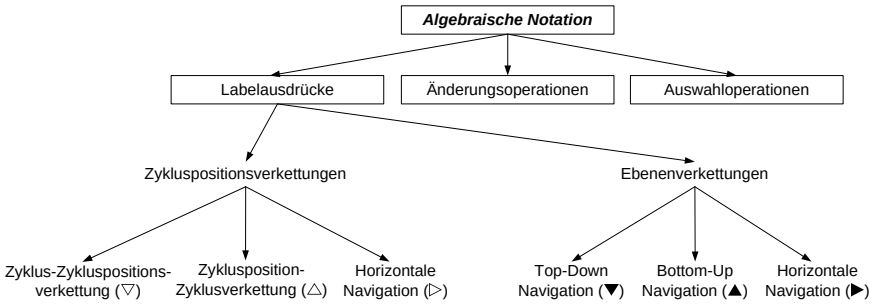


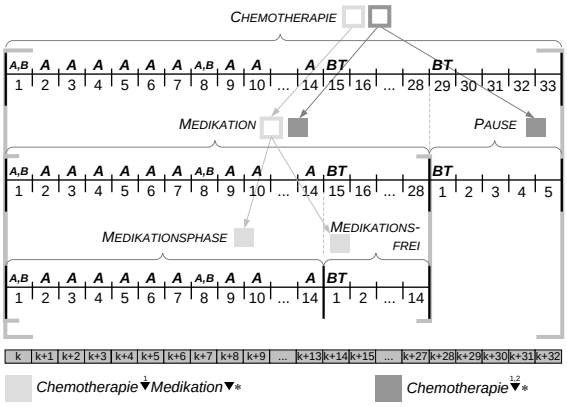
Abbildung 4.26: Übersicht zu Verkettungen in Labelausdrücke

Zyklen oder Zykluspositionen ermitteln, werden im HSP-Baum korrespondierende Mengen von Knoten bestimmt. Das bedeutet, dass ausgehend von einer übergebenen Knotenmenge und der spezifischen Semantik einer Verkettungsfunktion eine entsprechende Menge von Ergebnisknoten bestimmt wird. Im Folgenden wird die Realisierung von Verkettungsfunktionen für HSP-Bäume vorgestellt und formal definiert. Hierzu werden im ersten Schritt für ausgewählte Verkettungsfunktionen die entsprechenden Realisierungen im HSP-Baum vorgestellt. Im zweiten Schritt werden auf Basis dieser Funktion Labelausdrücke für HSP-Bäume eingeführt und formal definiert.

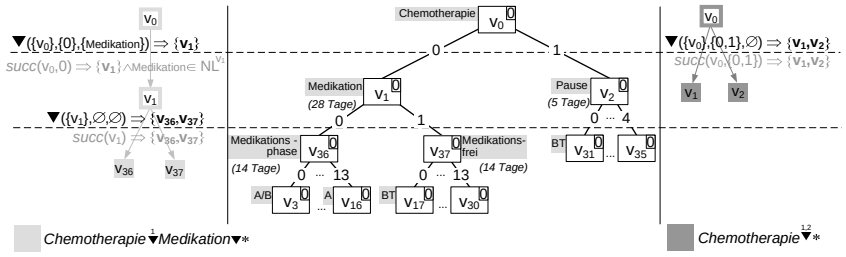
4.3.1.1 Ebenenverkettungen (Top-Down Navigation ▼)

Die Top-Down Navigation $\overset{x}{\blacktriangledown}$ navigiert im zyklenorientierten Modell, ausgehend von der höchsten Ebene, entlang der vertikalen Strukturierung nach *"unten"*. Ausgehend von einem Zyklus oder Teilzyklus werden mit der Verkettung bestimmte Teilzyklen der nächst tieferliegenden (vertikalen) Ebenen adressiert. Mit dem (optionalen) Parameter x können Teilzyklen zusätzlich anhand ihrer absoluten Position bezüglich des höheren Zyklus adressiert werden (siehe Abschnitt 3.4.1.2).

Die vertikale Strukturierung von Zyklen im logischen Modell wird im HSP-Baum durch die verschiedenen Baumebenen repräsentiert (siehe Abschnitt 4.2.1.1). Eine Top-Down Navigation im HSP-Baum entspricht somit einem Navigieren entlang der verschiedenen Baumebenen. In Abbildung 4.27 ist ein Beispiel für eine Top-Down Navigation im HSP-Baum und die korrespondierende Navigation im zyklenorientierten Modell dargestellt. In HSP-Bäumen, die lediglich aus H-Knoten bestehen, ist die Realisierung der Top-Down Navigation einfach und basiert auf der Nachfolgerfunktion der jeweiligen Baumknoten. Beispielsweise bildet das Ergebnis der Top-Down Navigation $\texttt{Chemotherapie}\overset{1,2}{\blacktriangledown}*$ die Nachfolger des Knotens v_0 (`Chemotherapie`) entlang der Kanten $\{0,1\}$ (Die Kante 0 im Baum entspricht der Zyklusposition = 1), d.h. $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}* \equiv \blacktriangledown(\{v_0\},\{0,1\}) \equiv succ(\{v_0\},\{0,1\}) = \{\mathbf{v_1},\mathbf{v_2}\}$. Die Verkettun-



(a) Top-Down Navigation im zyklenorientierten Modell



(b) Top-Down Navigation im HSP-Baum

Abbildung 4.27: Beispiel einer Top-Down Navigation im HSP-Baum und im zyklenorientierten Modell

gen in Infix-Notationsform (z.B. $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}*$) werden in eine korrespondierende Präfix-Notationsform bzw. Funktionen überführt (z.B. $\blacktriangledown(\{v_0\},\{0,1\})$), die auf der Nachfolgerfunktion im HSP-Baum basieren (z.B. $succ(\{v_0\},\{0,1\})$) und die Ergebnisknotenmenge bestimmt (z.B. $\{v_1, v_2\}$). Diese Überführung der Infix-Notation des logischen Modells auf korrespondierende Präfixnotationen wird später bei der Definition von Labelausdrücken noch näher erläutert (siehe Abschnitt 4.3.1.4).

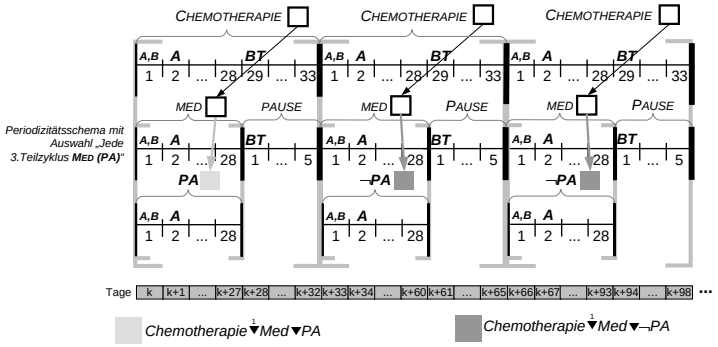
Analog zum logischen Model erfolgt auch im HSP-Baum der Umgang mit den Labelangaben bei Verkettungsfunktionen. Im logischen Modell wird bei einer Verkettungsfunktion stets zusätzlich das Label der adressierten (Teil-) Zyklen oder Zyklusspositionen mit berücksichtigt. Das heißt, das Ergebnis einer Top-Down Navigation wie etwa $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}\texttt{Medikation}$ enthält lediglich die Teilzyklen an Position eins (direkt) unterhalb von Zyklus `Chemotherapie`, die zusätzlich das Label `Medikation` aufweisen. Übertragen auf den HSP-Baum bedeutet dies, dass nach der Ermittlung der

Nachfolger eines Knotens auch die (optional) angegebenen Labelangaben überprüft werden müssen. Das heißt, anstelle der Verwendung der Positionen wie im Ausdruck $\texttt{Chemotherapie}\overset{1}{\blacktriangledown} * \overset{1}{\blacktriangledown}*$ können auch die konkreten Labelangaben verwendet werden, wie etwa bei dem Ausdruck $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}\texttt{Medikation}\overset{1}{\blacktriangledown}\texttt{Medikationsphase}$. Das Ergebnis beider Ausdrücke ist hierbei identisch. In diesem Beispiel könnten in der letzten Variante die Positionsangaben theoretisch auch entfallen. Allerdings würden dann die Positionsangaben nicht überprüft werden. Im Fall etwa, dass ein dritter Zyklus `Medikationsphase` nach dem Zyklus `Medikationsfrei` existieren würde, wäre dieser im Ergebnis des Labelausdrucks $\texttt{Chemotherapie}\blacktriangledown\texttt{Medikation}\blacktriangledown\texttt{Medikationsphase}$ ebenfalls enthalten.

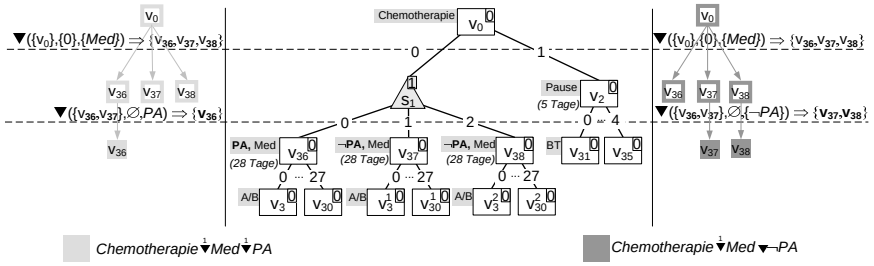
Umgang mit Auswahl- und Projektionsebenen: Bei Verkettungsfunktionen müssen die Besonderheiten von Auswahloperationen im zyklenorientierten Zeitmodell bzw. deren Abbildung im HSP-Baum berücksichtigt werden. Im zyklenorientierten Modell werden bei Auswahloperationen für jeden von einer Auswahl betroffenen (Teil-) Zyklus eine neue vertikale Ebene unmittelbar unterhalb seiner ursprünglichen Ebene eingeführt. Die neue Ebene (*Auswahlebene)* stellt eine Kopie der Originalebene dar, die lediglich um zusätzliche Labels angereichert wurde. Die Anzahl und Form der Auswahlebenen wird durch die Struktur des Auswahlschemas bei der Auswahloperation vorgegeben. Diese eingefügten Auswahlebenen entsprechen "einfachen" vertikalen Ebenen entlang derer bei einer Top-Down Navigation ebenfalls navigiert werden kann, um etwa "ausgewählte" Teilzyklen zu adressieren (siehe Abschnitt 3.4.3). Abbildung 4.28a zeigt ein Beispiel für eine Auswahl ("*Jede 2. Teilzyklus* `Med`*(ikation)"*) im zyklenorientierten Modell. Hierbei wurde für jeden Teilzyklus `Med` eine zusätzliche vertikale Ebene bzw. Auswahlebene eingeführt, wobei die ausgewählten Teilzyklen bzw. die entsprechende Ebene das Label `PA` erhalten bzw. die nichtausgewählten Teilzyklen das Label ¬`PA`. Zur Adressierung aller ausgewählten Teilzyklen muss lediglich ein weiterer Top-Down Navigationsschritt ausgehend von den Teilzyklus `Med` vorgenommen werden. Der Labelausdruck $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}\texttt{Med}\blacktriangledown\texttt{PA}$ adressiert somit alle ausgewählten Teilzyklen. Im ersten Navigationsschritt werden zunächst alle Teilzyklen mit Label `Med` adressiert, aus denen im zweiten Navigationsschritt alle ausgewählten Teilzyklen, d.h. alle Teilzyklen mit Label `PA`, adressiert werden.

Im HSP-Baum werden die Auswahlschemas und ihre Struktur durch entsprechende Auswahl- oder Projektionsebenen abgebildet. Die Struktur der Auswahlschemas bleibt im Gegensatz zum zyklenorientierten Modell vollständig erhalten und wird durch die entsprechende Ebene repräsentiert.[8] Im einfachen Fall, d.h. bei unstrukturierten Aus-

[8] Im zyklenorientierten Modell wird die Struktur des Auswahlschemas durch neue vertikale Ebenen abgebildet bzw. nachgebildet. Die Struktur des Auswahlschemas geht hierbei jedoch teilweise "verloren". In Abbildung 4.28a bestand das ursprüngliche Auswahlschema aus einem Zyklus der Länge drei mit Label `PA` für die 1.Zyklusposition und Label ¬`PA` für die 2. und 3. Zyklusposition. Die Information über den Aufbau des Auswahlschemas geht jedoch nach erfolgter Auswahl verloren. Das heißt, die Men-



(a) Top-Down Navigation im zyklenorientierten Modell



(b) Top-Down Navigation im HSP-Baum

Abbildung 4.28: Beispiel einer Top-Down Navigation im HSP-Baum und im zyklenorientierten Modell bei einfacher Auswahlebene

wahlschemas, entspricht die Auswahlebene einem einzelnen S-Knoten bzw. bei einer Projektionsebene einem einzelnen P-Knoten. Abbildung 4.28b zeigt ein entsprechendes Beispiel. Semantisch entsprechen die Blätter der Auswahlebene der Gesamtmenge an Teilzyklen, die von der Auswahl betroffen sind (unabhängig ob ausgewählt oder nicht). Mit dem Labelausdruck `Chemotherapie`$\overset{1}{\blacktriangledown}$`Med`$\blacktriangledown$`PA` bzw. `Chemotherapie`$\overset{1}{\blacktriangledown}$`Med`$\blacktriangledown\neg$`PA` können die ausgewählten bzw. nichtausgewählten Teilzyklen einzeln adressiert werden. Das heißt, für den ersten Ausdruck ergibt sich der mit ausgewählten Teilzyklen korrespondierende Knoten v_{36}. Für den zweiten Ausdruck ergibt sich die Knotenmenge $\{v_{37}, v_{38}\}$.

Im Unterschied zum zyklenorientierten Modell, erlaubt es die strukturerhaltende Abbildung von Auswahlschemas im HSP-Baum, zur Adressierung von Teilzyklen einer Auswahl zusätzlich zu den Labelangaben auch die Struktur des Auswahlschemas zu berücksichtigen. Beispielsweise können mit dem Labelausdruck `Chemotherapie`$\overset{1}{\blacktriangledown}$`Med`$\overset{3}{\blacktriangledown}\neg$`PA` alle nichtausgewählten Teilzyklen bzw. Knoten adressiert werden, die mit der dritten Zy-

ge der ausgewählten Teilzyklen kann später nur anhand ihres Label `PA` adressiert werden und nicht anhand etwa ihrer Position im ursprünglichen Auswahlschema (siehe Abschnitt 3.4.3.2).

klusposition des Auswahlschemas korrespondieren. Kennzeichnend ist hierbei, dass sich die Positionsangabe der letzten Verkettung $\overset{3}{\blacktriangledown}\neg PA$ auf die Struktur des Auswahlschemas bezieht und den Knoten v_{38} adressiert. Eine solche Adressierung des mit der dritten Zyklusposition korrespondierenden Teilzyklus ist im (logischen) zyklenorientierten Modell nicht möglich, da die Struktur des Auswahlschemas bei der Abbildung des Auswahlschemas im zyklenorientierten Modell verloren geht (siehe Abschnitt 3.4.3.2).

Problem der strukturerhaltenen Abbildung von Auswahlschemas: Durch die strukturerhaltende Abbildung von Auswahlschemas kann auf diese bei der Adressierung von Teilzyklen bzw. Knoten im HSP-Baum vollständig Bezug genommen werden. Allerdings ergibt sich dadurch auch ein Problem bezüglich der Realisierung der Verkettung im HSP-Baum. Das Problem zeigt sich bereits in dem einfachen Beispiel in Abbildung 4.28. Im zyklenorientierten Modell entspricht die Top-Down Navigation $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}\texttt{Med}$ der Menge aller Teilzyklen `Med` mit Position eins im höheren Zyklus `Chemotherapie`. Im HSP-Baum führt Verkettung $\blacktriangledown(\{v_0\}, 0)$ ausgehend von Knoten v_0 bzw. ihre Realisierung auf Basis der Nachfolgerfunktion $succ(\{v_0\}, 0)$ zunächst nur zu dem S-Knoten s_1. Aus semantischer Sicht muss die Navigation aber zu der Menge aller Blätter der Auswahlebene σ^{s_1} führen, d.h. zu den Knoten $\{v_{36}, v_{37}, v_{38}\}$, da diese der Menge aller Teilzyklen `Med` entsprechen (unabhängig ob ausgewählt oder nicht). Eine einfache (naive) Realisierungsmöglichkeit wäre anstelle des Knotens s_1 einfach die Menge aller Blätter als Ergebnis der Verkettung zurückzugeben (z.B. $\blacktriangledown(\{v_0\}, 0) = \{v_{36}, v_{37}, v_{38}\}$). Diese Realisierungsmöglichkeit wäre jedoch ebenfalls nicht für alle Fälle ausreichend. Wie bei dem Labelausdruck $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}\texttt{Med}\overset{3}{\blacktriangledown}\neg\texttt{PA}$ ist es grundsätzlich möglich, dass in nachfolgenden Verkettungen auf die Struktur des Auswahlschemas Bezug genommen wird. Die alleinige Rückgabe der Blätter der Auswahlebene als Ergebnis der ersten Verkettung würde es nachträglichen Verkettungen unmöglich machen auf die Struktur des Auswahlschemas zuzugreifen. Andererseits "weiß" die erste Verkettung nicht ob in nachfolgende Verkettungen die Struktur des Auswahlschemas benötigt wird.

Das Problem an dieser Stelle ist einerseits, dass die Struktur einer Auswahleben u.U. in mehreren Verkettungen benötigt wird. Andererseits dürfen im Ergebnis einer jeden Verkettung nur die "regulären" mit einem konkreten Teilzyklen korrespondierenden Teilzyklus enthalten sein und keine Strukturknoten des Auswahlschemas (z.B. H-Knoten, die die Struktur des Auswahlschemas repräsentieren).[9] Um diesem Problem zu begegnen, basiert die Realisierung von Verkettungen im HSP-Baum stets auf zwei Knotenmengen bzw. liefert im Ergebnis zwei Knotenmengen. Die eine Knotenmenge enthält die konkreten Teilzyklen bzw. Blätter einer Auswahlebene, die das "externe" Ergebnis der Verkettung darstellen, d.h. die Teilzyklen bzw. regulären H-Knoten, die später etwa von Änderungsoperationen manipuliert werden dürfen. Die zweite Knotenmenge bildet das

[9] Im zyklenorientierten Modell entspricht, unabhängig von der Struktur des Auswahlschemas, jede neu eingefügte Auswahlebene einem (Teil-) Zyklus. Dies bedeutet, dass pro Navigationsschritt das Ergebnis der Navigation stets einem einzelnen (Teil-) Zyklus bzw. einer Menge von (Teil-) Zyklen entspricht.

"interne" Ergebnis der Verkettung, dass lediglich die Strukturknoten des Auswahlschemas enthält, die sich bei der Navigation ausgehend von der übergebenen Knotenmenge ergeben. Auf das interne Ergebnis können nachfolgende Top-Down Navigationen aufsetzten und weiter entlang der Struktur des Auswahlschemas navigieren. Im Folgenden ist diese Grundidee der zwei Knotenmengen für das Beispiel aus Abbildung 4.28b dargestellt:

Labelausdruck	$\texttt{Chemotherapie}\overset{1}{\blacktriangledown}*$	$\overset{3}{\blacktriangledown}*$
Verkettung	$\blacktriangledown(E_0 = \{v_0\}, I_0 = \{v_0\}, 0)$	$\blacktriangledown(E_1 = \{v_{36}, v_{37}, v_{38}\}, I_1 = \{s_1\}, 3)$
Externe Ergebnis	$E_1 = \{v_{36}, v_{37}, v_{38}\}$	$E_2 = \{v_{38}\}$
Interne Ergebnis	$I_1 = \{s_1\}$	$I_2 = \{v_{38}\}$

Die Mengen E_0, E_1, E_2 stellen jeweils die externe Ergebnismenge einer Verkettungsfunktion dar, und die Mengen I_0, I_1, I_2 jeweils die interne Ergebnismenge. Initial werden die Mengen E_0 und I_0 mit den Wurzelknoten v_0 belegt.

Diese Form der zweifachen Ergebnismenge wird für alle Verkettungsfunktion im HSP-Baum verwendet. Dadurch werden aber auch jeder Verkettungsfunktion stets zwei Knotenmengen übergeben. Es hängt von der konkreten Verkettungsfunktion ab, wie die beiden (übergebenen) Mengen bei der Realisierung der Verkettung verwendet werden. Bei der Top-Down Navigation etwa wird stets die interne Ergebnismenge bei der Realisierung verwendet. Die externe Menge spielt zur Realisierung der Verkettung keine Rolle. Unabhängig von ihrer konkreten Semantik und Realisierung weisen alle Verkettungsfunktionen im HSP-Baum, im Folgenden zusammengefasst mit dem Symbol $\otimes$, die folgende Signatur auf:

$\otimes(\mathbf{T_{in}}, \mathbf{E_{in}}, \mathbf{I_{in}}, \mathbf{P}, \mathbf{L}) \mapsto (\mathbf{T_{out}}, \mathbf{E_{out}}, \mathbf{I_{out}})$	
Input:	
T_{in}	HSP-Baum $T_{in} = (N, ...)$
E_{in}	Übergebene externe Ergebnismenge mit: $E_{in} \subseteq N$ und $\forall k \in E_{in} : NT^k = \texttt{HNODE}, \nexists i \in N : k \in \sigma^i$
I_{in}	Übergebene interne Ergebnismenge mit $I_{in} \subseteq N$
P	Menge der optionalen Positionsangaben P
L	Menge der optionalen Labelangaben L
Output:	
T_{out}	Resultierende HSP-Baum
E_{out}	Resultierende externe Ergebnismenge mit: $E_{in} \subseteq N$ und $\forall k \in E_{in} : NT^k = \texttt{HNODE}, \nexists i \in N : k \in \sigma^i$
I_{out}	Resultierende interne Ergebnismenge mit $I_{in} \subseteq N$

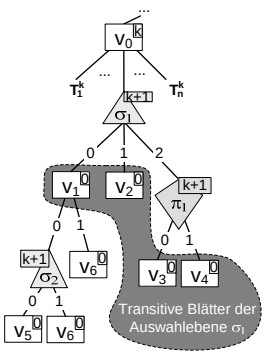


Abbildung 4.29: Beispiel für transitive Blätter einer Auswahlebene

Die angegebenen Bedingungen für die Mengen E_{in} und I_{in} und die Ergebnisknotenmengen E_{out} und I_{out} sind identisch. Dies stellt sicher, dass Verkettungsfunktionen später in Labelausdrücken (beliebig) miteinander kombiniert verwendet werden können. Für die Knotenmengen E_{in} und E_{out} wird stets gefordert, dass diese zum einen nur H-Knoten enthält und diese H-Knoten nicht in einer Auswahlebene enthalten sind, d.h. keine Strukturknoten einer Auswahlebene darstellen. Lediglich Knoten, die dieser Bedingung entsprechen, stellen konkrete Teilzyklen bzw. Zyklusspositionen dar, die durch eine Verkettung adressiert bzw. später durch entsprechende Operationen manipuliert werden dürfen.

Realisierung der Top-Down Navigation: Die Realisierung der Top-Down Navigation basiert auf einer besonderen Menge von Blättern bei Auswahl- und Projektionsebenen, d.h. den *transitiven Blättern,* die im Folgenden zunächst formal eingeführt werden. Semantisch entsprechen die transitiven Blätter allen Blätter einer Auswahlebene σ^i oder Projektionsebene π^i, die direkt oder transitiv mit dieser Ebene gekoppelt sind. Das heißt, im Gegensatz zu den "normalen" Blättern einer solchen Ebene umfassen die transitiven Blättern auch alle Blätter nachfolgender, gekoppelter Ebene (siehe Abschnitt 4.2.2.2). Abbildung 4.29 zeigt ein entsprechendes Beispiel.

Die Menge der transitiven Blätter der Auswahlebene σ_1 werden durch die Blätter der Ebene selber sowie durch die Blätter aller mit der Ebene σ_1 gekoppelten Auswahlebenen gebildet. Im vorliegenden Beispiel ist die Projektionsebene π_1 mit der Auswahlebene σ_1 gekoppelt, d.h. die Menge $\{v_1, v_2, v_3, v_4\}$ stellt die transitiven Blätter der Auswahlebene σ_1 dar. Neben H-Knoten wie in dem Beispiel könnten auch 0-Knoten in der Menge der transitiven Blätter enthalten sein. In dem Beispiel ist die Auswahlebene σ_2 ist nicht in der Menge der transitiven Blätter von σ_1 enthalten, da diese aufgrund des H-Knotens v_1 nicht mit Ebene σ_1 gekoppelt ist. Würde andererseits anstelle des Knoten v_4 eine weitere Auswahlebene σ_3 existieren, würden deren Blätter in der Menge der transitiven Blätter enthalten sein, da σ_3 dann mit π_1 gekoppelt wäre, π_1 mit σ_1 gekoppelt ist und

folglich σ_3 dann (transitiv) mit σ_1 gekoppelt wäre. Formal sind die transitiven Blätter einer Ebene wie folgt definiert:

Definition 4.7 (Transitive Blätter): *Sei $T = (N, ...)$ ein HSP-Baum, Σ die Menge aller Auswahlebenen und Π die Menge aller Projektionsebenen von T. Sei Weiterhin $A \subseteq \Sigma \cup \Pi$ die Menge aller Projektions- und Auswahlebenen, die nicht in einer anderen Ebene enthalten (eingeschachtelt) sind, d.h. es gilt: $\forall a \in A : \nexists k \in N : a \in \sigma^k \cup \pi^k$.*
Als transitive Blätter der Auswahl- oder Projektionsebene $E^i \in A$ wird die folgende Knotenmenge bezeichnet:

$$E^i_{leafs} \cup \bigcup_{m \in M} \{\sigma^m_{leafs} \cup \pi^m_{leafs}\}$$

wobei $M = \{m_1, ..., m_k\} \subseteq N$ die Knotenmenge ist, für die gilt:

1. *Die Menge M ist vollständig:*
   $\nexists x \in N, (\sigma^x \vee \pi^x) \in A : \; x \in \sigma^k_{leafs} \wedge k \in M \wedge x \notin M$
2. *Alle in der Menge M enthaltenen Knoten bzw. Auswahl- oder Projektionsebenen sind transitiv mit den Blätter der Ebene E^i verbunden:*
   $\forall m \in M : \exists k \in \{M \cup i\} : \; m \in \sigma^k_{leafs} \vee m \in \pi^k_{leafs}$

Die Funktion $TransitiveLeafs : N \mapsto N^*$ mit $TransitiveLeafs^i$ bestimmt die transitiven Blätter der Ebene E^i.

Aus der Definition folgt $E^i_{leaf} \subseteq TransitiveLeafs^i$, d.h. die Blätter einer Ebene bilden eine Teilmenge der transitiven Blätter der Ebene. Lediglich im Fall $M = \emptyset$ sind die Menge der Blätter und die Menge der transitiven Blätter identisch. Die Menge der transitiven Blätter ist jedoch nur für solche Ebenen definiert, die selber nicht in einer anderen Ebene enthalten sind (siehe Bedingung für Menge A), wie etwa bei geschachtelten Ebenen. Dies hängt damit zusammen, dass für die vorliegende Verkettung aber auch für Verkettungen im Allgemeinen lediglich die Blätter von Ebenen relevant sind, die konkreten Teilzyklen bzw. Zykluspositionen entsprechen. Diese Blätter sind jedoch immer bei den "obersten" Auswahlebenen vorzufinden. Die Blätter von Ebenen hingegen, die in einer Auswahlebene enthalten sind, stellen keine konkreten Teilzyklen oder Zykluspositionen dar, sondern eine Auswahl von Strukturknoten der Auswahlebene.

Kennzeichnend für die Realisierung der Top-Down Navigation bzw. für Verkettungen im Allgemeinen ist die Unterscheidung bei der Erzeugung der beiden resultierenden Knotenmengen E_{out} und I_{out}. Die Realisierung der Top-Down Navigation $\overset{P}{\blacktriangledown} L$ ist im Folgenden skizziert.

1. Im ersten Schritt werden zunächst für jeden Knoten i aus der Menge I_{in}, die Menge N_1 der Nachfolgerknoten bestimmt. Hierbei müssen die optionalen Positionsangaben P berücksichtig werden.

a) Falls die Bedingung $P \neq \emptyset$ gilt, werden die Positionsangaben bei der Bestimmung der Nachfolger berücksichtigt. Hierzu werden in der Funktion $N_1 = succ(i, \{p_0, ..., p_k\})$ für den Knoten i alle Nachfolger entlang der Kanten $p_0, ..., p_k$ bestimmt. Positionsangaben bzw. Kantenindizes die "ungültig" sind (d.h. bei denen dieser Kantenindex nicht existiert) werden ignoriert. In dem später vorgestellten konkreten Algorithmus erfolgt dies direkt durch die Nachfolgerfunktion $succ()$.

b) Im Falle einer leeren Menge von Positionsangaben werden alle Nachfolger des Knotens i bestimmt, unabhängig vom konkreten Kantenindex, d.h. $N_1 = succ(i)$.

2. Im zweiten Schritt werden die Labelangaben berücksichtig, falls die Bedingung $L \neq \emptyset$ gilt. Für jeden Knoten $x \in N_1$ werden die die Labelangaben überprüft und die Menge $N_2 \subseteq N_1$ abgeleitet. Zwei Fälle müssen hierbei unterschieden werden:

   a) Der Knoten x ist ein S-Knoten oder P-Knoten. In diesem Fall hat die Top-Down Navigation den "Anfang" einer Auswahl- oder Projektionsebene adressiert. An diesem Punkt hat jedoch noch keine Navigation "in" die Ebene selber begonnen und es wird (semantisch) die Menge aller von der Auswahl betroffenen Teilzyklen adressiert (unabhängig ob ausgewählt oder nicht). Die Labelangabe bezieht sich an dieser Stelle somit auf die konkreten Teilzyklen und nicht (wie im nachfolgenden Fall, auf die Strukturierungsebenen und damit auf den unmittelbaren Nachfolger). Zur Überprüfung der Labelangabe müssen somit die Blätter der Auswahl- oder Projektionsebene überprüft werden. Das heisst, für den den Knoten x muss überprüft werden, ob die Blätter der Ebene (mindestens) ein entsprechendes Label aufweisen, d.h. es muss gelten $\exists j \in \sigma^x_{leafs} : NL^j \subseteq L$. Nur falls eines der Blätter der Ebene von Knoten x diese Bedingung erfüllt, wird der Knoten x in die Menge N_2 übertragen.

   b) Der Knoten i ist ein H-Knoten. Hier müssen die Labelangaben direkt bei dem Knoten x überprüft, d.h. sowohl bei regulären H-Knoten (d.h. H-Knoten, die keinen Strukturierungsknoten in einer Auswahl- oder Projektionsebene entsprechen) oder auch für Strukturierungsknoten (d.h. H-Knoten in einer Ebene). In beiden Fällen bezieht sich die Labelangabe der Top-Down Navigation im HSP-Baum auf den Knoten x selber. Der Knoten x wird nur dann in die Menge N_2 übertragen, wenn mindestens eines seiner Label in der Menge der Labelangaben enthalten ist.

3. Im dritten Schritt wird die interne Knotenmenge I_{out} gebildet, in dem die Menge N_2 in die Menge I_{out} übertragen wird. In der Menge N_2 sind alle "internen" Knoten enthalten, die im nachfolgenden Schritt direkt oder indirekt zu der Menge der externen Knoten führen. Eine Überprüfung der Knoten etwa bezüglich ihres Knotentyps findet hier nicht statt. Es ist Aufgabe der "nachfolgenden" Verket-

tungen diese "Rohmenge" an Knoten entsprechend auszuwerten bzw. gemäß ihrer Semantik zu interpretieren.

4. Im vierten Schritt wird basierend auf im vorherigen Schritt gebildeten Menge I_{out} die externe Knotenmenge E_{out} ermittelt. Für jeden Knoten $x \in N_2$ gilt es dazu zu überprüfen ob dieser in einer Auswahlebene enthalten ist oder einen "regulären" H-Knoten darstellt (d.h. ein H-Knoten, der einen konkreten Teilzyklus repräsentiert und keinen Strukturknoten einer Auswahlebene).

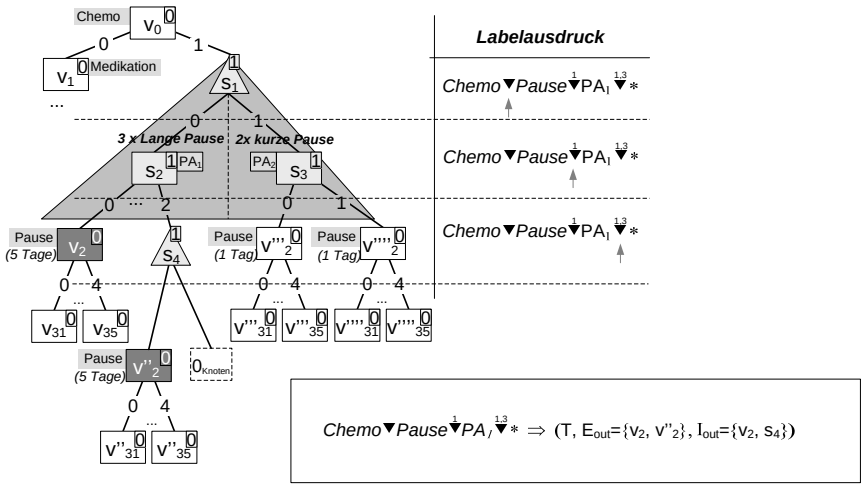
   a) Im ersteren Fall wird für den Knoten x zunächst die zughörige "oberste" Auswahl- oder Projektionsebene E^i ermittelt, d.h. die Ebene, in der der Knoten x oder die Ebene des Knoten x ggfs. enthalten ist. Für die Ebene E^i wird die Menge der transitiven Blätter mit der Menge (aller) Nachfolgerknoten von x geschnitten. Im Ergebnis erhält man alle transitiven Blätter der Ebenen E^i die von Knoten x aus (nach unten) erreichbar sind. Die transitiven Blätter stellen stets konkrete Teilzyklen dar, die direkt in die Menge E_{out} übertragen werden können.

   b) Im zweiten Fall stellt der Nachfolgerknoten einen regulären Nachfolgerknoten dar, der direkt in die Menge E_{out} übertragen werden kann.

5. Im letzten Schritt wird die erzeugte externe Knotenmenge E_{out} bereinigt. Hierzu werden alle Knoten aus der Menge gelöscht, die entweder ein 0-Knoten oder ein Blattknoten sind. 0-Knoten werden im HSP-Baum als "interner" Platzhalter benötigt, die nicht von "außen" adressiert werden dürfen. Blattknoten wiederum entsprechen Zyklusspositionen, die durch diese Verkettung nicht adressiert werden.

Das oben skizzierte Vorgehen zur Realisierung der Top-Down Navigation ist in Abbildung 4.30 für ein Beispiel dargestellt. Im oberen Bereich der Abbildung ist der entsprechende HSP-Baum dargestellt. Im unteren Bereich ist die Auswertung des Labelausdrucks $Chemo \blacktriangledown Pause \overset{1}{\blacktriangledown} PA \overset{1,3}{\blacktriangledown} *$ dargestellt. Hierbei ist die von Ebene zu Ebene wechselnde Belegung der externen und internen Knotenmenge gut zu erkennen.

Abschließend ist die Realisierung der Top-Down Navigation vollständig in Algorithmus 4.1 gezeigt. Die in der Signatur angegebenen Bedingungen sind zusätzlich zu den in der allgemeinen Signatur angegebenen Bedingungen zu verstehen. Hierbei nehmen wir zwei Hilfsfunktionen an. Die Hilfsfunktion $InsertElement(A, x)$ fügt in eine Liste A ein Element x ein. Die Hilfsfunktion $InsertElement(A, B)$ ist analog, fügt jedoch anstatt eines einzelnen Elements eine Menge bzw. Liste B dieser ein.



Labelausdruck	Verkettungsfkt.	Erzeugung von I_{out}	Erzeugung von E_{out}
$Chemo▼Pause^{1}▼PA_I^{1,3}▼*$	▼(T,E_{in}={v_0},I_{in}={v_0}, P=∅,L={*Pause*})	P=∅ ⇒ *succ*(v_0)={v_1,s_1} L≠∅ ⇒ $NL^{\sigma^{S_1}_{leafs}} \subseteq L$ $NL^{v_1} \not\subseteq L$ **I_{out} = {s_1}**	$s_1 \in \sigma^{S_1} \Rightarrow E_{out}$ = {v_2,v'_2,v''_2, 0_{Knoten},v'''_2, v''''_2} **E_{out} = {v_2,v'_2,v''_2,v'''_2, v''''_2}** (nach Bereinigung)
$Chemo▼Pause^{1}▼PA_I^{1,3}▼*$	▼(T, E_{in}={v_2,v''_2,v''''_2}, I_{in}={s_1},P={0},L={PA_1})	P≠∅ ⇒ *succ*(s_1,{0})={s_2} L≠∅ ⇒ $NL^{s_2} \subseteq L$ **I_{out} = {s_2}**	$s_2 \in \sigma^{S_1} \Rightarrow E_{out}$ = {v_2,v'_2,v''_2, 0_{Knoten}} **E_{out} = {v_2,v'_2,v''_2}** (nach Bereinigung)
$Chemo▼Pause^{1}▼PA_I^{1,3}▼*$	▼(T, E_{in}={v_2,v'_2,v''_2}, I_{in}={s_2}, P={0,2},L=∅)	P≠∅ ⇒ *succ*(s_2,{0,2})={v_2,s_4} L=∅ **I_{out} = {v_2,s_4}**	$v_2 \notin \sigma^{*} \Rightarrow E_{out_1}$ = {v_2} $s_4 \in \sigma^{S_4} \Rightarrow E_{out}$ = {v''_2, 0_{Knoten}}∪E_{out_1} **E_{out} = {v_2,v''_2}** (nach Bereinigung)

Abbildung 4.30: Komplexes Beispiel einer Top-Down Navigation

Algorithmus 4.1 $\blacktriangledown(\mathbf{T_{in}}, \mathbf{E_{in}}, \mathbf{I_{in}}, \mathbf{P}, \mathbf{L}) \mapsto (\mathbf{T_{out}}, \mathbf{E_{out}}, \mathbf{I_{out}})$

Input:

T_{in}	HSP-Baum $T_{in} = (N, ...)$
E_{in}	–
I_{in}	–
P	Menge der optionalen Positionsangaben $P = \{p_0, ..., p_k\}$
L	Menge der optionalen Labelangaben $L = \{l_0, ..., l_h\}$

Output:

T_{out}	$T_{out} = T_{in}$
E_{out}	$\forall n \in E_{out} : NT^n = \texttt{HNODE} \wedge n \neq$ Blattknoten
I_{out}	$I_{out} \subseteq N$

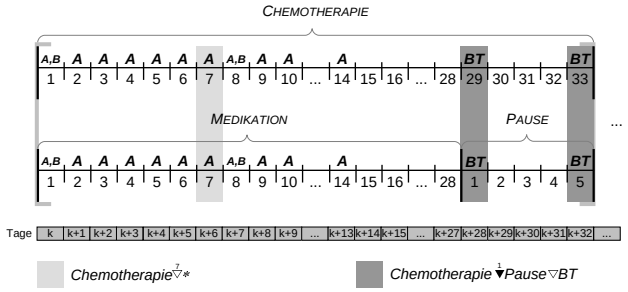
1: **for all** $k \in I_{in}$ **do**
2: **if** $P \neq \emptyset$ **then** {Berücksichtigung der Positionsangaben}
3: $N_1 = succ(k, p_0, ..., p_n)$
4: **else**
5: $N_1 = succ(k)$
6: **end if**
7: **if** $L \neq \emptyset$ **then** {Berücksichtigung der Labelangaben}
8: **for all** $n \in N_1$ **do**
9: **if** $(\nexists\sigma^n \vee \nexists\pi^n) \wedge (NL^n \subseteq L)$ **then**
10: $InsertElement(N_2, n)$
11: **else if** $(\exists\sigma^n \vee \exists\pi^n) \wedge ((\exists i \in (\sigma^n_{leafs} \cup \pi^n_{leafs}) : NL^i \subseteq L)$ **then**
12: $InsertElement(N_2, n)$
13: **end if**
14: **end for**
15: **end if**
16: $InsertElement(I_{out}, N_2)$ {Erzeugung der internen Knotenmenge I_{out}}
17: **for all** $x \in N_2$ **do** {Erzeugung der externen Knotenmenge E_{out}}
18: **if** $\exists i \in N : (x \in \sigma^i \vee x \in \pi^i) \wedge (\nexists j \in N : i \in \sigma^j \vee i \in \pi^j)$ **then**
19: $tmpEout = SUCC(x) \cap TransitivLeafs^i$
20: **else**
21: $tmpEout = \{x\}$
22: **end if**
23: **for all** $e \in tmpEout$ **do** {Bereingung der externen Knotenmenge E_{out}}
24: **if** $e \neq$ Blattknoten $\vee\ NT^e \neq \texttt{ONODE}$ **then**
25: $InsertElement(E_{out}, e)$
26: **end if**
27: **end for**
28: **end for**
29: **end for**

4.3.1.2 Zykluspositionsverkettungen (Zyklus-Zykluspositionsverkettung $\triangledown$)

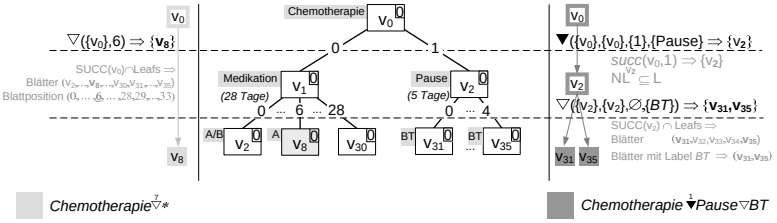
Die Zyklus-Zykluspositionsverkettung $\overset{x}{\triangledown}$ ermöglicht die Adressierung von Zyklusposition-nen in einem Zyklus (siehe Abschnitt 3.4.1.1). Zum einen können hierzu die absoluten Positionnummer verwendet werden, wie etwa bei dem Labelausdruck `Chemotherapie`$\overset{7}{\triangledown}*$, der die siebte Zyklusposition eines jeden Zyklus `Chemotherapie` adressiert. Zum anderen können lediglich Labelangaben verwendet werden, wie etwa im Ausdruck `Chemotherapie`$\triangledown$`A`, der alle Zykluspositionen mit Label `A` adressiert. Abbildung 4.31a zeigt ein weiteres Beispiel einer Zyklus-Zykluspositionsverkettung im zyklenorientierten Modell. Kennzeichnend für die Verkettung ist, dass diese unabhängig von der Strukturierungsebene angewendet werden kann. Das heißt, die Adressierung der Zykluspositionen eines Zyklus kann bereits auf höchster Zyklusebene erfolgen und damit unabhängig von der (tieferliegenden) horizontalen und vertikalen Strukturierung des betreffenden Zyklus. Bei dem Ausdruck `Chemotherapie`$\overset{7}{\triangledown}*$ (siehe Abbildung 4.31a) wird etwa die tieferliegende Strukturierung des Zyklus `Chemotherapie` ignoriert und dieser als ein unstrukturierter Zyklus der Länge 33 mit den Zykluspositionen $\{0, ..., 32\}$ verstanden. Die Adressierung der Zykluspositionen kann jedoch auch erst auf der "tiefsten" Strukturierungsebene und somit unter Berücksichtigung der horizontalen und vertikalen Strukturierung des betreffenden Zyklus erfolgen. Im Ausdruck `Chemotherapie`$\blacktriangledown$`Pause`$\triangledown$`BT` in Abbildung 4.31a wird zunächst entlang der Strukturierung des Zyklus `Chemotherapie` navigiert und erst auf unterster Ebene für den Zyklus `Pause` die Adressierung der Zykluspositionen mit Label `BT` bzw. Positionen eins und fünf vorgenommen.

Im HSP-Baum korrespondieren die Blattknoten des Baumes mit den Zykluspositionen des zyklenorientierten Modells. Kennzeichnend für den HSP-Baum ist, dass, im Gegensatz zum allgemeinen Modell, die einzelnen Teilzyklen eines strukturierten Zyklus bzw. deren Zykluspositionen nicht in ausmaterialisierter Form repräsentiert werden. Stattdessen werden lediglich die Teilzyklen der tiefsten Strukturierungsebene direkt im HSP-Baum repräsentiert. (Teil-) Zyklen auf höherer Ebene werden lediglich indirekt abgebildet und es gilt, die Menge entsprechender Zykluspositionen von den im Baum tieferliegenden Zyklen abzuleiten. Genauer gesagt ergeben sich die Zykluspositionen eines (Teil-) Zyklus aus der Summe der erreichbaren Blattknoten des mit dem (Teil-) Zyklus korrespondierenden Knotens (siehe Abschnitt 4.2.1.1).

Die Abbildung lediglich der tiefsten Strukturierungsebene im HSP-Baum muss bei der Realisierung der Zyklus-Zykluspositionsverkettung besonders berücksichtigt werden. Im einfachsten Fall, d.h. einem HSP-Baum bestehend aus H-Knoten erweist sich die Realisierung als einfach. In diesem Fall muss lediglich die (geordnete) Menge der von einem H-Knoten erreichbaren Blattknoten ermittelt werden. Die Position in der Menge entspricht der bei einer Zyklus-Zyklusposition (optional) angegebenen Positionsangabe. Abbildung 4.31b zeigt ein entsprechendes Beispiel. Der Labelausdruck `Chemotherapie`$\overset{7}{\triangledown}*$ bzw. die Verkettung $\triangledown(\{v_0\}, 6)$ führt im Zwischenergebnis, d.h. für alle von Knoten v_0 erreichbaren Blätter, zu der Menge aller Blätter des Baums. Im Endergebnis der Verket-



(a) Zyklus-Zykluspositionsverkettung im zyklenorientierten Modell

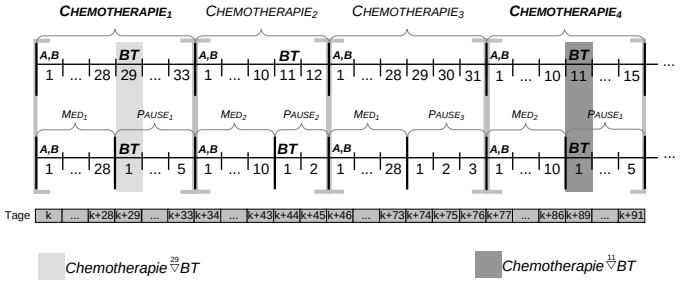


(b) Zyklus-Zykluspositionsverkettung im HSP-Baum

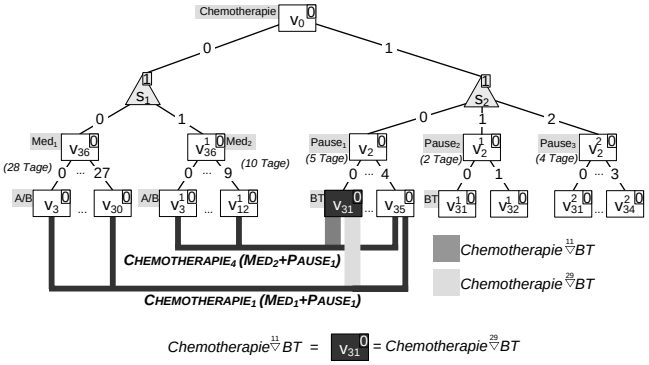
Abbildung 4.31: Beispiel einer Zyklus-Zykluspositionsverkettung im HSP-Baum und im zyklenorientierten Modell

tung ergibt sich der Knoten v_8 als der mit der Zyklusposition sieben korrespondierende Knoten. In dem zweiten Beispiel, d.h. bei dem Ausdruck `Chemotherapie▼Pause▽BT` werden die Zykluspositionen erst auf der tiefsten Strukturierungsebene adressiert, d.h. die Zykluspositionen mit Label `BT`. Im Ergebnis führt dies zu der Menge an Blattknoten $\{v_{31}, v_{35}\}$, die mit den Zykluspositionen korrespondieren. Das analoge Ergebnis würde auch bei dem Ausdruck $\texttt{Chemotherapie▼Pause}\overset{1,5}{\triangledown} *$ ohne Labelangabe und nur mit Positionsangaben entstehen.

Problem der Zyklus-Zykluspositionsverkettung im Umgang mit Auswahl- und Projektionsebenen: Die Realisierung der Zyklus-Zykluspositionsverkettung erweist sich im Falle von Bäumen mit Auswahl- und Projektionsebenen als problematischer. Abbildung 4.32 zeigt ein entsprechendes Beispiel. In Abbildung 4.32a ist das Beispiel im zyklenorientierten Modell dargestellt. Kennzeichnend für das Beispiel ist, dass aufgrund der unterschiedlichen Längen der (tieferliegenden) Teilzyklen $\texttt{Med}_*$ und $\texttt{Pause}_*$ die (einzelnen) Chemotherapiezyklen auf oberster Ebene (`Chemotherapie`) ebenfalls unterschiedliche Längen aufweisen. Die dargestellten Zyklus-Zykluspositionsverkettungen adressieren jeweils die Zyklusposition mit dem Label `BT` (Bluttest). Allerdings wird durch den Aus-



(a) Zyklus-Zykluspositionsverkettung im zyklenorientierten Modell



(b) Zyklus-Zykluspositionsverkettung in einem HSP-Baum mit Auswahlebenen

Abbildung 4.32: Problem der Zyklus-Zykluspositionsverkettung im HSP-Baum

druck $\texttt{Chemotherapie}\overset{29}{\triangledown}\texttt{BT}$ der Bluttest in dem Chemotherapiezyklus $\texttt{Chemotherapie}_1$ mit 28 Tagen Medikation ($\texttt{Med}_1$) und fünf Tagen Pause ($\texttt{Pause}_1$) adressiert. Durch den Ausdruck $\texttt{Chemotherapie}\overset{11}{\triangledown}\texttt{BT}$ wiederum wird der Bluttest in dem Chemotherapiezyklus $\texttt{Chemotherapie}_4$ mit 10 Tagen Medikation ($\texttt{Med}_2$) und auch fünf Tagen ($\texttt{Pause}_1$) adressiert. Die jeweils adressierten Zykluspositionen sind in unterschiedlichen Chemotherapiezyklen ($\texttt{Chemotherapie}_1$ und $\texttt{Chemotherapie}_4$) enthalten und somit auch als zwei verschiedene Zykluspositionen zu betrachten.

Die Darstellung des Beispiels als HSP-Baum zeigt Abbildung 4.32b. Sowohl für den Teilzyklus Medikation ($\texttt{Med}_1, \texttt{Med}_2$) als auch für den Teilzyklus Pause ($\texttt{Pause}_1$, $\texttt{Pause}_2$, $\texttt{Pause}_3$) bestehen entsprechende Auswahlebenen. Ebenfalls dargestellt sind die mit den Zykluspositionen der Zyklen $\texttt{Chemotherapie}_1$ und $\texttt{Chemotherapie}_4$ korrespondierende Menge von Blättern. Für den Zyklus $\texttt{Chemotherapie}_1$ ist dies die Knotenmenge $\{v_3^1, ..., v_{12}^1, v_{31}, ..., v_{35}\}$ und für den Zyklus $\texttt{Chemotherapie}_4$ die Knotenmenge

$\{v_3, ..., v_{30}, v_{31}, ..., v_{35}\}$. Anhand der Knotenmengen zeigt sich bereits das Problem im Umgang mit Auswahlebenen bei der Zyklus-Zyklusspositionsverkettung. Die beiden Ausdrücke $\texttt{Chemotherapie}\overset{11}{\triangledown}\texttt{BT}$ und $\texttt{Chemotherapie}\overset{29}{\triangledown}\texttt{BT}$ würden jeweils den Knoten v_{31} adressieren. Das heißt wiederum, dass die beiden im zyklenorientierten Modell unterschiedlichen Zyklusspositionen mit Label `BT` im HSP-Baum durch einen einzelnen Knoten repräsentiert werden. Eine semantische Unterscheidung beider Positionen bzw. Knoten ist im HSP-Baum somit nicht möglich. So würde etwa das Löschen des Labels `BT` für die Zyklusposition 11 ($\texttt{Chemotherapie}\overset{11}{\triangledown}\texttt{BT}$), dazu führen, dass das Label am Knoten v_{31} gelöscht wird. Dies führt automatisch jedoch auch dazu, dass das Label auch für die Position 29 ($\texttt{Chemotherapie}\overset{29}{\triangledown}\texttt{BT}$) unerwünschterweise gelöscht wäre.

Das generelle Problem an dieser Stelle ist, dass im HSP-Baum keine "wechselnden" Vorgänger- bzw. Nachfolgerbeziehungen von (Teil-) Zyklen oder auch Zyklusspositionen berücksichtigt werden. Das heißt, dass etwa der Knoten v_{31} bzw. v_2 einmal den Knoten v_{36} und einmal den Knoten v_{36}^1 als Vorgänger aufweist. Eine von der Beziehung abhängige Unterscheidung der Eigenschaften eines Knotens, wie etwa die Labelbelegung, wird im HSP-Baum (derzeit) nicht unterstützt. Für das obige Beispiel würde das etwa bedeuten, dass die Labelbelegung des Knotens v_{31} nur für die Nachfolgebeziehungen $v_{36}^1 \Rightarrow v_2$ gelöscht wird. In dem anderen Fall, d.h. für die Nachfolgebeziehung $v_{36} \Rightarrow v_2$ müsste das Label `BT` weiterhin dem Koten v_2 (Teilzyklus $\texttt{Pause}_1$) bzw. dem Knoten v_{31} (Zyklusposition 29) zugeordnet sein. In der vorliegenden Arbeit wird eine solche Berücksichtigung von Vorgänger- und Nachfolgebeziehungen im HSP-Baum nicht näher betrachtet. Im Fokus dieser Arbeit ist weniger, die Vollständigkeit, sondern viel mehr die Durchgängigkeit der vorgestellten Ansätze aufzuzeigen. Wie wir später noch zeigen werden, ist dieses Problem auch für andere (ebenfalls nicht dieser Arbeit betrachteten) Verkettungsfunktionen relevant (siehe Abschnitt 4.3.1.4). Wir überlassen es zukünftigen Arbeiten, diesem Problem des HSP-Baums zu begegnen, um eine vollständige Unterstützung der Verkettungsfunktionen zu gewährleisten.

Um eine grundlegende Unterstützung der Zyklus-Zyklusspositionsverkettung zu gewährleisten und gleichzeitig das beschriebene Problem im Zusammenhang mit Auswahlebenen zu vermeiden, wird die Verwendung der Verkettung in dieser Arbeit eingeschränkt. Die Verkettung darf hierzu nur auf der tiefsten Strukturierungsebene angewendet werden. Das heißt, die Verkettung darf nur für die im HSP-Baum in "materialisierter" Form dargestellten Teilzyklen verwendet werden. Für indirekt abgebildete Teilzyklen, d.h. alle Teilzyklen, die eine (tieferliegende) horizontale Strukturierung aufweisen und im HSP-Baum erst durch einen Zusammenführen ihrer tieferliegenden Teilzyklen abgeleitet werden können, ist diese Verkettung nicht zulässig. Genau in diesem Fall müssten ansonsten die Vorgänger- und Nachfolgerbeziehungen berücksichtigt werden. Allerdings führt die Einschränkung auch dazu, dass stets durch die vollständige Struktur eines Zyklus navigiert werden muss, um Zyklusspositionsangaben zu adressieren. Für das obige Beispiel etwa muss die Zyklusposition mit Label `BT` durch den Ausdruck $\texttt{Chemotherapie}\overset{1}{\blacktriangledown}\texttt{Pause}\triangledown\texttt{BT}$ bzw. $\texttt{Chemotherapie}\overset{1}{\blacktriangledown} * \overset{1}{\triangledown} *$ adressiert werden.

Anwendung bei strukturierten Auswahl- oder Projektionsebenen: Bei der Adressierung von Zyklusspositionen, die von einer Zyklusspositionsauswahl betroffen sind, kann die Struktur der korrespondierenden Auswahl- oder Projektionsebene ebenfalls verwendet werden. In Abbildung 4.33 ist ein Beispiel für eine Zyklusspositionsauswahl mit strukturierten Auswahlschema bzw. Ebene für die erste Zyklusposition des Teilzyklus `Pause` dargestellt. Im einfachen Fall kann etwa mit den Ausdruck $\texttt{Chemotherapie}\blacktriangledown\texttt{Pause}\overset{1}{\triangledown}*$ die Zyklusposition eins ohne Berücksichtigung der Auswahl adressiert werden, d.h. im Ergebnis erhält man die Menge an Blattknoten $\{v_{31}^1, ..., v_{35}^5\}$.

Die Verwendung der Struktur des Auswahlschemas einer Zyklusspositionsauswahl basiert auf der Top-Down Navigation (▼). Dies ergibt sich dadurch, das im zyklenorientierten Modell bei einer Zyklusspositionsauswahl für die Zyklen deren Positionen von der Auswahl betroffen sind, erneut eine zusätzliche (tieferliegende) Strukturierungsebene (für den Zyklus) eingeführt wird (siehe Abschnitt 3.4.3.2). Eine Adressierung der Zyklusspositionen bei einer strukturieren Auswahl entspricht somit erneut einem Navigieren entlang der entsprechend eingeführten Strukturierungsebenen. So navigiert etwa der Ausdruck $\texttt{Chemotherapie}\blacktriangledown\texttt{Pause}\overset{1}{\blacktriangledown} * \triangledown\texttt{BT}$ mit der zweiten Top-Down Verkettung entlang der Strukturierung des Zyklusspositionsauswahl und adressiert die Blattknoten $\{v_{31}^1, v_{31}^2, v_{35}^3\}$. Wie gut zu erkennen ist, findet zur Navigation in der Struktur des Auswahlschemas der Zyklusspositionsauswahl kein Wechsel auf die Zyklusspositionsebene mittels der Verkettung $\triangledown$ statt. Stattdessen wird die Struktur des Auswahlschemas, analog wie bei anderen Auswahlen, auch als zusätzliche vertikale Ebenen verstanden. Der Übergang zu den Zyklusspositionen bzw. die Anwendung der Verkettung $\triangledown$ kann an beliebiger Stelle entlang der Strukturierungsebenen der Zyklusspositionsauswahl erfolgen. Im Ergebnis führt diese dann zu einer Menge an Zyklusspositionen. Kennzeichnend ist, dass die optionale Positionsangabe x der Verkettung $\overset{x}{\triangledown}$ bei einer Zyklusspositionsauswahl keine Auswirkung hat. Semantik und Ziel einer Zyklusspositionsauswahl ist es, eine Menge von Zyklusspositionen (periodisch) auszuwählen, unabhängig davon in welchem Zyklus diese enthalten sind. Im Ergebnis führt dies zu einer Menge an Positionen, bei denen jedoch die ursprüngliche Zyklusposition bzgl. des höheren Zyklus irrelevant ist und somit auch die Positionsangabe bei der Verkettung bei erfolgter Zyklusspositionsauswahl.

Realisierung der Zyklus-Zyklusspositionsverkettung: Kennzeichnend für die Realisierung der Zyklus-Zyklusspositionsverkettungen ist, dass diese nahezu identisch zur Realisierung der Top-Down Navigation ist. Lediglich im ersten und letzten Schritt ergeben sich Abweichungen aufgrund der Semantik der Verkettung. Die Realisierung der Top-Down Navigation $\overset{P}{\triangledown}L$ ist im Folgenden skizziert.

1. Im ersten Schritt werden zunächst für jeden Knoten i aus der Menge I_{in}, die Menge N_1 der Nachfolgerknoten bestimmt. Hierbei müssen die optionalen Positionsangaben P berücksichtig werden.

   a) Falls die Bedingung $P \neq \emptyset$ gilt, werden die Positionsangaben bei der Bestim-

mung der Nachfolger berücksichtigt. Allerdings müssen zwei Teilfälle berücksichtigt werden

i. Befindet sich der Knoten i innerhalb einer Auswahlebene werden die Positionsangaben ignoriert und stattdessen alle Nachfolgerknoten ermittelt, d.h. $N_1 = succ(i)$. Es handelt sich hier um den oben erläuterten Fall, dass eine Zykluspositionsauswahl besteht, bei der die Positionsangaben irrelevant sind.

ii. Ist der Knoten i in keiner Auswahl- oder Projektionsebene enthalten, werden die Positionsangaben bei der Bestimmung berücksichtigt, d.h. $N_1 = succ(i, \{p_0, ..., p_k\})$.

b) Im Falle einer leeren Menge von Positionsangaben werden alle Nachfolger des Knotens i bestimmt, unabhängig vom konkreten Kantenindex, d.h. $N_1 = succ(i)$.

2. Im zweiten Schritt werden die Labelangaben berücksichtig, falls die Bedingung $L \neq \emptyset$ gilt. Für jeden Knoten $x \in N_1$ werden die die Labelangaben überprüft und die Menge $N_2 \subseteq N_1$ abgeleitet. Zwei Fälle müssen hierbei unterschieden werden:

   a) Der Knoten x ist ein S-Knoten oder P-Knoten. In diesem Fall findet die Zyklus-Zykluspositionsverkettung am "Anfang" einer Auswahl- oder Projektionsebene einer Zykluspositionsauswahl statt. An diesem Punkt hat jedoch noch keine Navigation "in" die Ebene selber begonnen und es wird (semantisch) die Menge aller von der Auswahl betroffenen Zykluspositionen adressiert (unabhängig ob ausgewählt oder nicht). Die Labelangabe bezieht sich an dieser Stelle somit auf die konkreten Teilzyklen und nicht (wie im nachfolgenden Fall) auf die Strukturierungsebenen und damit auf den unmittelbaren Nachfolger. Zur Überprüfung der Labelangabe müssen somit die Blätter der Auswahl- oder Projektionsebene überprüft werden. Das heißt, für den Knoten x muss überprüft werden, ob die Blätter der Ebene (mindestens) ein entsprechendes Label aufweisen, d.h. es muss gelten: $\exists j \in \sigma^x_{leafs} : NL^j \subseteq L$. Nur falls eines der Blätter der Ebene von Knoten x diese Bedingung erfüllt, wird der Knoten x in die Menge N_2 übertragen.

   b) Der Knoten i ist ein H-Knoten. Hier müssen die Labelangaben direkt bei dem Knoten x überprüft werden, d.h. sowohl bei regulären H-Knoten (d.h. H-Knoten, die keinen Strukturierungsknoten in einer Auswahl- oder Projektionsebene entsprechen) als auch für Strukturierungsknoten (d.h. H-Knoten in einer Ebene). In beiden Fällen bezieht sich die Labelangabe der Top-Down Navigation im HSP-Baum auf den Knoten x selber. Der Knoten x wird nur dann in die Menge N_2 übertragen, wenn mindestens eines seiner Label in der Menge der Labelangaben enthalten ist.

3. Im dritten Schritt wird die interne Knotenmenge I_{out} gebildet, in dem die Menge N_2 in die Menge I_{out} übertragen wird. In der Menge N_2 sind alle "internen"

Knoten enthalten, die im nachfolgenden Schritt direkt oder indirekt zu der Menge der externen Knoten führen. Eine Überprüfung der Knoten etwa bezüglich ihres Knotentyps findet jedoch nicht statt. Es ist Aufgabe der "nachfolgenden" Verkettungen diese "Rohmenge" an Knoten entsprechend auszuwerten bzw. gemäß ihrer Semantik zu interpretieren.

4. Im vierten Schritt wird basierend auf der Menge I_{out} die externe Knotenmenge E_{out} ermittelt. Für jeden Knoten $x \in N_2$ gilt es dazu zu überprüfen ob dieser in einer Auswahlebene enthalten ist oder einen "regulären" H-Knoten darstellt (d.h. ein H-Knoten, der einen konkreten Teilzyklus repräsentiert und keinen Strukturknoten einer Auswahlebene).

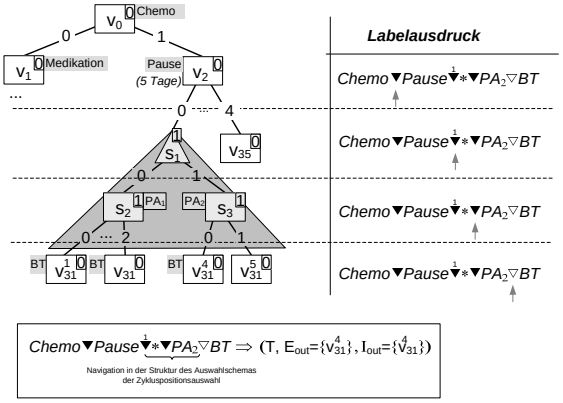
   a) Im ersteren Fall wird für den Knoten x zunächst die zughörige "oberste" Auswahl- oder Projektionsebene E^i ermittelt, d.h. die Ebene, in der der Knoten x oder die Ebene des Knoten x ggfs. enthalten ist. Für die Ebene E^i wird die Menge der transitiven Blätter mit der Menge (aller) Nachfolgerknoten von x geschnitten. Im Ergebnis erhält man alle transitiven Blätter der Ebenen E^i, die von Knoten x aus (nach unten) erreichbar sind. Die transitiven Blätter stellen stets konkrete Teilzyklen dar, die direkt in die Menge E_{out} übertragen werden können.

   b) Im zweiten Fall stellt der Nachfolgerknoten einen regulären Nachfolgerknoten dar, der direkt in die Menge E_{out} übertragen werden kann.

5. Im letzten Schritt wird die erzeugte externe Knotenmenge E_{out} bereinigt. Erst an dieser Stelle "greift" die Semantik der Zyklus-Zyklusspositionsverkettung. Das heißt, es werden alle Knoten aus der Menge gelöscht, die entweder einen 0-Knoten darstellen oder **keinem** Blattknoten entsprechen. Lediglich Blattknoten entsprechen Zykluspositionen, die durch diese Verkettung adressiert werden dürfen.

Das oben skizzierte Vorgehen zur Realisierung der Zyklus-Zyklusspositionsverkettung ist in Abbildung 4.33 für ein Beispiel dargestellt. Kennzeichnend für das Beispiel ist, dass für die erste Zyklusposition, d.h. für Knoten v_{31}, eine Zyklusspositionsauswahl mit strukturierten Auswahlschemas besteht. In der Tabelle ist gut zu erkennen, wie ab den zweiten Navigationsschritt die Top-Down Navigation nur die leere Menge für die externe Knotenmenge erzeugt. Dies ist semantisch korrekt, da es sich ab diesen zweiten Navigationsschritt um eine Navigation entlang der Strukturierung des Auswahlschemas der Zyklusspositionsauswahl handelt. Da es sich hier nur um Strukturierungsebenen handelt und nicht um konkrete Teilzyklen, muss die Navigation an der Stelle die leere Menge zurückliefern. Erst die abschließende Zyklus-Zyklusspositionsverkettung führt zu einer nichtleeren Ergebnismenge. An dieser Stelle erweist sich die Verwendung einer internen und externen Knotenmenge zur Realisierung der Verkettungen erneut als vorteilhaft.

Nachfolgend ist in Algorithmus 4.2 die Realisierung der Zyklus-Zyklusspositionsverkettung dargestellt:



Labelausdruck	Verkettungsfkt.	Erzeugung von I_{out}	Erzeugung von E_{out}
$Chemo▼Pause^{1}▼*▼PA_2▽BT$	▼(T,E_{in}={v_0},I_{in}={v_0}, P=∅,L={*Pause*})	P=∅ ⇒ *succ*(v_0)={v_1,v_2} L≠∅ ⇒ $NL^{v_1} \not\subseteq$ L $NL^{v_2} \subseteq$ L **I_{out} = {v_2}**	$v_2 \notin \sigma^* \Rightarrow E_{out} = \{v_2\}$ **E_{out} = {v_2}** (nach Bereinigung)
$Chemo▼Pause^{1}▼*▼PA_2▽BT$	▼(T, E_{in}={v_2}, I_{in}={v_2}, P={0},L= ∅)	P≠∅ ⇒ *succ*(v_2,{0})={s_1} L=∅ **I_{out} = {s_1}**	$s_1 \in \sigma^{s_1} \Rightarrow E_{out} = \{v_{31}^1, v_{31}^2, v_{31}^3, v_{31}^4, v_{31}^5\}$ **E_{out} = ∅** (nach Bereinigung)
$Chemo▼Pause^{1}▼*▼PA_2▽BT$	▼(T, E_{in}=∅, I_{in}={s_1}, P=∅,L={PA_2})	P=∅ ⇒ *succ*(s_1)={s_2,s_3} L≠∅ ⇒ $NL^{s_2} \not\subseteq$ L $NL^{s_3} \subseteq$ L **I_{out} = {s_3}**	$s_3 \in \sigma^{s_1} \Rightarrow E_{out} = \{v_{31}^4, v_{31}^5\}$ **E_{out} = ∅** (nach Bereinigung)
$Chemo▼Pause^{1}▼*▼PA_2▽BT$	▽(T, E_{in}=∅, I_{in}={s_3}, P=∅,L={*BT*})	$s_3 \in \sigma^{s_1} \Rightarrow$ *succ*(s_3)= {v_{31}^4,v_{31}^5} L≠∅ ⇒ $NL^{v_{31}} \subseteq$ L $NL^{v_{31}} \not\subseteq$ L **I_{out} = {v_{31}^4}**	$v_{31}^4 \notin \sigma^* \Rightarrow E_{out} = \{v_{31}^4\}$ **E_{out} = {v_{31}^4}** (nach Bereinigung)

Abbildung 4.33: Komplexes Beispiel einer Zyklus-Zyklusspositionsverkettung

Algorithmus 4.2 $\triangledown(\mathbf{T_{in}}, \mathbf{E_{in}}, \mathbf{I_{in}}, \mathbf{P}, \mathbf{L}) \mapsto (\mathbf{T_{out}}, \mathbf{E_{out}}, \mathbf{I_{out}})$

Input:

T_{in}	HSP-Baum $T_{in} = (N, ...)$
E_{in}, I_{in}	–
P	Menge der optionalen Positionsangaben $P = \{p_0, ..., p_k\}$
L	Menge der optionalen Labelangaben $L = \{l_0, ..., l_h\}$

Output:

T_{out}	$T_{out} = T_{in}$
E_{out}	$\forall n \in E_{out} : NT^n = \texttt{HNODE} \wedge n =$ Blattknoten
I_{out}	$I_{out} \subseteq N$

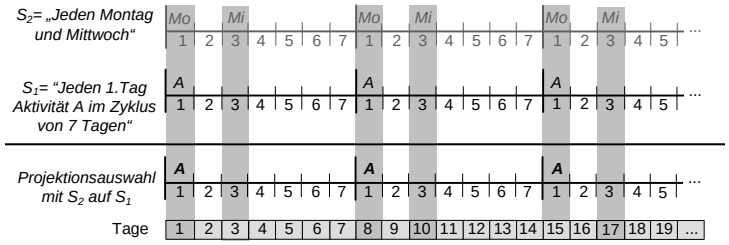
1: **for all** $k \in I_{in}$ **do**
2: **if** $P \neq \emptyset \wedge \nexists j \in N : k \in \sigma^j \vee \pi^j$ **then** {Positionsangaben berücksichtigen}
3: $N_1 = succ(k, p_0, ..., p_n)$
4: **else**
5: $N_1 = succ(k)$
6: **end if**
7: **if** $L \neq \emptyset$ **then** {Berücksichtigung der Labelangaben}
8: **for all** $n \in N_1$ **do**
9: **if** $(\nexists \sigma^n \vee \nexists \pi^n) \wedge (NL^n \subseteq L)$ **then**
10: $InsertElement(N_2, n)$
11: **else if** $(\exists \sigma^n \vee \exists \pi^n) \wedge (\exists i \in \sigma^n_{leafs} \vee \pi^v_{leafs} : NL^i \subseteq L)$ **then**
12: $InsertElement(N_2, n)$
13: **end if**
14: **end for**
15: **end if**
16: $InsertElement(I_{out}, N_2)$ {Erzeugung der internen Knotenmenge I_{out}}
17: **for all** $x \in N_2$ **do** {Erzeugung der externen Knotenmenge E_{out}}
18: **if** $\exists i \in N : (x \in \sigma^i \vee x \in \pi^i) \wedge (\nexists j \in N : i \in \sigma^j \vee i \in \pi^j)$ **then**
19: $tmpE_{out} = SUCC(x) \cap TransitivLeafs^i$
20: **else**
21: $tmpE_{out} = \{x\}$
22: **end if**
23: **for all** $e \in tmpE_{out}$ **do** {Bereingung der externen Knotenmenge E_{out}}
24: **if** $e =$ Blattknoten $\vee NT^e \neq \texttt{ONODE}$ **then**
25: $InsertElement(E_{out}, e)$
26: **end if**
27: **end for**
28: **end for**
29: **end for**

4.3.1.3 Erkennung von Dead-Leafs

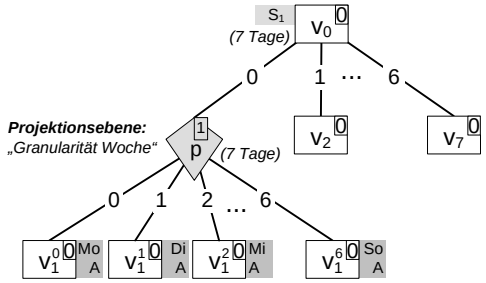
Bei Auswahl- und Projektionsebenen sind wir bisher davon ausgegangen, dass alle Blätter der Ebenen adressiert werden dürfen bzw. die Adressierung semantisch korrekt ist. Allerdings gilt diese Annahme im Allgemeinen nicht uneingeschränkt. Stattdessen existieren Konstellationen, in denen bestimmte Blätter der Ebene nicht adressiert werden dürfen, da diese sonst zu semantisch falschen Ergebnissen führen würden. Dieser Fall tritt jedoch ausschließlich bei Auswahl- oder Projektionsebenen auf und betrifft lediglich einzelne Blätter der Ebenen, die im Folgenden als *Dead-Leafs* bezeichnet werden.

Ein einfaches Beispiel ist in Abbildung 4.34 dargestellt. Es handelt sich um eine Projektionsauswahl des Schemas S_2 "*Jeden Montag (*`Mo`*) und Mittwoch (*`Mi`*) einer* Woche" auf dem Schema S_1 "*Jeden 1. und 2. Tag im Zyklus der Länge=7 findet Aktivität* `A` *statt*". Hier werden alle Aktivitäten A ausgewählt, die auf einen Montag oder Mittwoch fallen. Wie leicht in Abbildung 4.34a anhand der zyklusorientierten Darstellung zu sehen ist, fallen der Mittwoch (`Mi`) und die Aktivität `A` nie zusammen, d.h. die Auswahlmenge der Zykluspositionen, die mit Aktivität `A` und dem Mittwoch korrelieren, ist leer. Das gleiche Beispiel ist in Abbildung 4.34b im HSP-Baum dargestellt. Im HSP-Baum wird hierzu die Projektionsauswahl in Form einer Projektionsebene abgebildet. Kennzeichnend hierbei ist, dass die Projektionsebene zunächst lediglich das Projektionsschema und seine Struktur repräsentiert. Die korrekte Reihenfolge der Projektion, d.h. wie etwa die Aktivität `A` von Schema S_1 mit den Zykluspositionen `Mo` oder `Mi` von Schema S_2 zusammenfallen, wird im HSP-Baum nicht repräsentiert. Stattdessen wir diese Reihenfolge erst bei der Auswertung des Baumes ermittelt (siehe Abschnitt 4.2.3). Aufgrund dieser Abbildung der Projektionsauswahl (aber auch der (mengenwertigen) Zyklus- oder Zykluspositionsauswahl) ergibt sich das Problem, dass etwa mit dem Ausdruck $\mathtt{S}_1\overset{1}{\triangledown}\mathtt{Mi}$ der Knoten v_1^2 grundsätzlich adressiert werden könnte. Semantisch würde dies jedoch bedeuten, dass der Knoten v_1^2 den Knoten darstellt, bei dem die Aktivität `A` auf einen Mittwoch fällt. Wie in Abbildung 4.34a gezeigt, ist dies jedoch semantisch falsch, da die Aktivität `A` niemals auf einen Mittwoch (`Mi`) fällt. Im Bezug auf die Anforderung Korrektheit (siehe Kapitel 3, Abschnitt 3.1) gilt es, semantisch falsche, inkonsistente oder widersprüchliche Angaben zu erkennen bzw. zu vermeiden. Bei der Adressierung von Zyklen oder Zykluspositionen mit Labelausdrücken gilt es somit solche *Dead-Leafs* zu erkennen und deren Adressierung zu vermeiden. Dies stellt sicher, dass lediglich semantisch korrekte Adressierungen vorgenommen werden können.

Die Grundlage der Erkennung von Dead-Leafs bilden die Auswahl- und Projektionsparameter entsprechender Ebenen (siehe Definition 4.1), die die Erkennung von Dead-Leafs ermöglichen. Wir werden im Folgenden hierzu die erst später im Kapitel 5 eingeführte Funktion $getPattern()$ verwenden. Die Funktion $getPattern(T, z_T) = L$ leitet das Wiederholungsmuster für den Baum T ab, d.h. eine geordnete Menge $L \subseteq N$ von Knoten, wobei zusätzlich der Instanziierungspunkt z_T einer periodischen Spezifikation bzw. Wiederholungsmuster berücksichtigt wird (siehe Kapitel 3). In der Funktion werden die pro Ebene hinterlegten Auswahl- oder Projektionsparameter verwendet. Die theoretischen



(a) Projektionsauswahl im zyklenorientierten Modell



(b) Projektionauswahl im HSP-Baum

Abbildung 4.34: Beispiel für Dead-Leafs bei einer Projektionsauswahl

Grundlagen der Funktion werden an dieser Stelle nicht weiter vertieft. Die Charakterisierung und theoretischen Hintergründe von Dead-Leafs werden im Detail in Kapitel 5 vorgestellt (bzw. durch den Korollar 5.2 charakterisiert). Gleiches gilt für die Auswahl- und Projektionsparameter.

Kennzeichnend für die Funktion *getPattern* ist, dass diese auch separat für einzelne Auswahl- oder Projektionsebenen angewendet werden kann. Das heißt, die Funktion erlaubt es, für den Teilbaum, gebildet etwa aus einer Auswahlebene und ihren Blättern, das entsprechende Wiederholungsmuster der Ebene abzuleiten. Im Ergebnis erhält man das Wiederholungsmuster der Auswahl- oder Projektionsebenen, d.h. eine geordnete Menge von Blättern der Ebenen. Aufgrund der in der Funktion verwendeten Auswahl- und Projektionsparameter erfolgt die Bestimmung des Musters in Abhängigkeit von den spezifischen Eigenschaften des restlichen HSP-Baums. Bei Projektionsebenen bedeutet dies etwa, dass das “echte” Wiederholungsmuster der Projektionsebene bestimmt wird (siehe Abschnitt 4.2.3). Bezogen auf das Beispiel in Abbildung 4.34 heißt das, es kann für jedes Vorkommen der von der Auswahl betroffenen Zyklusposition in Schema S_1 angegeben werden, mit welchen Zyklusposition und in welcher Reihenfolge diese mit den Zyklusspositionen von Schema S_2 zusammenfallen. Wesentlich ist, dass Dead-Leafs im “echten” Wiederholungsmuster nicht mehr enthalten sind, da diese durch die Funktion *getPattern*() eliminiert werden. Im Beispiel in Abbildung 4.34b bedeutet dies, dass ausgehend von dem Instanziierungspunkt $z_T = 0$ für Schema S_1, der Funktionsaufruf

$getPattern(\pi^p \cup \pi^p_{leafs}, 0)$ zu dem Muster $L = (v^0_1)$ führt (es wurde der Instanziierungspunkt=0 für die Projektionsebene π^p angenommen). Das heißt, bei der Projektion von Schema S_2 auf die 1.Zyklusposition von Schema S_1 fällt diese nur mit der 1.Position (`Mo`) von Schema S_2 zusammen, was dem Knoten v^0_1 entspricht. Alle weiteren Blätter der Ebene sind Dead-Leafs.

Dead-Leafs sind formal wie folgt definiert, wobei die konkrete Definition, der hierbei verwendeten Funktion $getPattern()$ in Kapitel 5 und in Anhang A zu finden ist.

Definition 4.8 (Dead-Leafs): *Sei $T = (N, ...)$ ein HSP-Baum, E^i eine Auswahl- oder Projektionsebene von T und $z \in \mathbb{Z}$ der Instanzierungspunkt von E^i mit $SP^i = (z, ...)$. Als Dead-Leafs werden alle Blätter der Ebene E^i_{leafs} bezeichnet, für die gilt:*

$$n \in E^i_{leafs} \wedge n \notin getPattern(E^i \cup E^i_{leafs}, z)$$

Die Funktion $getDeadLeafs : N \mapsto 2^N$ mit $getDeadLeafs^i$ ermittelt die Menge der Dead-Leafs der Ebene E^i.

Die Erkennung von Dead-Leafs bei Labelausdrücken bzw. Verkettungen bedeutet, dass lediglich das Enthaltensein von adressierten Knoten in der Menge der Dead-Leafs einer Ebene überprüft werden muss. Es dürfen nur solche Knoten bzw. Blätter bei Auswahl- oder Projektionsebenen adressiert werden, die kein Dead-Leaf einer Ebene darstellen. Diese Bedingung lässt sich leicht überprüfen und wird später bei Labelausdrücken verwendet werden (siehe unten).

4.3.1.4 Definition von Labelausdrücken

Auf Basis der vorgestellten Verkettungsfunktionen sowie den Betrachtungen zu Dead-Leafs kann der bisher informell verwendete Begriff des *Labelausdrucks* formal exakt definiert werden. Ein Labelausdruck repräsentiert eine Menge von Knoten in einem HSP-Baum, die durch die Anwendung verschiedener Verkettungsfunktionen und durch das Eleminieren von Dead-Leafs erzeugt wird. Kennzeichnend für diese Knotenmenge ist, dass diese lediglich solche Knoten enthält, die in der Zyklusalgebra durch entsprechende Operationen manipuliert werden dürfen, d.h. Labelausdrücke erlauben keine Adressierung etwa von Strukturknoten bei Auswahlebenen. Labelausdrücke bilden somit die Grundlage für den semantisch korrekten Umgang bzw. Manipulation von HSP-Bäumen.

In der folgenden Definition zu Labelausdrücken bildet die Funktion $\boxtimes : T \times N \times N \mapsto T \times N \times N$ eine Hilfsfunktion, welche die Dead-Leafs aus einer übergebenen Knotenmenge löscht, d.h. es gilt $\boxtimes(T, N_1, N_2) = (T, N'_1, N'_2)$ mit $N'_1 \subseteq N_1, N'_2 \subseteq N_2$, $N_1, N_2 \subseteq N$ und $\nexists e \in N'_1, N'_2 : e \in \sigma^n_{leaf} \cup \pi^n_{leaf} \wedge e \in getDeadLeafs^n$. Die Knotenmengen N_1 bzw. N_2 stehen hier für die externe bzw. interne Knotenmenge im Ergebnis einer Verkettungsfunktion. Bei Labelausdrücken werden Dead-Leafs sowohl in der internen als

auch in der externen Knotenmenge eliminiert. Basierend auf dieser Hilfsfunktion, sind Labelausdrücke wie folgt definiert:

Definition 4.9 (Labelausdruck): *Sei $T = (N, ...)$ ein HSP-Baum mit Wurzel w und $l \in NL^w$. Des Weiteren sei $\otimes$ eine Verkettungsfunktion aus der Menge an Verkettungen $\otimes \in \{\blacktriangledown, \triangledown\}$ sowie die Mengen $X, Y \in \mathbb{N}$ und $L, M \in 2^{STRING \cup \emptyset}$.*
Ein Labelausdruck $l \overset{X}{\otimes_1} L \otimes_2 ... \overset{Y}{\otimes_n} M$ bezeichnet das Tupel (T', N_1) mit dem HSP-Baum $T' = (N', ...)$ und der Menge von Knoten $N_1 \subseteq N'$, für das gilt:

$$\boxtimes \left(\otimes_n \left((... \boxtimes \left(\otimes_2 \left(\boxtimes \left(\otimes_1 \left(T, \emptyset, \{w\}, X, L \right) \right), ... \right) \right) \right), Y, M \right) \right) = (T', N_1, N_2)$$

Die obige Definition zeigt ebenfalls, wie die im (logischen) zyklenorientierten Modell in Infix-Notation anzutreffenden Labelausdrücke für die Realisierung im HSP-Baum in die entsprechende Präfix-Notation überführt werden. Das Ergebnis eines Labelausdrucks bildet das Tupel (T', N_1) bestehend aus dem HSP-Baum T und der Menge adressierter Knoten N_1. Diese Knotenmenge entspricht der externen Knotenmenge N_1 der letzten Verkettungsfunktion im Ausdruck. Die interne Knotenmenge der Verkettungsfunktion (N_2) wird im Ergebnis des Labelausdrucks nicht weiter berücksichtigt.

In diesem Abschnitt wurde die Realisierung von Verkettungsfunktionen des zyklenorientierten Modells lediglich für einige ausgewählte Funktionen vorgestellt. Alle weiteren Verkettungsfunktionen (siehe Abschnitt 3.4.1) werden in dieser Arbeit nicht weiter berücksichtigt. Dies liegt zum einen daran, dass diese Arbeit auf Durchgängigkeit der Ansätze fokusiert und nicht auf Vollständigkeit. Zum anderen weisen diese Verkettungen ein Problem auf, das es noch zu lösen gilt. Wie wir bereits bei Zyklus-Zyklusspositionsverkettung gezeigt haben, werden im HSP-Baum derzeit keine "wechselnden" Vorgänger- bzw. Nachfolgerbeziehungen von (Teil-) Zyklen oder auch Zyklusspositionen berücksichtigt (siehe Abschnitt 4.3.1.2). Bei der Zyklus-Zyklusspositionsverkettung konnten wir dieses Problem durch entsprechende Einschränkung bei der Anwendung der Verkettung begegnen. (Diese Verkettung ist nur auf Blattebene erlaubt, auf welcher die wechselnden Nachbarschaftsbeziehungen hier keine Rolle mehr spielen.) Bei den anderen Verkettungen ist dies jedoch nicht mehr so einfach möglich. Stattdessen ist die Problematik der Nachbarschaftsbeziehungen inhärent. Einen ersten Lösungsansatz zu diesem Problem haben wir bereits in Abschnitt 4.3.1.2 skizziert.

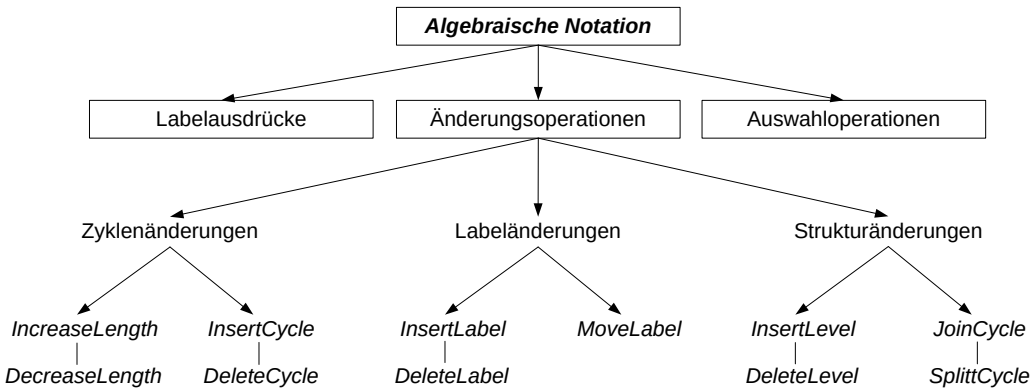


Abbildung 4.35: Übersicht zu den Änderungsoperationen

4.3.2 Änderungsoperationen

Änderungsoperationen dienen im zyklenorientierten Modell der Änderung bzw. dem generellen Aufbau von Periodizitätsschemas. In Bezug auf die Grundelemente des zyklenorientierten Modells können Änderungen an (Teil-) Zyklen, Labelzuordnungen sowie der horizontalen und vertikalen Strukturierungen unterschieden werden. Abbildung 4.35 zeigt die in Kapitel 3 vorgestellten Änderungsoperationen in einer Übersicht.

Im Folgenden wird die Realisierung für ausgewählte Änderungsoperationen auf Basis des HSP-Baums vorgestellt. Voraussetzung hierfür bilden "klassische" Baumoperationen, etwa zum Kopieren, Verschieben oder Löschen von Knoten oder Teilbäumen. Diese Basisoperationen werden in den nachfolgenden Beschreibungen der Änderungsoperationen als Hilfsfunktionen verwendet. Aus Gründen der Vereinfachung wird die Semantik dieser Hilfsfunktionen im Folgenden meist informell vorgestellt. Des Weiteren gehen wir im Folgenden bei allen Änderungsoperationen davon aus, dass die übergebene (d.h. von der Änderungsoperation zu manipulierenden) Knotenmenge stets durch einen Labelausdruck erzeugt wurde. Hierdurch wird sichergestellt, dass durch Änderungsoperationen stets nur die Knoten manipuliert werden können, die mit konkreten Teilzyklen oder Zykluspositionen korrespondieren. Änderungen etwa an Strukturierungsknoten von Auswahl- oder Projektionsebenen sind im zyklenorientierten Modell nicht vorgesehen und werden im HSP-Baum ebenfalls nicht berücksichtigt. Labelausdrücke verhindern, dass entsprechende Knoten fälschlicherweise adressiert und manipuliert werden können.

4.3.2.1 Zyklenänderungen (*DeleteCycle*)

Die Operation *DeleteCycle*() ermöglicht das Löschen von (Teil-) Zyklen (siehe Abschnitt 3.4.2.1). Im zyklenorientierten Modell führt die Löschung eines (Teil-) Zyklus jedoch automatisch dazu, dass auch alle tieferliegenden Zyklen (die eine horizontale und/oder vertikale Strukturierung des zu löschenden (Teil-) Zyklus darstellen) gelöscht

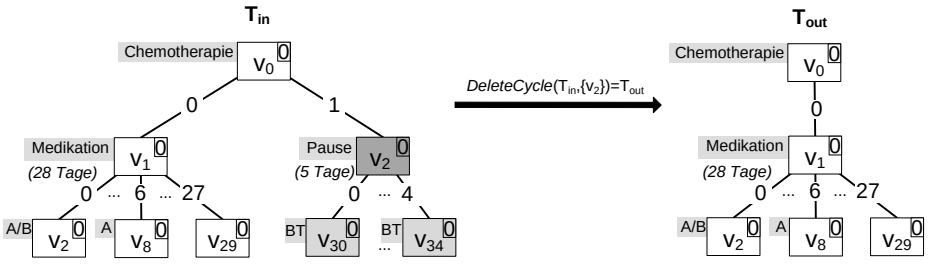


Abbildung 4.36: Einfaches Löschen von Zyklen im HSP-Baum

werden. Bei der Übertragung auf den HSP-Baum entspricht das Löschen eines (Teil-) Zyklus dem Löschen des entsprechenden Knotens bzw. des durch den Knoten als Wurzel gebildeten Teilbaums, der seine tieferliegende Strukturierung repräsentiert. Allerdings gilt es, zwei Fälle zu unterscheiden.

Im ersten Fall handelt es sich bei dem zu löschenden Zyklus bzw. Knoten k um einen Knoten, der kein Blatt einer Auswahl- oder Projektionsebene darstellt. In diesem Fall kann der Knoten k bzw. der Teilbaum mit den Knoten k als Wurzel "einfach" gelöscht werden. Abbildung 4.36 zeigt ein entsprechendes Beispiel.

In zweiten Fall bildet der zu löschende Zyklus ein Blatt einer Auswahl- oder Projektionsebene. Semantisch entspricht dies einem "periodischen" Löschen eines Zyklus auf Basis einer entsprechenden Auswahl, die durch die Ebene repräsentiert wird. Wie in Abschnitt 4.2.4 gezeigt, darf in diesem Fall der Knoten nicht gelöscht werden, da sonst die Semantik der entsprechenden Ebene verändert werden würde. Stattdessen stehen im HSP-Baum für diesen Fall der Knotentyp `ONODE` zur Verfügung. 0-Knoten bilden eine Art "Platzhalter" für den gelöschten Knoten – ohne dabei die Semantik der Ebene zu verändern. In Hinblick auf das Löschen eines Zyklus wird in diesem Fall zwar ebenfalls der Teilbaum mit den zu löschenden Knoten als Wurzel gelöscht. Jedoch bleibt der zu löschende Knoten bzw. die Wurzel des Teilbaums bestehen und es wird ihm der Knotentyp `ONODE` zugewiesen. Abbildung 4.37 zeigt ein entsprechendes Beispiel.

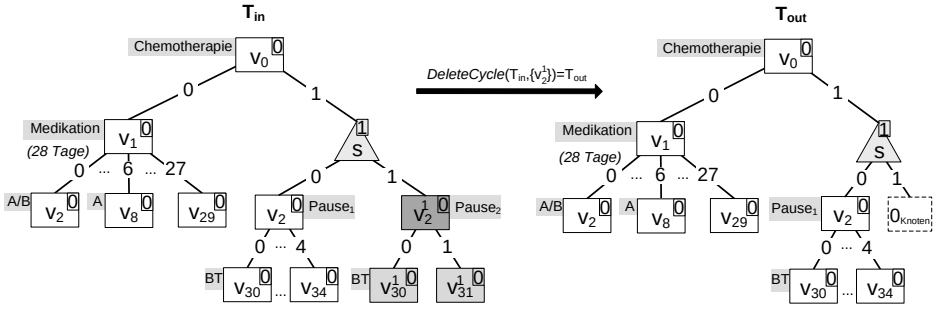


Abbildung 4.37: Periodisches Löschen von Zyklen im HSP-Baum

Algorithmus 4.3 $DeleteCycle(T_{in}, N') \mapsto T_{out}$

Input:

T_{in} HSP-Baum $T_{in} = (N, ...)$

N' Menge der zu löschenden Knoten, für die gilt:
$\exists$ ein Labelausdruck $L: L \mapsto (T_{in}, N')$ und keiner der Knoten stellt eine Zyklusposition dar, d.h. $\forall n \in N' : n$ ist kein Blattknoten

Output:

T_{out} Resultierender HSP-Baum

1: **for all** $n \in N'$ **do**
2: **if** $\exists i \in N : n \in \sigma^i_{leaf}$ **then**
3: $T_{out} = DeleteSubTree(T_{in}, succ(n))$
4: $T_{out} = ChangeNodeType(T_{out}, n, \texttt{ONODE})$
5: **else**
6: $T_{out} = DeleteSubTree(T_{in}, n)$
7: **end if**
8: **end for**
9: **return** T_{out}

Algorithmus 4.3 zeigt die Realisierung der Operation $DeleteCycle()$, die die oben erwähnten beiden Fälle berücksichtigt. Die Hilfsfunktion $DeleteSubTree(T, \{n_1, ..., n_k\})$ löscht die Teilbäume mit den Knoten $n_1, ..., n_k$ als Wurzel aus dem Baum T. Die zweite Hilfsfunktion $ChangeNodeType(T, n, type) = T'$ ändert den Knotentyp eines Knotens n in Baum T, so dass für $n \in T'$ gilt: $NT^n = type$.

4.3.2.2 Labeländerungen ($InsertLabel$)

Die Operation $InsertLabel()$ dient dem Einfügen eines Labels für einen Zyklus oder einer Zyklusposition. Übertragen auf den HSP-Baum, müssen bei der Realisierung der Operation die beiden unterschiedlichen Labelmengen und ihre Semantik berücksichtigt werden. Wie in Abschnitt 4.2.1.2 gezeigt, können die beiden Labelmengen *StructureSet* und *ActivitySet* als anwendungsabhängiges, kontrolliertes Vokabular verstanden werden. Labels aus der Menge *StructureSet* dienen der Bezeichnung von Zyklen, Teilzyklen oder Zyklusspositionen. Labels aus der Menge *ActivitySet* hingegen korrespondieren mit konkreten Aktivitäten (z.B. Medikamentengabe) der realen Welt und können in Anwendungen auch zeitliche Dauern aufweisen. Labels aus der letzteren Labelmenge dürfen deshalb nur für Zyklusspositionen verwendet werden, da diese an anderen Knoten zu Unschärfen führen würden, die in der vorliegenden Arbeit nicht betrachtet werden (siehe Abschnitt 4.2.1.2).

Algorithmus 4.4 zeigt die Realisierung der Operation $InsertLabel()$, wobei die Semantik der beiden Labelmengen in den Vorbedingungen berücksichtigt wird. Hierbei gehen wir

Algorithmus 4.4 $InsertLabel(T_{in}, N', l) \mapsto T_{out}$

Input:

T_{in} HSP-Baum $T_{in} = (N, ...)$

N' Menge an Knoten für die gilt:
(1.) $\exists$ ein Labelausdruck $L: L \mapsto (T_{in}, N')$
(2.) Für das Label l gilt die Strukturregel SR-2

l Label mit $l \in StructuredSet \cup ActivitySet$

Output:

T_{out} Resultierender HSP-Baum

1: **for all** $n \in N'$ **do**
2: $T_{out} = AddLabel(T_{in}, n, l)$
3: **end for**
4: **return** T_{out}

grundsätzlich davon aus, dass neue einzufügende Labels zunächst in einer der beiden Labelmengen *StructureSet* oder *ActivitySet* bekannt gemacht worden sind (im Sinne eines kontrollierten Vokabulars), bevor diese im HSP-Baum eingefügt werden. Darüber hinaus muss überprüft werden, ob, ausgehend von den übergebenen Knoten, das einzufügende Label in einem Pfad nach oben (zur Wurzel) oder "nach unten" bereits existiert. Gemäß Strukturregel SR-2 ist dies nicht zulässig (siehe Abschnitt 4.2.1.2). Die im Folgenden verwendete Hilfsfunktion $AddLabel(T, n, l) = T'$ fügt ein Label in die Labelmenge eines Knotens $n \in T$ hinzu, sodass für $m \in T'$ gilt: $NL^m = NL^n \cup l, m = n$.

4.3.2.3 Strukturänderungen (*SplitCycle*)

Diese Klasse von Operationen dient der Änderung der horizontalen (vertikalen) Strukturierung von Zyklen. Die Operation *SplitCycle* ermöglicht es, einen Zyklus in zwei Teilzyklen aufzuteilen. Übertragen auf den HSP-Baum bedeutet dies, dass der ursprünglich durch einen einzelnen Knoten repräsentierte Zyklus auf zwei Knoten aufgeteilt wird. Dabei gilt es einen neuen Knoten "nach" dem ursprünglichen Knoten einzufügen und die entsprechenden Nachfolger das ursprünglichen Knotens "umzuhängen". Abbildung 4.38 zeigt dieses Vorgehen an einem Beispiel. Der aufzuteilende Zyklus von 28.Tagen wird durch den Knoten v_1 repräsentiert. In dem Beispiel wird mit der Splittoperation der ursprüngliche Zyklus in zwei Teilzyklen zu je 14 Tagen Länge gesplittet. Hierzu wird der Knoten v_1 zunächst kopiert und die Kopie, d.h. der Knoten v_1^1, bezüglich des Kantenindex unmittelbar nach dem Knoten v_1 eingefügt. Im Anschluss daran werden die Knoten mit Kantenindex $14 - 27$ an den Knoten v_1^1 umgehängt. Nachträglich können, wenn gewünscht, den beiden Knoten v_1 und v_1^1 mit Labeloperationen entsprechende Label zugewiesen werden, um diese zu unterscheiden.

Allerdings müssen bei dieser Operation Auswahl- und Projektionsebenen berücksichtigt werden. So ist das Splitten eines (Teil-) Zyklus, der Blatt einer Auswahlebene ist, nicht

zulässig. Ein solches Aufteilen würde die Semantik der Auswahlebenen verfälschen. Dies würde nämlich dazu führen, dass sich der neue Knoten bzw. Teilzyklus im Baum wie ein "ausgewählter" Teilzyklus darstellt. Dies ist jedoch semantisch falsch, da der neue Knoten lediglich ein Teil des ursprünglich ausgewählten (großen) (Teil-) Zyklus darstellt. Erst zusammen bilden die beiden bei der Aufsplittung sich ergebenen Teilzyklen den ursprünglich ausgewählten Zyklus. Um solche semantischen Probleme zu umgehen, ist das direkte Aufsplitten von ausgewählten Zyklen, d.h. von Knoten, die Blätter von Auswahl- bzw. Projektionsebenen sind, nicht zulässig. Stattdessen muss zunächst ein neuer Level unterhalb des aufzusplittenden Zyklus eingefügt werden ($InsertLevel()$). Anschließend kann dann der Zyklus auf dem neuen Level wie gewünscht aufgeteilt werden. Der ursprünglich ausgewählte Zyklus bleibt auf dem höheren Level erhalten und somit die korrekte Semantik der Auswahl- bzw. Projektion.

Algorithmus 4.5 zeigt die Realisierung der Operation $SplitCycle()$. Zu beachten ist, dass die Länge des ursprünglichen Zyklus und in Summe die Längen der beiden resultierenden Teilzyklen gleich ist, d.h. die beiden Teilzyklen entsprechen in Summe wieder der Länge des ursprünglichen Zyklus (siehe Bedingung zu p_1 und p_2). Die in dem Algorithmus verwendeten Hilfsfunktionen weisen die folgenden Semantiken auf:

- Die Hilfsfunktion $CopyNode(n) = m$ kopiert einen Knoten n. Das Ergebnis m erhält eine eigene eindeutige Knoten-ID, weist jedoch bzgl. der Funktionen $ND()$, $NT()$ und $NL()$ des HSP-Baums die gleichen Werte wie n auf.
- Die Funktion $getEdgeIndex(n) = i$ liefert die Kantennummer eines Knotens n bezüglich seines (direkten) Vorgängers m, sodass gilt $succ(m, i) = n$.
- Die Funktion $InsertNode(T, m, n, i) = T'$ fügt in dem Baum T einen neuen Knoten n als Nachfolger von Knoten m entlang der Kante i ein, sodass für den Ergebnisbaum $T' = (N', E', ...)$ gilt: $succ(m, i) = n$ mit $m, n \in N'$.
- Mit der Funktion $MoveNode(T, m, i, j, n) = T'$ werden alle Nachfolger des Knoten m mit Kantenindex $i, ..., j$ auf den Knoten n umgehangen, d.h. sie werden Nachfolger des Knotens n. Die verschobenen Knoten werden hinter dem letzten (vor der Operation bereits bestandenen) Nachfolger mit Kantenindex $k = grad(n) - 1$ von Knoten n angefügt. Die ursprüngliche (relative) Reihenfolge der verschobenen Knoten bleibt von der Operation unberührt.

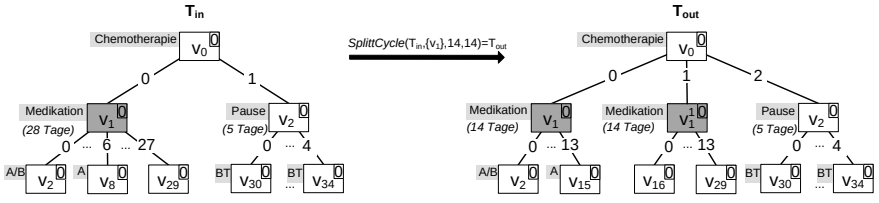


Abbildung 4.38: Aufsplitten eines Zyklus im HSP-Baum

Algorithmus 4.5 $SplitCycle(T_{in}, N', p_1, p_2) \mapsto T_{out}$

Input:

T_{in} HSP-Baum $T_{in} = (N, ...)$

N' Menge der aufzusplittenden Knoten, für die gilt:
(1.) $\exists$ ein Labelausdruck $L: L \mapsto (T_{in}, N')$
(2.) $\forall n \in N' : n$ ist kein Blattknoten, d.h. es sind nur (Teil-) Zyklen enthalten

p_1, p_2 Länge der beiden resultierenden Teilzyklen, für die gilt:
$\forall n \in N' : grad(n) = (p_1 + p_2) - 1$

Output:

T_{out} Resultierender HSP-Baum

1: **for all** $n \in N'$ **do**
2: $new = CopyNode(n)$
3: $new_EdgeIndex = GetEdgeIndex(n) + 1$
4: $InsertNode(T_{out}, pred(n), new, new_EdgeIndex)$
5: $MoveNode(T_{out}, n, (p_1), (p_2 - 1), new)$
6: **end for**
7: **return** T_{out}

4.3.3 Auswahloperationen

Die Klasse der Auswahloperationen ermöglicht es, basierend auf einem Auswahlschema, bestimmte Zyklen, Teilzyklen oder Zykluspositionen eines Periodizitätsschemas auszuwählen. Diese ausgewählten Strukturen können dann etwa mit den vorgestellten Änderungsoperationen manipuliert werden. Abbildung 4.39 zeigt die im zyklenorientierten Modell eingeführten Auswahloperationen in Übersicht. Die Realisierungen dieser Auswahloperationen im HSP-Baum wird in folgenden Abschnitten vorgestellt.

4.3.3.1 Zyklus- und Zykluspositionsauswahl

Die Zyklus- und Zykluspositionsauswahl dient der Auswahl von Zyklen bzw. Zykluspositionen. Kennzeichnend für Auswahloperationen im zyklenorientierten Modell ist, dass das zur Auswahl verwendete Auswahlschema erneut ein (zyklenorientiertes) Periodizitätsschema darstellt (siehe Abschnitt 3.4.3). Im HSP-Baum wird eine Auswahl durch eine entsprechende Auswahlebene dargestellt. Die Auswahlebene repräsentiert hierbei das Auswahlschema der Operation mit seinen evtl. bestehenden Strukturierungen und Ausnahmen. Die Grundlage hierfür bildet das Auswahlschema, das ebenfalls als HSP-Baum beschrieben ist. Ein Auswahlschema wird zunächst als "normales" Periodizitätsschema algebraisch beschrieben und der resultierende HSP-Baum bei der Auswahloperation übergeben. Zur Realisierung der Zyklus- und Zykluspositionsauswahl gilt es, dieses Auswahlschema als entsprechende Ebene in den HSP-Baum bzw. für jeden von der Auswahl

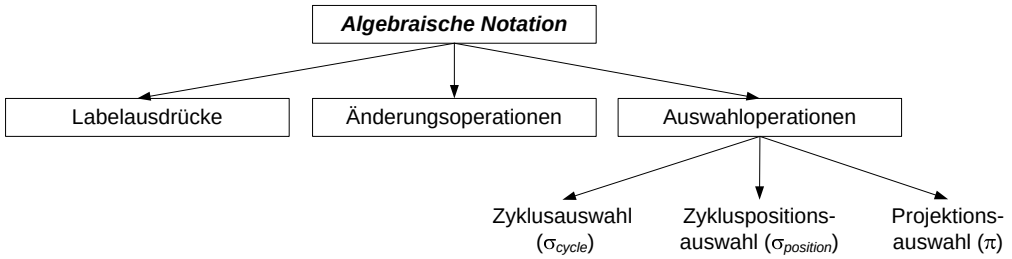


Abbildung 4.39: Übersicht zu Auswahloperationen

betroffenen Knoten einzufügen.

Unabhängig von der konkreten Semantik ist das grundsätzliche Vorgehen bei der Realisierung von Auswahloperationen bei allen Operationen gleich bzw. vergleichbar. Das heißt, es muss zunächst aus dem Auswahlschema eine entsprechende Auswahlebene (bzw. Projektionsebene) gebildet werden, die dann in den von der Auswahl betroffenen HSP-Baum eingefügt wird. Anschließend müssen je nach Semantik der Ebene die Auswahl- bzw. Projektionsparameter bestimmt werden. Wir werden im Folgenden zunächst eine Hilfsoperation für das Erstellen einer Auswahl- oder Projektionsebene einführen. Aufbauend auf dieser werden dann im Anschluss die Realisierung der Zyklus- und Zykluspositionsauswahloperation und die Bestimmung der jeweiligen Auswahlparameter vorgestellt.

Hilfsoperation $InsertSelection$**:** Die Hilfsoperation fügt eine Auswahl- oder Projektionsebene für einen (einzelnen) von der Auswahl betroffenen Knoten ein. Diese Operation bildet die Basis für die Realisierung der Zyklus-, Zykluspositions- und Projektionsauswahl. Mit der Operation $InsertSelection(T_{in}, T_{sel}, n, nodetype)$ wird für den Knoten n im Baum T_{in} je nach Knotentyp *nodetype* eine Auswahl- oder Projektionsebene, basierend auf dem Auswahlschema T_{sel}, eingefügt. In der Operation werden im Wesentlichen die folgenden drei Schritte durchgeführt:

1. Im ersten Schritt müssen die Schachtelungstiefe des Auswahlschemas gemäß der Strukturregel SR-5 (siehe Abschnitt 4.2.6) angepasst werden. Die Regel besagt, dass die Knoten der Auswahlebene eine um eins höhere Schachtelungstiefe als der von der Auswahl betroffene Knoten aufweisen muss. Folglich muss die Schachtelungstiefe aller Knoten des Auswahlschemas um eins erhöht werden. Zusätzlich wird an dieser Stelle der Knotentyp des Wurzelknotens des Auswahlschemas gemäß den übergebenen Knotentyp angepasst. Abbildung 4.40 zeigt ein entsprechendes Beispiel mit Knotentyp `SNODE`.

2. Im zweiten Schritt werden die Blätter des Auswahlschemas durch Kopien des von der Auswahl betroffenen Knotens n bzw. des Teilbaums mit den Knoten n als

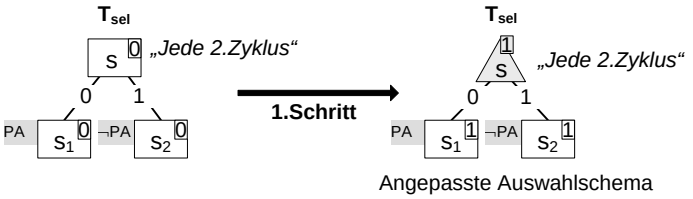


Abbildung 4.40: Anpassung der Schachtelungstiefe und des Knotentyps des Auswahlschemas

Wurzel ersetzt. Dies ergibt sich dadurch, dass die Blätter des Auswahlschemas semantisch die (später) ausgewählten Knoten darstellen. Dies wiederum würde nach dem Einfügen des Auswahlschemas in den HSP-Baum dazu führen, dass jedes Blatt der Auswahlebene mit genau einer Kopie des von der Auswahl betroffenen Knotens korrespondiert. Um diese (redundante) Darstellung bzw. 1:1 Beziehung zu vermeiden, werden die Blätter des Auswahlschemas direkt durch die Kopien des Knotens ersetzt und die Labelzuordnungen auf die Kopien von Knoten n übertragen (siehe Abschnitt 4.2.2). Abbildung 4.41 zeigt ein entsprechendes Beispiel für die Auswahl "*Jede 2.Pause einer Chemotherapie*". Bei dem Knoten v_2 handelt es sich um den von der Auswahl betroffenen Knoten für den entsprechende Kopien erzeugt werden. Diese Kopien ersetzen die Blätter bzw. Knoten s_1, s_2 des angepassten Auswahlschemas aus dem 1.Schritt.

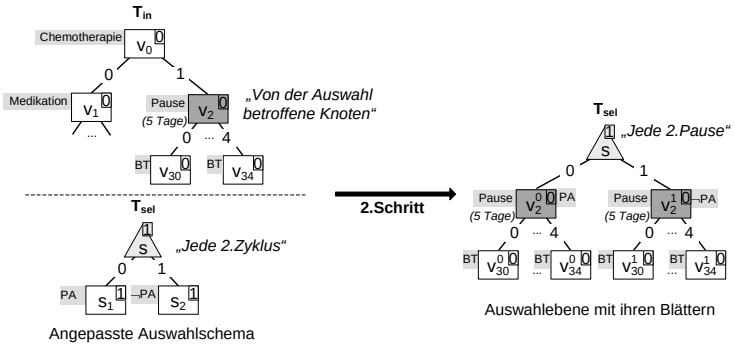


Abbildung 4.41: Ersetzen der Blätter des Auswahlschemas durch Kopien des von der Auswahl betroffenen Knotens v_2

3. Im letzten Schritt wird der von der Auswahl betroffene Knoten n bzw. der Teilbaum mit n als Wurzel im Baum T_{in} durch die im 2.Schritt gebildete Auswahlebene ersetzt. Abbildung 4.42 zeigt dies für das obige Beispiel.

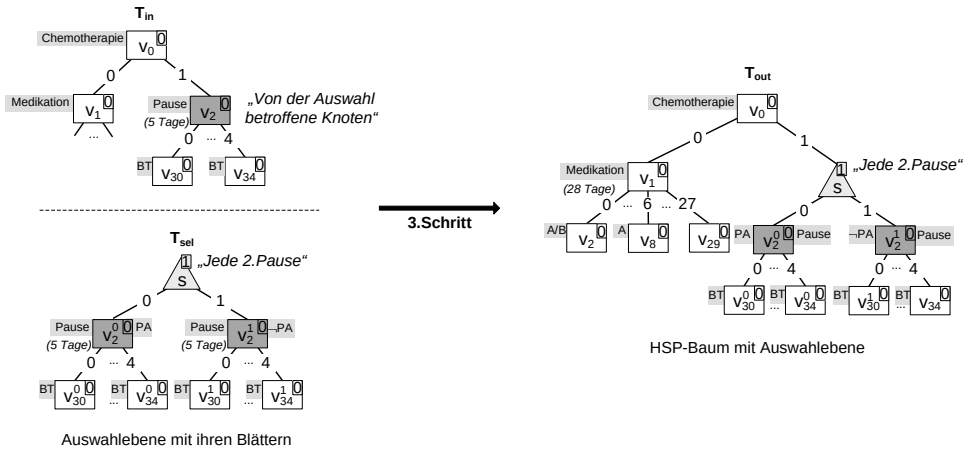


Abbildung 4.42: Einfügen der Auswahlebene mit ihren Blättern in den übergebenen HSP-Baum T_{in}

Algorithmus 4.6 zeigt die Realisierung der Hilfsoperation $InsertSelection$. Die verwendeten Hilfsfunktionen haben folgende Bedeutung:

- Die Funktion $GetSubTree(T, n) = T'$ liefert den Teilbaum T' von T mit Wurzel n.
- Die Funktion $GetRoot(T) = n$ liefert den Wurzelknoten n vom Baum T.
- Die Funktion $InsertSubTree(T_1, T_2, n) = T_e$ ersetzt im Baum T_1 den Knoten $n \in T_1$ (inklusive aller seiner Nachfolger) durch den Teilbaum T_2 (mit Wurzel k). Das heißt, falls im Baum T_1 gilt: $succ(m, i) = n$ und T_n bildet den von Knoten n erreichbaren Teilbaum (bis auf Blattebene), dann gilt nach der Operation im Baum T_e: $succ(m, i) = k$ und T_2 ist von Knoten k erreichbare Teilbaum.

Realisierung der Zyklusauswahl $\sigma_{cycle}(T_{in}, N', T_{sel})$**:** Bei dieser Auswahloperation werden Zyklen oder Teilzyklen des Periodizitätsschemas ausgewählt. Bezüglich der Realisierung im HSP-Baum bedeutet dies, dass in der übergebenen Auswahlmenge N' nur Strukturen vom Typ Zyklus oder Teilzyklus enthalten sein dürfen. Blattknoten, die mit Zyklusspositionen korrespondieren, sind nicht zulässig. Für jeden Knoten der Auswahlmenge muss, basierend auf dem Auswahlschema bzw. dessen korrespondierenden HSP-Baum T_{sel}, eine Auswahlebene im Baum T_{in} eingefügt werden.

Wie erwähnt, bildet die Bestimmung der Auswahlparameter den abschließenden Schritt der Realisierung von Auswahloperationen. Wir haben in Abschnitt 4.2 gezeigt, dass jeder Auswahl- bzw. Projektionsebene ein Tupel von Auswahl- bzw. Projektionsparametern zugeordnet ist. Mit der Funktion SP^n kann für jede Auswahlebene σ^n das entsprechende Tupel bestimmt werden. Diese Parameter sind für die Auswertung entsprechender Ebenen bzw. des zugehörigen HSP-Baums essentiell (siehe Kapitel 5). Allerdings sind zum

Algorithmus 4.6 $InsertSelection(T_{in}, T_{sel}, n, nodetyp) \mapsto T_{out}$

Input:

T_{in}	HSP-Baum $T_{in} = (N, ...)$
T_{sel}	Auswahlschema der Selektion als HSP-Baum $T_{sel} = (N_{sel}, ...)$
n	Von einer Auswahl betroffene Knoten mit $n \subseteq N$
$nodetyp$	Typ der einzufügenden Ebene mit $nodetyp \in \{\texttt{SNODE}, \texttt{PNODE}\}$, d.h. es handelt sich um eine Auswahl- oder Projektionsebene

Output:

T_{out}	Resultierender HSP-Baum

1: $tmp = GetSubTree(T_{in}, n)$ {Teilbaum mit Wurzel n wird zwischengespeichert}
2: $NT^{GetRoot(T_{sel})} := nodetyp$ {Beginn Aufbau der Selektionsebene}
3: **for all** $n_{sel} \in T_{sel}$ **do** {Sicherstellung von Strukturregel SR-5}
4: $\quad ND^{n_{sel}} = ND^{n} + ND^{n_{sel}} + 1$
5: **end for**
6: **for all** $leaf \in T_{sel}$ **do** {Ersetzen aller Blätter $leaf$ von Baum T_{sel} durch den zwischengespeicherten Teilbaum tmp des ausgewählten Knotens n. Bei $InsertSubTree$ wird der zur Adressierung verwendete Knoten (leaf) durch den Wurzelknoten des Teilbaums tmp ersetzt}
7: $\quad InsertSubTree(T_{sel}, tmp, leaf)$
8: **end for**
9: $InsertSubTree(T_{in}, T_{sel}, n)$ {Einfügen der Selektionsebene}
10: **return** T_{in}

Parameter	Zugeordnete Werte
z	Instanziierungspunkt
l_p	Umfang der (geordneten) Auswahlmenge N'
l_A	Länge des Wiederholungsmusters des Auswahlschemas AS
$POS = \{i\}$	Position i des von der Auswahl betroffenen Knotens in der Auswahlmenge

Tabelle 4.2: Auswahlparameter und ihre Semantik

genaueren Verständnis der Parameter tiefergehende theoretische Grundlagen notwendig, die im Detail in Kapitel 5 erläutert werden. An dieser Stelle werden wir daher, lediglich die allgemeine Semantik der Parameter vorstellen und wie diese Parameter bestimmt werden können. Tabelle 4.2 gibt eine Übersicht über die Auswahlparameter und ihre Semantik.

Der Instanziierungspunkt z wird lediglich bei Projektionsebenen benötigt und ist bei Auswahlebenen standardmäßig auf Null gesetzt. In dem Parameter l_P wird der Umfang der Auswahlmenge N' gespeichert. Die Länge des Wiederholungsmusters des Auswahlschemas wird in dem Parameter l_A hinterlegt. Zur Bestimmung des Wiederholungsmusters des Auswahlschemas wird die Funktion $GetPattern()$ verwendet, die in Kapitel 5 eingeführt wird. Die Länge des so ermittelten Musters kann mit der Funktion $GetSize()$ bestimmt werden. In dem letzten Parameter POS wird für den Knoten n (für den eine Auswahlebene erzeugt wird) die absolute Position in der Auswahlmenge N' hinterlegt. Wir gehen davon aus, dass die Auswahlmenge N' geordnet ist (entsprechend der Reihenfolge im HSP-Baum von "links nach rechts"). In dem Beispiel in Abbildung 4.42 ergibt sich für die Auswahlebene σ^s die Auswahlparameter $SP^s = (0, 1, 2, \{0\})$. Das heißt, der Instanziierungspunkt ist standardmäßig=0. Die Auswahlmenge $N' = \{v_2\}$ enthält lediglich den einzelnen Knoten v_2. Die Länge des Musters des Auswahlschemas T_{sel} ist zwei und die Position $POS = \{0\}$ des Knotens v_2 in der Menge N' ist gleich Null (Positionen beginnend bei Null).

Im Folgenden ist die Realisierung der Zyklusauswahl auf Basis der zuvor vorgestellten Hilfsoperation $InsertSelection$ im Algorithmus 4.7 dargestellt. Für jeden Knoten in der übergebenen Auswahlmenge N' werden hierbei eine entsprechende Auswahlebene erzeugt und die Auswahlparameter bestimmt. Die Hilfsfunktion $GetSize(L) = i$ ermittelt die Anzahl der Elemente in der Liste L. Die Hilfsfunktion $GetPositionInSet(L, e) = j$ ermittelt die (absolute) Position eines Elements e in einer (sortierten) Liste L. Die restlichen Hilfsfunktionen wurden bereits im Zusammenhang mit Algorithmus 4.6 vorgestellt (siehe oben).

Realisierung der Zyklusauswahl $\sigma_{position}(T_{in}, N', T_{sel})$: Die Realisierung der Zyklusauswahl ist identisch zur vorgestellten Realisierung der Zyklusauswahl. Lediglich in der Bedingung für die Auswahlmenge N' muss im Gegensatz zur Zyklus-

Algorithmus 4.7 $\sigma_{cycle}(T_{in}, N', T_{sel}) \mapsto T_{out}$

Input:

T_{in} HSP-Baum $T_{in} = (N, ...)$

N' Auswahlmenge $N' \subseteq N$, d.h. geordnete Menge der von der Auswahl betroffenen Knoten, für die gilt:
(1.) $\exists$ ein Labelausdruck $L: L \mapsto (T_{in}, N')$
(2.) $\forall n \in N' : n$ ist kein Blattknoten, d.h. es sind nur (Teil-) Zyklen enthalten

T_{sel} HSP-Baum des Auswahlschemas

Output:

T_{out} Resultierende HSP-Baum

1: **for all** $n \in N'$ **do**
2: $tmp_\sigma = GetRoot(T_{sel})$ {Zwischenspeichern des Wurzelknotens des Auswahlschemas}
3: $T_{out} = InsertSelection(T_{in}, T_{sel}, n, \texttt{SNODE})$ {Einfügen der Auswahlebene}
4: $z = 0$ {Beginn Erzeugen der Auswahlparameter}
5: $l_P = GetSize(N')$
6: $l_A = GetSize(GetPattern(T_{sel}, 0))$
7: $POS = GetPositionInSet(N', n)$
8: $SP^{tmp_\sigma} = (z, l_P, l_A, POS)$
9: **end for**
10: **return** T_{out}

auswahl überprüft werden, ob hier nur Zykluspositionen enthalten sind, d.h. ob jeder Knoten in N' einen Blattknoten darstellt.

4.3.3.2 Projektionsauswahl $\pi(T_{in}, N', T_{sel})$

Die Realisierung einer Projektionsauswahl erfolgt analog zu der oben beschriebenen Realisierung der beiden anderen Auswahloperationen. Lediglich bei der Bestimmung der Projektionsparameter ergeben sich Unterschiede. Wie oben erwähnt, gehen wir an dieser Stelle nur auf die allgemeine Semantik der Projektionsparameter und ihrer Bestimmung ein. Tabelle 4.3 zeigt die Projektionsparameter und ihre Semantik in Übersicht.

Im Unterschied zu den Auswahlparametern spielt bei Projektionsparametern auch die Länge des Wiederholungsmusters des Baums T_{in} zum Zeitpunkt der Projektionsoperation eine Rolle. Im Zusammenhang mit auswahlerhaltenden und nichtauswahlerhaltenden Operationsreihenfolgen (siehe Abschnitt 3.4.3.3) wird durch die Bestimmung der Länge des Wiederholungsmusters die Projektion vor nachträglichen, nichtauswahlerhaltenden Änderungen "geschützt". Das heißt, sollten nachträgliche Operationen zur einer Änderung der Länge des Musters von T_{in} führen (und somit theoretisch zu einer Änderung

Parameter	Zugeordnete Werte
z	Instanziierungspunkt von Schema bzw. Baum T_2
l_p	Länge des Wiederholungsmusters von T_1
l_A	Länge des Wiederholungsmusters von T_2
$POS = \{i_1, ..., i_n\}$	Menge der Positionen des von der Projektion betroffenen Knotens $n \in N'$ innerhalb des Musters von T_1

Tabelle 4.3: Projektionsparameter und ihre Semantik

der Projektionsauswahl), wird bei der späteren Auswertung stets die "alte" Länge in Parameter l_P verwendet. Dies führt dazu, dass die Länge des Musters und somit die Projektion selbst zum Zeitpunkt der Projektionsauswahl "eingefroren" wird. Gleiches gilt für die Parameter z, l_A und POS.

Bei dem Instanziierungspunkt z gehen wir davon aus, dass dieser zum Zeitpunkt der Projektion, für das mit dem Baum T_{sel} korrespondierenden Periodizitätsschema bereits bekannt ist. Die Funktion $GetInstantiationPoint(T_{sel})$ ermittelt den zum Zeitpunkt der Projektionsauswahl bekannten Instanziierungspunkt von Baum T_{sel}. Die Bestimmung der Länge der Wiederholungsmuster von T_{in} und dem Auswahlschema T_{sel} erfolgt analog wie oben über die später eingeführte Funktion $GetPattern()$ und $GetSize()$. Bei der Bestimmung von POS gilt es zu beachten, dass bei einer Projektionsauswahl ein einzelner Knoten n der Projektionsmenge N' im Wiederholungsmuster des Baums T_{in} unterschiedliche Positionen aufweisen kann (Bei den anderen Auswahlebenen kann dies stets nur eine einzelne Position sein). Wie wir später in Kapitel 5, Abschnitt 5.4.2 zeigen werden, ergibt sich dies aufgrund von Null-Knoten. Die unterschiedlichen Positionen des Knotens n im Wiederholungsmuster von T_{in} werden durch die Hilfsfunktion $GetPositionsInSet(N', n)$ bestimmt und im Parameter POS hinterlegt.

Algorithmus 4.8 zeigt die Realisierung der Projektionsauswahl auf Basis der zuvor vorgestellten Hilfsoperation $InsertSelection()$.Für jeden Knoten in der übergebenen Auswahlmenge N' wird eine entsprechende Projektionsebene erzeugt und die Projektionsparameter bestimmt. Die Realisierung der Funktion $getPattern()$ findet sich in Anhang A. Die restlichen Hilfsfunktion wurden bereits zu Algorithmus 4.7 vorgestellt.

Algorithmus 4.8 $\pi(T_{in}, N', T_{sel}) \mapsto T_{out}$

Input:

T_{in} HSP-Baum $T_{in} = (N, ...)$

N' Projektionsmenge $N' \subseteq N$, d.h. geordnete Menge der von der Auswahl betroffenen Knoten, für die gilt:
(1.) $\exists$ ein Labelausdruck $L: L \mapsto (T_1, N')$
(2.) $\forall n \in N' : n = Blattknoten$, d.h. es sind nur Zyklusposition enthalten

T_{sel} HSP-Baum des Auswahlschemas

Output:

T_{out} Resultierende HSP-Baum

1: **for all** $n \in N'$ **do**
2: $tmp_\pi = GetRoot(T_{sel})$ {Zwischenspeichern des Wurzelknotens des Projektionsschemas}
3: $T_{out} = InsertSelection(T_{in}, T_{sel}, n, \texttt{PNODE})$ {Einfügen der Projektionsebene}
4: $z = GetInstantiationPoint(T_{sel})$ {Erzeugen der Projektionsparameter}
5: $l_P = GetSize(GetPattern(T_{in}, 0))$
6: $l_A = GetSize(GetPattern(T_{sel}, 0))$
7: $POS = GetPositionsInSet(GetPattern(T_{in}, 0), n)$
8: $SP^{tmp_\pi} = (z, l_P, l_A, POS)$
9: **end for**
10: **return** T_{out}

4.4 Beispiele

Bei der Einführung der algebraischen Notationsform in Kapitel 3 haben wir die Verwendung der einzelnen Operationen anhand eines praktischen Anwendungsfall beispielhaft dargestellt (siehe Abschnitt 3.4.4). Im Folgenden werden wir das gleiche Beispiel verwenden, um den Zusammenhang zwischen den algebraischen Operationen und den resultierenden HSP-Bäume aufzuzeigen. Die Anwendung der Operationen und die Darstellung der resultierenden HSP-Bäume erfolgt auf Basis des im Rahmen dieser Arbeit erstellten Proof-of-Concept Prototyps (siehe Kapitel 7). In dem ursprünglichen Beispiel handelt sich um eine Chemotherapie mit Medikationszyklus der Länge 28 Tage und nachfolgenden Pausenzyklus der Länge 5 Tage. Aus darstellungstechnischen Gründen reduzieren wir jedoch die Länge des Teilzyklus `Medikation` von 28 Tage auf 10 Tage. Ausgangspunkt bildet das Periodizitätsschema P_1 bestehend aus strukturierten Zyklen mit Label `Chemotherapie` sowie zwei Teilzyklen `Medikation` und `Pause` jeweils mit Länge 1. Den entsprechenden HSP-Baum zeigt Abbildung 4.43.

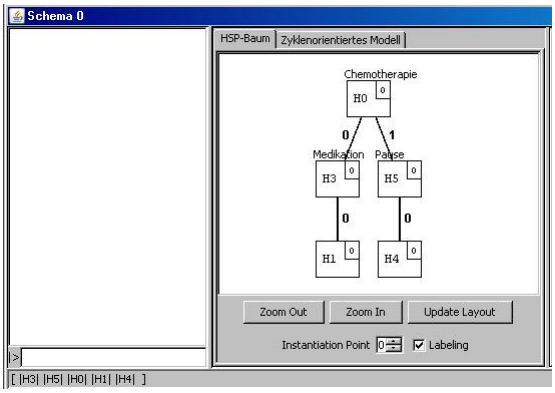


Abbildung 4.43: Initialer HSP-Baum

In den Schritten 1 und 2 werden die Zykluslängen angepasst.

Anpassen der Zykluslängen:
Schritt 1: $P_2 = IncreaseLength(\text{"}\texttt{Chemotherapie▼Medikation}\text{"}, +9)$
Schritt 2: $P_3 = IncreaseLength(\text{"}\texttt{Chemotherapie▼Pause}\text{"}, +4)$

Das Ergebnis der ersten beiden Operationen zeigt Abbildung 4.44. Die linke untere Seite in Abbildung 4.44 zeigt die Befehlszeile des Prototypen, in der die entsprechenden algebraischen Operationen abgesetzt werden können. Im vorliegen Fall etwa ist die im Schritt 2 verwendetet algebraische Operation zu erkennen.[10] Im oberen Bereich der linken Seite werden die von den verwendeten Labelausdrücken betroffenen Knoten im

[10] Die in der logischen Beschreibung notwendige Angabe des konkreten Periodizitätsschema (z.B. P_1 bei der Operation $IncreaseLength(P_1, \text{"}\texttt{Chemotherapie▼Medikation}\text{"}, +27)$ entfällt bei dem dargestellten Prototypen, da dieser jede Operation stets auf den angezeigten HSP-Baum ausführt.

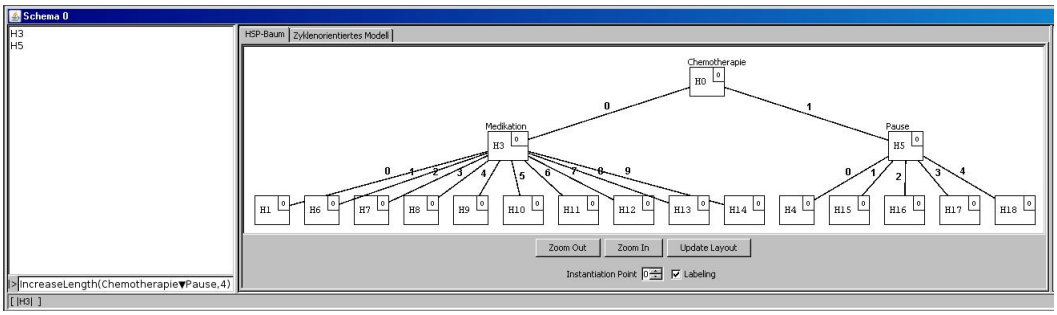


Abbildung 4.44: HSP-Baum nach Anpassen der Zykluslängen (nach Schritt 1 und 2)

HSP-Baum angezeigt, d.h. im Schritt 1 entspricht der Labelausdruck dem Knoten H_3 und im Schritt 2 den Knoten H_5. Aus darstellungstechnischen Gründen verzichten wir in den folgenden Abbildungen (meist) auf die Anzeige der Befehlszeile bzw. der von einem Labelausdruck betroffenen Knoten und stellen lediglich den HSP-Baum dar. In Kapitel 7 werden beide Aspekte nochmal detailiert an konkreten Beispielen erläutert.

Innerhalb des Medikationsteilzyklus finden vom 1.-5. Tag die Gaben von Medikament A (`Med_A`) und am 1. und 8. Tag die Gaben von Medikament B (`Med_B`) statt. Hierfür müssen die jeweiligen Labels an den korrekten Positionen eingefügt werden (vgl. Schritte 3 und 4).

Einfügen von Labels für die Medikationen:

Schritt 3: $P_4 = InsertLabel(\text{"}\texttt{Chemotherapie▼Medikation}\overset{1-5}{\triangledown} *\text{"}, \texttt{Med_A})$

Schritt 4: $P_5 = InsertLabel(\text{"}\texttt{Chemotherapie▼Medikation}\overset{1,8}{\triangledown} *\text{"}, \texttt{Med_B})$

Das Ergebnis von Schritt 3 und Schritt 4 zeigt Abbildung 4.45. Die verwendeten Labels wurde zuvor in einem entsprechenden Repository angelegt, wodurch deren Semantik eindeutig festgelegt wird (siehe Labelmengen *ActivitySet* und *StructuredSet* in Abschnitt 4.2.1.2).

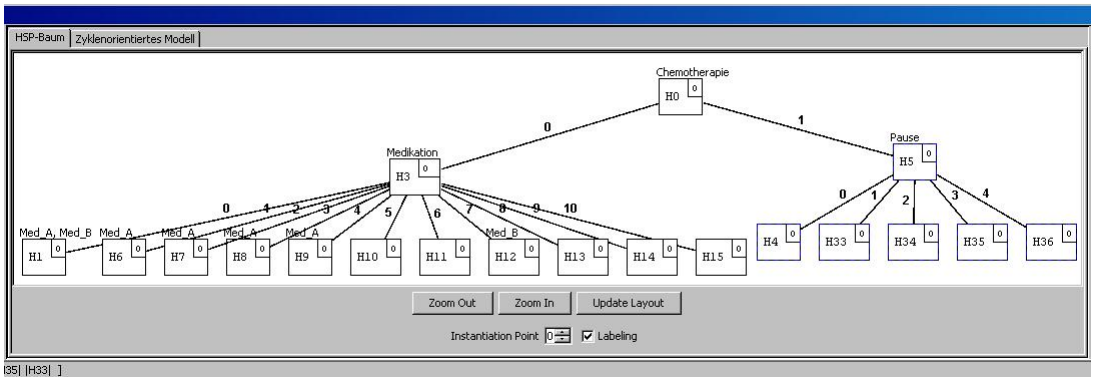


Abbildung 4.45: HSP-Baum nach Einfügen der Labels (nach Schritt 3 und 4)

Während der Chemotherapie erfolgen in jeder zweiten Pause (beginnend bei der zweiten) am letzten Tag ein zusätzlicher Bluttest (`BT`). Um diese (periodische) Ausnahme abzubilden, wird zunächst das Auswahlschema definiert. Ausgangspunkt bildet ein Periodizitätsschema AS mit Zyklen der Länge eins. In Schritt 5.1 wird zunächst die Zykluslänge des Auswahlschemas angepasst und in Schritt 5.2 das Label `PA` an 2.Zyklusposition eingefügt.

Auswahlschema beschreiben:
Schritt 5.1: $AS_2 = IncreaseLength(" * ", +1)$ (Anpassen der Zykluslänge)
Schritt 5.2: $AS_3 = InsertLabel(" * \overset{2}{\triangledown} * ", \texttt{PA})$ (Einfügen von Labels)

Der entsprechende HSP-Baum für das Auswahlschema zeigt Abbildung 4.46.

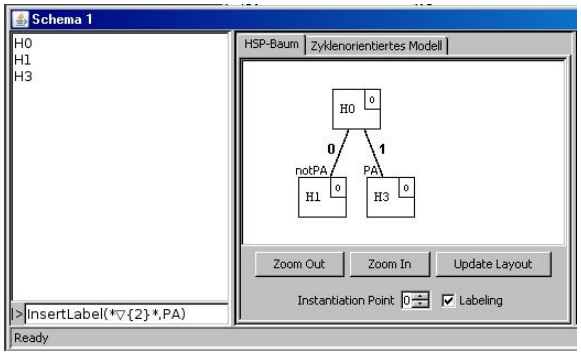


Abbildung 4.46: HSP-Baum des Auswahlschemas (nach Schritt 5.1 und 5.2)

Nach Definition des Auswahlschemas (AS_3) wird im nächsten Schritt eine Zyklusauswahl vorgenommen (vgl. Schritt 6) und für die ausgewählten Pausezyklen (`PA`) der Bluttest am letzten Tag einer Pause (5.Zyklusposition) eingefügt (vgl. Schritt 7).

Periodische Zyklusauswahl vornehmen:
Schritt 6: $P_6 = \sigma_{cycle}(\text{"}\texttt{Chemotherapie▼Pause}, AS_3)$

Einfügen von Labels für den Bluttest:
Schritt 7: $P_7 = InsertLabel(\text{"}\texttt{Chemotherapie▼Pause▼PA}\overset{5}{\triangledown} * \text{"}, \texttt{BT})$

Das Ergebnis der Zyklusauswahl zeigt Abbildung 4.47.

Die Zyklusauswahl wird im HSP-Baum durch die entstandene Auswahlebene S_7 mit ihren Blättern H_8 und H_{14} repräsentiert. Das Blatt bzw. der Knoten H_{14} repräsentiert alle Pausen, die von der Auswahl betroffen sind (`PA`) bzw. der Knoten H_8 alle Pause, die nicht von der Auswahl betroffen sind (`notPA`).

Im ursprünglichen Beispiel in Abschnitt 3.4.4 besteht für das Periodizitätsschema P_7 zusätzlich der Zusammenhang, dass alle Gaben von Medikament `B,` die auf einen Sonntag

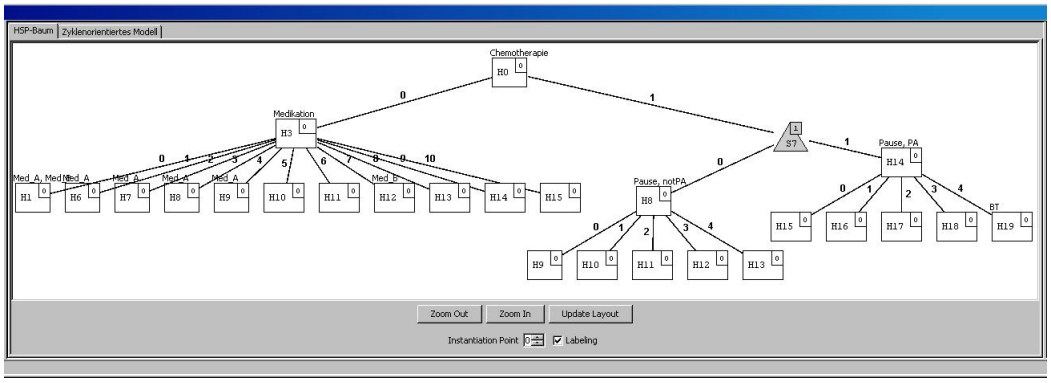


Abbildung 4.47: HSP-Baum nach erfolgter Zyklusauswahl (nach Schritt 6 und 7)

fallen, um einen Tag nach hinten (auf Montag) verschoben werden sollen. Wie jedoch in Abschnitt 4.3.1 gezeigt, ist die dafür notwendige Horizontale Verkettungsfunktion ($\overset{\pm x}{\triangleright}$) für den HSP-Baum derzeit noch nicht realisierbar. Stattdessen wandeln wir das Beispiel so ab, dass alle Bluttests (`BT`), die auf einen Sonntag (`So`) fallen, ersatzlos entfallen sollen. Hierfür wird das Periodizitätsschema für "*Woche*" benötigt, das eine Woche auf Basis von Tagen beschreibt. Ausgangspunkt bildet das Periodizitätsschema G_1 bestehend aus Zyklen `Wochen` mit Zykluslänge = 1.

Periodizitätsschema G_{Woche} beschreiben:
Schritt 8.1: $G_2 = IncreaseLength(" * ", +6)$
Schritt 8.2: $G_{Woche} = InsertLabel(...InsertLabel(" * " \overset{1}{\triangledown} " * ", \texttt{Mo})...), " * " \overset{7}{\triangledown} " * ", \texttt{So})$

Der HSP-Baum für das Periodizitätsschema G_{Woche} zeigt Abbildung 4.48.

Im nächsten Schritt wird eine Projektionsauswahl von "Woche" auf die Bluttests (`BT`) in

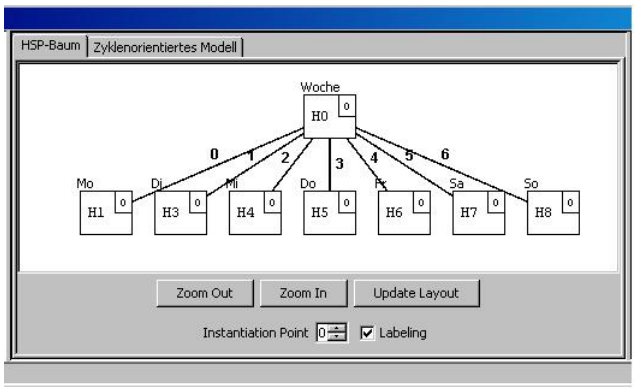


Abbildung 4.48: HSP-Baum des Periodizitätsschemas G_{Woche} (nach Schritt 8.1 und 8.2)

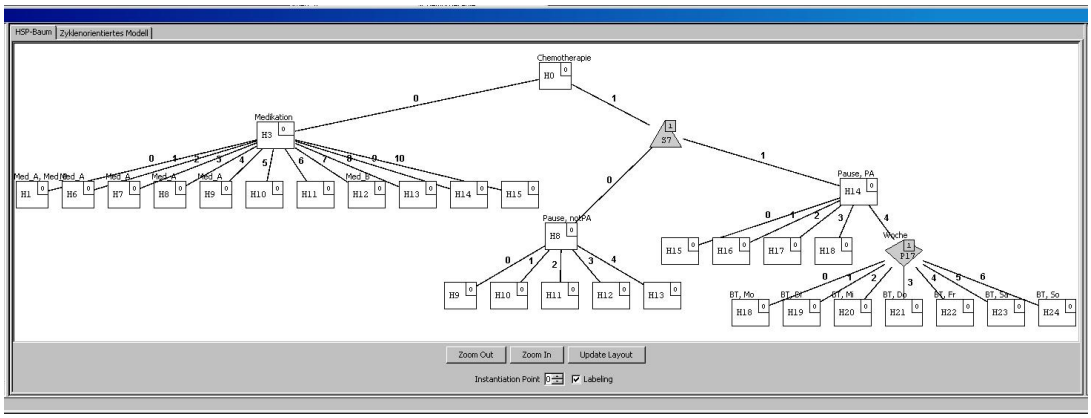


Abbildung 4.49: HSP-Baum nach erfolgter Projektionsauswahl (nach Schritt 9)

Schema P_7 benötigt, um alle von der Löschung betroffenen Bluttests zu kennzeichnen. Für das Schema G_{Woche} wird der Instanziierungspunkt 0 angenommen.

Projektionsauswahl vornehmen:
Schritt 9:$P_8 = \pi(\text{"}\texttt{Chemotherapie}\blacktriangledown\texttt{Pause}\triangledown\texttt{BT}\text{"}, G_{Woche}, 0)$

Abbildung 4.49 zeigt den HSP-Baum nach erfolgter Projektionsauswahl.

Mit dem Labelausdruck "$\texttt{Chemotherapie}\blacktriangledown\texttt{Pause}\blacktriangledown\texttt{PA}\overset{5}{\triangledown}\texttt{So}$" kann nach erfolgter Projektionsauswahl auf alle Bluttests `BT` zugegriffen werden, die auf einen Sonntag fallen. Mit der Top-Down Verkettung "...$\blacktriangledown\texttt{PA}\overset{5}{\triangledown}\texttt{So}$" werden zunächst alle von der Auswahl betroffenen Pausen (`PA`) adressiert (vgl. Schritt 6 und 7), da nur dort ein Bluttest stattfindet. Im Anschluss wird mit der Zyklus-Zyklusspositionsverkettung "...$\texttt{PA}\overset{5}{\triangledown}\texttt{So}$" die 5.Zyklusposition der von der Auswahl betroffenen Pausen (`PA`) adressiert, da ein Bluttest stets am letzten Tag einer Pause und somit an der 5.Zyklusposition stattfindet. Durch die letzte Angabe "...$\overset{5}{\triangledown}\texttt{So}$" werden nur die jeweils fünften Zyklusspositionen adressiert, die zusätzlich das Label Sonntag (`So`) aufweisen. Das Vorkommen beider Labels an einer Zyklusposition kennzeichnet einen Bluttest (`BT`), der an einem Sonntag (`So`) stattfindet. Im HSP-Baum in Abbildung 4.49 führt der Labelausdruck zu den Knoten H_{24}. Für diesen Knoten muss das entsprechende Label `BT` gelöscht werden, um aus semantischer Sicht alle Bluttests zu löschen, die auf einen Sonntag fallen (vgl. Schritt 10). Das Ergebnis von Schritt 10 zeigt Abbildung 4.50.

Labels löschen:
Schritt 10: $\mathit{DeleteLabel}(\text{"}\texttt{Chemotherapie}\blacktriangledown\texttt{Pause}\blacktriangledown\texttt{PA}\overset{5}{\triangledown}\texttt{So}\text{"}, \texttt{BT})$

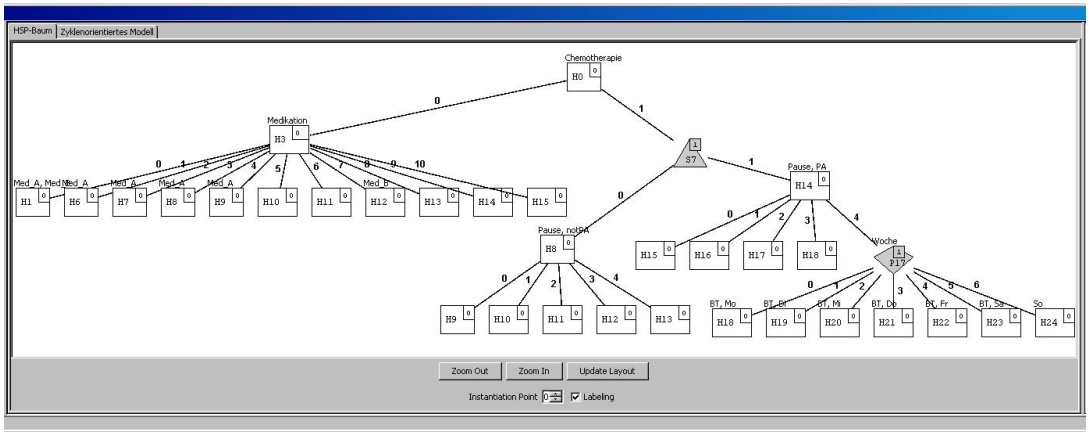


Abbildung 4.50: HSP-Baum nach Löschen aller Bluttests (`BT`), die auf einen Sonntag (`So`) fallen (nach Schritt 10)

4.5 Ausblick

In den bisherigen Betrachtungen sind wir stets davon ausgegangen, dass ein einzelner HSP-Baum mit einer einzelnen Granularitätsebene wie etwa *Tage* oder *Stunden* korrespondiert. Allerdings besteht an eine umfassende logische Beschreibung von Periodizitätsschemas die Anforderung nach *Granularitätsfähigkeit* (siehe Abschnitt 3.1.3). Die Verwendung unterschiedlicher Granularitäten ist weit verbreitet in der Praxis. In Abhängigkeit von Anwendung und Einsatzgebiet ergibt sich zusätzlich die Notwendigkeit, mit verschiedenen Kalendern umgehen zu können (z.B. gregorianischer Kalender, persischer Kalender, islamischer Kalender, Mondkalender). In diesem Zusammenhang sollten periodische Spezifikationen, die mehrere (unterschiedliche) Granularitätsebenen verwenden, abbildbar sein (z.B. "*Alle 2 Stunden an jeden 3.Tag*").

Grundsätzlich kann das zyklenorientierte Modell bzw. der HSP-Baum der Anforderungen nach Granularitätsfähigkeit begegnen. Hierbei können über die horizontale und vertikale Strukturierung von Zyklen bzw. die korrespondierenden Ebenen im HSP-Baum entsprechende Granularitäten abgebildet werden. Abbildung 4.51 zeigt ein Beispiel in dem die Granularitätsebenen *Jahre*, *Monate*, *Tage* und *Stunden* abgebildet sind. Für die Schaltjahresproblematik im Monat `Februar` wurden entsprechende Auswahlebenen eingeführt. Auf Basis des Baumes könnten ebenfalls periodische Angaben auf mehren Granularitätsebenen abgebildet werden, wie etwa "*Jeden 2.Januar an jeden 3.Tag*". Hierzu muss lediglich eine entsprechende Auswahlebenen für `Januar` (Knoten v_2) und für jeden Tag im Januar (Knoten $v_2, ..., v_{32}$) eine (abhängige) Auswahlebene eingefügt werden. Abbildung 4.52 zeigt den für das Beispiel resultierenden HSP-Baum bzw. einen entsprechenden Ausschnitt.

Auch wenn somit im zyklenorientierten Modell bzw. in HSP-Bäumen periodische Angaben über mehrere Granularitätsebene grundsätzlich abgebildet werden können, so

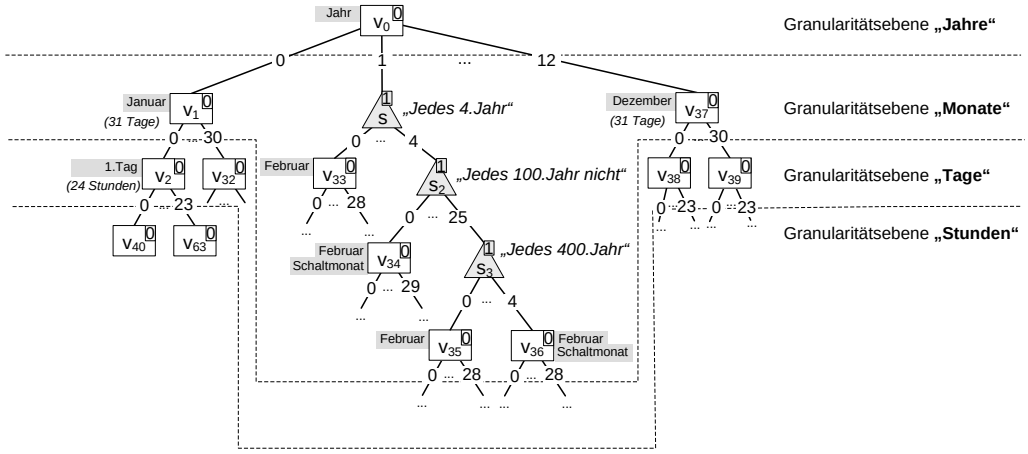


Abbildung 4.51: HSP-Baum mit verschiedenen (semantischen) Granularitätsebenen

empfiehlt sich die dargestellte Vorgehensweise nicht. Das Hauptproblem an dieser Stelle ist, dass hier zwei semantisch verschiedene Aspekte – Granularitäten und periodische Spezifikationen – miteinander vermischt werden. So weisen zwar sowohl Granularitäten als auch periodische Spezifikationen bezüglich ihrer Periodizitätseigenschaften identische Anforderungen auf, allerdings bestehen im Zusammenhang mit Granularitäten und ihrer Handhabung in Anwendungen weitreichende zusätzliche Anforderungen [8, 36, 41, 50, 66]. Um diesen zu begegnen hat es sich bewährt, einzelne Granularitäten als eigenständige Elemente abzubilden [8, 36, 66, 65]. Die Granularitäten sind jedoch zusätzlich in einer “Granularitätshierarchie” eingebunden. Diese beschreibt die bzw. den aufeinander aufbauenden Charakter und Eigenschaften der Granularitäten und stellt ein Metamodell für den (semantisch korrekten) Umgang mit diesen dar. Eine solche Hierarchie mit ihren einzelnen Granularitäten kann als Abbildung eines spezifischen Kalenders verstanden werden. Abbildung 4.53 zeigt auf der linken Seite zwei verschiedene Granularitätshierarchien bzw. zwei verschiedene Kalender.

Für die Realisierung eines solchen Metamodells bzw. seiner einzelnen Granularitäten kann der HSP-Baum wiederum als Basis dienen. Jede Granularität bildet in diesem Fall einen separaten HSP-Baum, in der die Zykluspositionen der “oberen” Granularität die Zyklen seiner (unmittelbar) tieferen Granularität repräsentieren. Jede Granularitätsebene ist hierbei vollständig beschrieben, d.h. dass etwa die Granularität *Monate* bereits inklusive seiner Schalttage repräsentiert wird. Die Grundidee hierbei ist, dass eine Granularität auf höherer Ebene durch die Zyklen der (unmittelbar) tieferen Granularität gebildet wird. Das heißt, spezifische Besonderheiten der unteren Granularität müssen nicht bei der Bildung der höheren Granularität berücksichtigt werden. Stattdessen wird die tiefere Granularität einfach als eine (theoretisch unendliche) Menge von Elementen bzw. Zyklen verstanden, auf der die höhere Granularität seine Struktur abbildet. Beispielsweise wird die Granularität *Jahre* lediglich als Zyklus der Länge 12 (Monate) repräsentiert. Die Ausnahmen bzw. Abweichungen der Monatslängen im Monat `Februar`

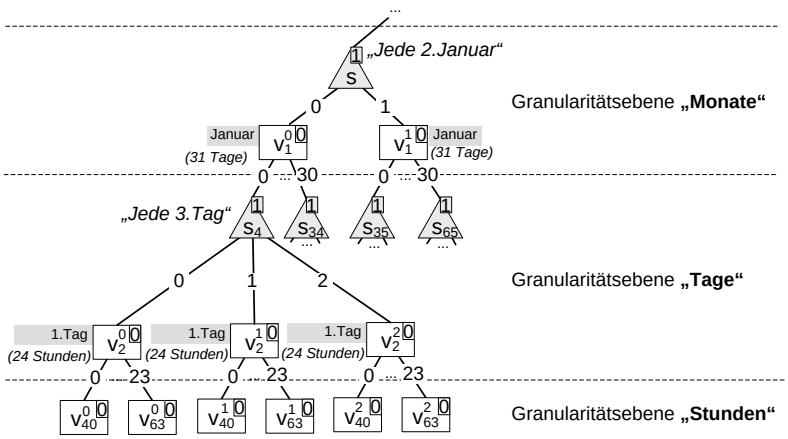


Abbildung 4.52: HSP-Baum mit periodischen Angaben auf mehreren Granularitätsebenen

sind bereits auf Monatsebene abgebildet (siehe Abbildung 4.53).

Die in der vorliegenden Arbeit eingeführten Projektionsoperation π ermöglicht es darüber hinaus, dass Granularitäten auch transitiv aufeinander aufbauen können. Beispielsweise müssen zur Berücksichtigung von Sommer und Winterzeit die beiden Granularitäten *Woche* und *Monat* aufeinander projiziert werden (*"Letzte Sonntag im Monat März"*). Mit der Projektionsoperation können solche Abhängigkeiten zwischen Granularitäten abgebildet werden. Die Eigenschaften des transitiven Aufbaus kann auch auf unterschiedliche Kalender übertragen werden. Wie in Abbildung 4.53 gezeigt, können so etwa ein alternativer Kalender parallel zum gregorianischen Kalender verwendet werden. Somit können unter anderem Granularitäten bereitgestellt werden, die bestimmte Eigenschaften in beiden Kalendern berücksichtigen, wie etwa "*Jeder Sonntag an dem Vollmond ist*" (alternativer Kalender = Mondkalender) oder "*Jeder Monat an dem Ramadan (islamische Fastenmonat) ist*" (alternativer Kalender = Islamischer Kalender). In existierenden Arbeiten werden solche Granularitäten bzw. die Bildung solcher Granularitäten bisher nicht berücksichtigt.

Auf Basis eines solchen Metamodells für Granularitäten und ihre Ebenen können dann periodische Spezifikationen über mehrere Granularitätsebenen hinweg abgebildet werden. Eine solche Spezifikation würde dann aus verschiedenen Granularitätsebene bestehen und für jede einzelne Ebene eine entsprechende periodische Angabe aufweisen. Das oben verwendete Beispiel "*Jeden 2.Januar an jeden 3.Tag*" etwa basierte auf den beiden Granularitäten G_{Monate} und G_{Tag}, wobei auf der Granularitätsebene $Monate$ die periodische Angabe "*Jeden 2.Januar*" und auf der Ebene $Tage$ "*Jeden 3.Tag*" repräsentiert wird. Die Angaben werden dann durch HSP-Bäume repräsentiert, die die entsprechende Granularitätsebenen als Basis haben. Das heißt, in einem solchen Modell besteht eine periodische Spezifikation aus einen oder mehreren HSP-Bäumen und den zugehörigen

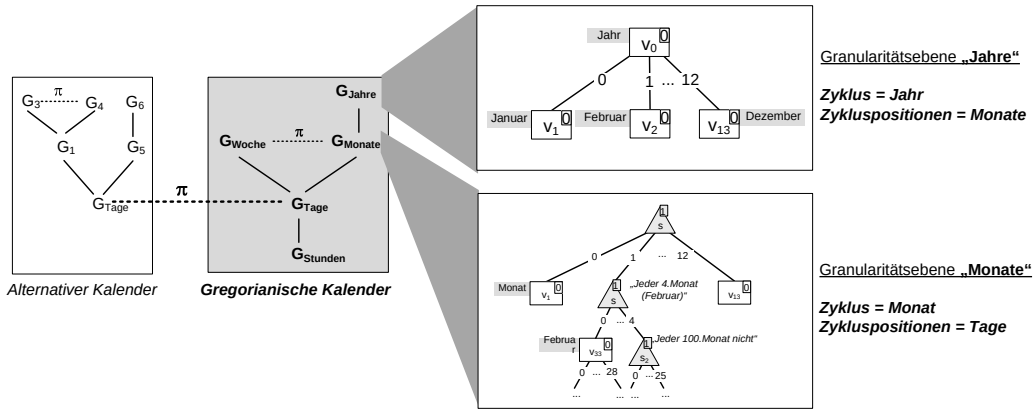


Abbildung 4.53: Grundidee separater Granularitätsebenen und Kalender in Verbindung mit dem HSP-Baum

Granularitätsebenen. Auf Basis der Granularitätshierarchie kann so auch die semantische Korrektheit solcher Spezifikationen bzgl. der verwendeten Granularität und ihrer Eigenschaften untereinander überprüft werden.

Für die Realisierung sowohl des Granularitätsmodell als auch granularitätsabhängiger periodischer Spezifikationen bildet das zyklenorientierte Modell bzw. der HSP-Baum (als dessen endliche Repräsentation) ein solides Fundament. Wir haben die Grundideen vorgestellt, wie die in der vorliegenden Arbeit eingeführten Modelle und Theorien zur Realisierung solch semantisch “höherwertiger” Modell bzw. Strukturen verwendet werden können. Dies gilt sowohl für die grundsätzliche Abbildbarkeit solcher Strukturen als auch für deren Auswertungsfähigkeit. Um jedoch der Anforderung nach Granularitätsfähigkeit bzw. die mit der Handhabung unterschiedlicher Granularität verbundenen zusätzlichen Anforderungen vollumfänglich begegnen zu können, bedarf es umfassender, weiterführender Forschungsarbeiten, die über den Fokus der vorliegenden Arbeit hinausgehen.

4.6 Zusammenfassung

In diesem Kapitel haben wir den HSP-Baum als kompakte Repräsentationsform des zyklenorientierten Modells eingeführt. Die Grundidee des HSP-Baums ist, dass anstelle der theoretisch unendlichen Menge an Zyklen des zyklenorientierten Modells lediglich das Wiederholungsmuster der Zyklenreihenfolgen repräsentiert wird. Der vorstehend beschriebene HSP-Baum entspricht somit einer semantisch reichhaltigen Repräsentation eines Wiederholungsmusters. Hierbei werden neben Zyklusreihenfolgen auch bestehende Strukturierungen von Zyklen sowie Ausnahmen in kompakter und endlicher Form dargestellt.

Wir haben zeigen können, dass die unterschiedlichen Strukturen des zyklenorientierten Modells durch entsprechende Knotentypen im HSP-Baum abbildbar sind. So können etwa die horizontalen und vertikalen Strukturierungen von Zyklen durch korrespondierende Baumebenen und H-Knoten abgebildet werden. Im Zusammenhang mit Auswahloperationen zeigt sich, dass das Ergebnis solcher Operationen im HSP-Baum über Auswahl- oder Projektionsebenen (S-Knoten/P-Knoten) kompakt repräsentiert werden kann. Im Unterschied zum zyklenorientierten Modell erlaubt es der HSP-Baum zusätzlich, dass die Struktur des bei Auswahloperationen verwendeten Auswahlschemas vollständig erhalten bleibt. Diese Struktur des Auswahlschemas kann dann etwa bei Labelausdrücken zur Adressierung etwa von (Teil-) Zyklen bzw. korrespondierenden Knoten verwendet werden. Für jeden Knotentyp des HSP-Baums sind Strukturregeln angegeben, die in der Summe die Korrektheit des HSP-Baums und somit der Abbildung des zyklenorientierten Modells sicherstellen.

Der HSP-Baum dient auch als Grundlage für eine präzise formale Beschreibung der algebraischen Notationsform des zyklenorientierten Modells. Für ausgewählte Operationen der algebraischen Notationsform wurden entsprechende Realisierungen der Operationen im HSP-Baum vorgestellt. Es zeigt sich, dass Labelausdrücken und ihrer Realisierung im HSP-Baum hierbei eine wichtige Rolle zukommt, da mit diesen der Anforderung nach *Korrektheit* begegnet werden kann. Entsprechende Ausdrücke stellen zum einen sicher, dass nur solche Knoten im HSP-Baum adressiert werden können, die semantisch mit (Teil-) Zyklen und Zyklspositionen korrespondieren. Im Baum enthaltene "Hilfsknoten" (etwa Strukturknoten von Auswahlebenen) werden durch solche Ausdrücke nicht adressiert. Zum anderen stellt die eingeführte Erkennung und Bereinigung von *Dead-Leafs* sicher, dass semantisch falsche, inkonsistente oder widersprüchliche Angaben erkannt und korrigiert werden können.

Im Zusammenhang mit der Anforderung *Granularitätsfähigkeit* zeigt sich, dass ein HSP-Baum dieser Anforderung ebenfalls grundsätzlich begegnen kann. Allerdings kann der HSP-Baum nur als Grundlage für eine solche Anforderung verstanden werden. Die in Verbindung mit Granularitätsfähigkeit einhergehenden zusätzlichen Anforderungen, bedürfen erweiterter Repräsentationsmodelle und weiterführende Untersuchungen, die über den Rahmen dieser Arbeit hinausgehen.

Teil III

Operationale Unterstützung für Periodizität

5 Auswertung von HSP-Bäumen

Eine adäquate operationale Unterstützung ist unerlässlich für einen effektiven Umgang mit periodischen Spezifikationen. In Kapitel 2 wurden hierzu unterschiedliche Möglichkeiten der Realisierung entsprechender Operationen diskutiert. Im Ergebnis zeigte sich, dass die Verwendung von Wiederholungsmustern diejenige Repräsentationsform darstellt, die den Anforderungen einer operationalen Unterstützung am besten begegnen kann (vgl. Abschnitt 2.4). Deshalb haben wir für das zyklenorientierte Zeitmodell die Mindestanforderung nach Ableitbarkeit von Wiederholungsmustern gestellt (siehe Abschnitt 3.1). Dieselbe Anforderung besteht folglich auch für das HSP-Baummodell, d.h. die von uns gewählte kompakte und berechenbare Repräsentation des zyklenorientierten Zeitmodells.

Im vorliegenden Kapitel wird eine Auswertung des HSP-Baums vorgestellt, die eine Ableitung von Wiederholungsmuster ermöglicht. In Abschnitt 5.1 skizzieren wir die Grundidee zur Ableitung von Mustern. Abschnitt 5.2 zeigt das allgemeine Vorgehen zur Ableitung von Wiederholungsmustern. In Abschnitt 5.3 werden die theoretischen Grundlagen der Auswertung von Projektionsebenen vorgestellt. In diesem Zusammenhang stellen wir die Theorie der periodischen Projektion vor, die wesentlich für die Auswertung von HSP-Bäumen ist. In den Abschnitten 5.4 und 5.5 werden diese theoretischen Grundlagen auf die Auswertung von Projektions- bzw. Auswahlebenen übertragen. Eine detaillierte algorithmische Beschreibung der Ableitung von Wiederholungsmustern erfolgt schließlich in Abschnitt 5.7.

5.1 HSP-Bäume und Wiederholungsmuster

Wie zu Beginn von Kapitel 4 dargestellt, handelt es sich bei HSP-Bäumen um eine semantisch reichhaltige Repräsentation eines Wiederholungsmusters. Das heißt, die Baumrepräsentation ermöglicht es, horizontale und vertikale Strukturierungen sowie Labelzuweisungen von Zyklen und Teilzyklen durch die Baumstruktur und die unterschiedlichen Knotentypen abzubilden (siehe Abschnitt 4.2.1).

Allerdings werden zur Realisierung der operationalen Unterstützung "einfachere" Wiederholungsmuster benötigt. Im Wesentlichen stehen Wiederholungsmuster im Vordergrund, die lediglich die tatsächlich resultierenden Zeitwerte einer periodischen Spezifikation repräsentieren. Die horizontalen und vertikalen Strukturierungen spielen bei solchen Wiederholungsmustern lediglich eine untergeordnete Rolle.[1] Stattdessen wird

[1] Entsprechende Strukturierungen und Labels wurden eingeführt, um die Beschreibung periodischer

durch das Wiederholungsmuster lediglich die kleinste Periode eines Periodizitätsschemas repräsentiert, d.h. der kleinste Zeitraum des Schemas, nach dem sich die möglichen Abstände zwischen Werten sowie deren Reihenfolgen wiederholen. In Abschnitt 2.3.3 sind verschiedene Repräsentationsformen für solche Wiederholungsmuster vorgestellt. Eine Repräsentationsform sind *lineare Musterrepräsentationen*, in dem Wiederholungsmuster als Bit- bzw. Stringvektoren repräsentiert werden. Die Länge der Vektoren bilden die Periodenlänge des Musters und die einzelnen Elemente des Vektors die Musterpositionen. Bei entsprechender Interpretation verläuft ein solches Wiederholungsmuster über einen (diskreten) Wertebereich, wobei die Positionen des Musters mit den Werten des Wertebereichs korrespondieren. Die konkreten Belegungen der Positionen, d.h. die Stützstellen des Musters, treffen Aussagen, ob der zur Position korrespondierende Wert in der Spezifikation enthalten ist oder nicht (siehe Abschnitt 2.3.3.1). Wie wir in Kapitel 6 noch zeigen werden, kann für diese Repräsentationsform eine kompakte Repräsentation und effiziente operationale Realisierung angegeben werden, weswegen diese im Folgenden im Vordergrund stehen.

Die Auswertung des HSP-Baums führt zu einfachen, für die operationale Realisierung benötigten Wiederholungsmustern. Hierbei werden die (nicht benötigten) Strukturierungen der Teilzyklen "aufgelöst". Vereinfacht ausgedrückt, führt die Auswertung dazu, dass die vielen Teilzyklen mit ihren Ausnahmen und Ebenen in einen "großen", unstrukturierten Zyklus überführt werden. Die Länge des resultierenden Zyklus entspricht der Länge des Wiederholungsmusters und die Zykluspositionen den Musterpositionen. Im Ergebnis führt die Auswertung zu einer geordnete Menge von Blattknoten $(v_1, ..., v_k)$, die als großer unstrukturierten Zyklus bzw. als Wiederholungsmuster der Länge k verstanden werden kann. Die Knoten der Menge entsprechen den Zyklusposition bzw. den Musterposition im Wiederholungsmuster. So wird etwa der Knoten v_1 in der obigen Menge als erste Musterposition und der Knoten v_k als k-te Position des Wiederholungsmusters der Länge k verstanden.

Die Erzeugung dieser geordneten Menge von Blattknoten wird mit der Auswertung des HSP-Baums im Folgenden vorgestellt. Abbildung 5.1 zeigt beispielhaft den Zusammenhang zwischen von HSP-Baum und Wiederholungsmustern. Das Beispiel enthält hierbei eine Auswahlebene bzw. Auswahl auf dem Teilzyklus `Pause` auf, d.h. es besteht folgende (Teil-) Zyklusreihenfolge: `Medikation` → `Pause/¬PA` → `Medikation` → `Pause/PA`. Die Auswertung des HSP-Baums führt zu der dargestellten (geordneten) Menge an Blattknoten $L = (v_3, v_4, ..., v'_{33})$. Diese Menge L entspricht einem Wiederholungsmuster der Länge=64 (2x Zyklus `Medikation` von 28 Tagen + 1x `Pause/¬PA` von 5 Tagen + 1x `Pause/PA` mit 3 Tagen), in der jeder der enthaltenen Knoten einer Musterposition entspricht.

Die bei der Auswertung entstehende Menge an Blattknoten bildet jedoch nur die Grundlage für die Erzeugung von Wiederholungsmustern, die direkt bei der operationalen

Spezifikationen für den Anwender zu vereinfachen. Für eine operationale Unterstützung auf Basis von Wiederholungsmustern stellen diese Strukturen jedoch keine Voraussetzung dar.

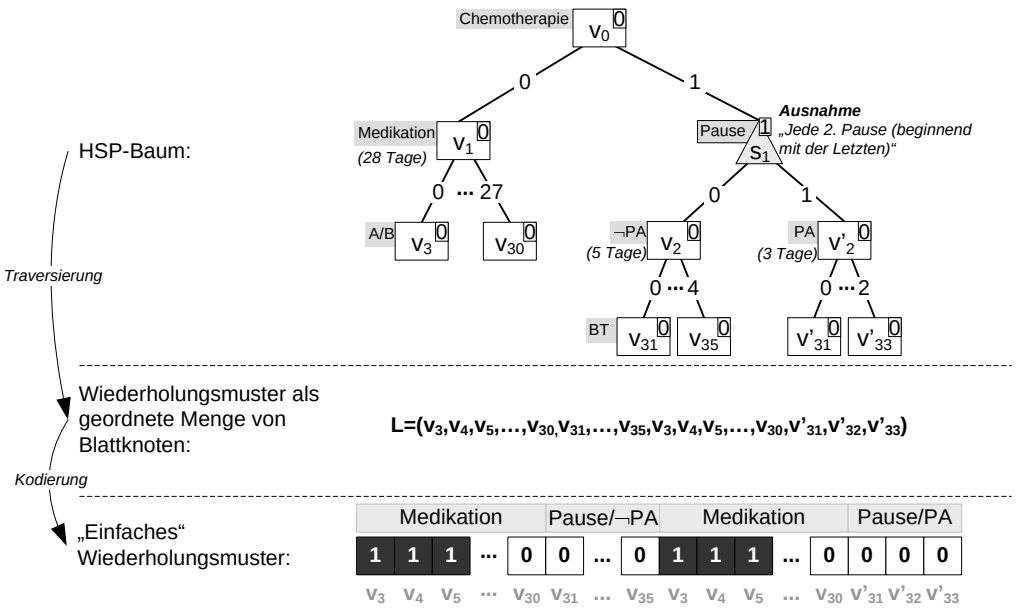


Abbildung 5.1: Zusammenhang HSP-Baum und Wiederholungsmuster

Unterstützung verwendet werden. Bei diesen werden einfache Wiederholungsmustern wie etwa Bitvektoren benötigt (siehe Abschnitt 2.3.3), die mit den Werten “1” und “0”, die mit dem Muster korrelierende Wertemengen kodieren. Im Ergebnis der Auswertung erhalten wir zunächst nur eine Menge von Blattknoten mit verschiedenen Labelzuweisungen. Für diese gilt es, die oben angesprochene Kodierung abzuleiten. So kann etwa eine “1”-Belegung durch jedes (beliebige) Label eines Knotens erfolgen, oder auch nur für einige ausgewählte Labels. In Abbildung 5.1 führen nur Knoten, die das Label `A` enthalten zu einer “1”-Belegung. Welche Labels eines Knotens wie in das Muster übertragen bzw. im Muster kodiert werden, ist anwendungsabhängig und wird hier nicht weiter vertieft. Unabhängig von der konkreten Kodierung, ist die Grundvoraussetzung stets, die bei der Auswertung resultierende Menge an Blattknoten.

5.2 Allgemeine Vorgehen zur Ableitung von Wiederholungsmustern

In diesem Abschnitt werden der Algorithmus und seine verschiedenen logischen Schritte vorgestellt. Hierzu wird zunächst eine grobe Übersicht zu den grundsätzlichen, logischen Schritten bei der Ableitung von Wiederholungsmustern gegeben. Im Anschluss daran werden die verschiedenen logischen Schritte im Detail erläutert.

5.2.1 Kurzübersicht

Die Ableitung von Wiederholungsmustern basiert im Kern auf einer "klassischen" Level-Order Traversierung von (geordneten) Bäumen [68]. Das heißt, bei dieser Traversierungsform werden, ausgehend vom Wurzelknoten auf jedem Level des Baumes, die Knoten vom äußerst linken zum äußerst rechten Knoten "durchlaufen". Die Nachfolger der Knoten eines Levels bilden den nächsttieferen Level, der durchlaufen wird. Die Auswertung von HSP-Bäumen erfolgt im Wesentlichen analog. Auch hier wird der Baum von Level zu Level durchlaufen. Im Gegensatz zum klassischen Vorgehen, müssen hierbei die verschiedenen Knotentypen des HSP-Baums beim Übergang von einem Level zum nächst tieferen Level gesondert berücksichtigt werden.

Abbildung 5.2 zeigt den grundsätzliche Ablauf des Algorithmus. Die äußere Schleife stellt die Leveltraversierung des HSP-Baums dar. Diese terminiert, wenn alle Knoten eines Levels nur noch Blattknoten darstellen. Der erste Schritt im Algorithmus bildet die Bestimmung der Knoten des nächsttieferen Levels (ausgehend von einem Level oder vom Wurzelknoten). Die nachfolgenden Schritte 2.1 – 2.3 dienen der Auswertung der unterschiedlichen Knotentypen (S-Knoten, P-Knoten, 0-Knoten) in einem Level.

Im Schritt 2.1 werden die individuellen Wiederholungsmuster einer jeden Auswahl- oder Projektionsebene gebildet. Die ermittelten Wiederholungsmuster aller in einem Level enthaltenen Ebenen zusammen mit den H-Knoten des Levels bilden die *kompakte Leveldarstellung*.

Im Schritt 2.2 werden die individuellen Wiederholungsmuster für Projektionsebenen und bestimmte Formen von Auswahlebenen transformiert, da diese nach Schritt 2.1 noch nicht vollständig bzw. korrekt sind.

Den Abschluss bildet das "Entpacken" der kompakten Leveldarstellung in Schritt 2.3, was zum vollständigen und korrekten Wiederholungsmuster eines Levels führt.

Der Algorithmus beginnt dann von vorne, d.h. für die ermittelten Levelmuster, die Knoten des nächsttieferen Levels zubestimmen (1.Schritt). Hierbei muss lediglich der Sonderfall von gekoppelten Auswahl- oder Projektionsebenen berücksichtigt werden. In diesem Fall wird das ermittelte Levelmuster bereits als neuer Level verstanden und durchläuft lediglich die dargestellte innere Schleife der Auswertung. Der 1. Schritt des Algorithmus entfällt bei diesem Sonderfall.

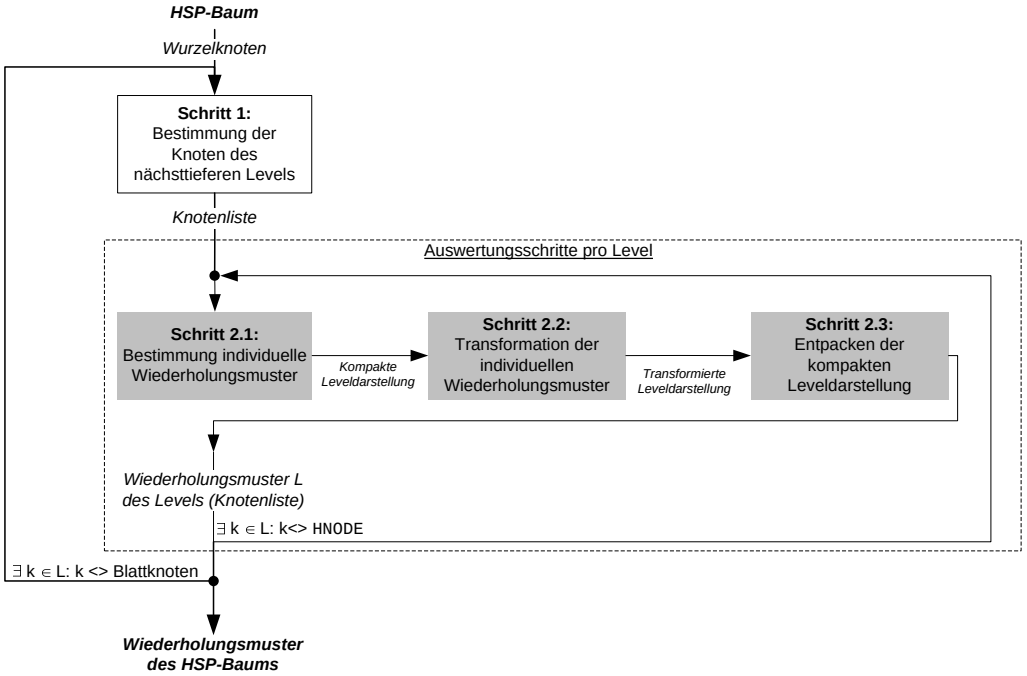


Abbildung 5.2: Ablaufübersicht des Algorithmus zur Ableitung von Wiederholungsmustern

Der Algorithmus terminiert, wenn alle Knoten in einem Levelmuster Blattknoten entsprechen. Dieses resultierende Levelmuster entspricht dem Wiederholungsmuster des HSP-Baums.

In den folgenden Abschnitten werden die einzelnen Schritte des Algorithmus im Detail vorgestellt.

5.2.2 Schritt 1: Angepasste Level-Order Traversierung

Die Traversierung von HSP-Bäumen ist vergleichbar mit der "klassischen" Level-Order Traversierung in geordneten Bäumen [68]. Bei dieser Traversierungsform werden, ausgehend vom Wurzelknoten, auf jedem Level des Baumes die Knoten vom äußerst linken zum äußerst rechten Knoten "durchlaufen". Die Nachfolger der Knoten eines Levels bilden das nächsttiefere Level, das durchlaufen wird. Diese Traversierung wird solange fortgesetzt, bis alle Knoten eines Levels nur noch Blattknoten entsprechen. Abbildung 5.3 zeigt ein Beispiel für eine klassische Level-Order Traversierung von Bäumen. Das Ergebnis der Traversierung ist die folgende Knotenreihenfolge $[1-2-3-4-5-6-7-8]$ bzw. als Reihenfolge pro Level ergeben sich $L_0 = [1]$, $L_1 = [2-3]$, $L_2 = [4-5-6]$, $L_3 = [7-8]$.

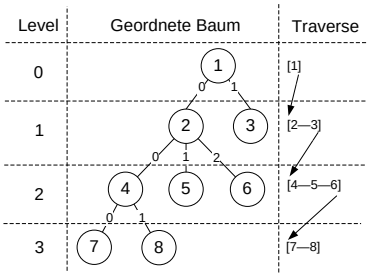


Abbildung 5.3: Level-Order Traversierung von geordneten Bäumen

Zur Ableitung von Wiederholungsmustern auf Basis von HSP-Bäumen wird grundsätzlich ein analoges Vorgehen verwendet. Das heißt, es wird eine levelbasierte Traversierung des HSP-Baums verwendet. Ausgehend vom Wurzelknoten bzw. einer Liste von Knoten eines bestimmten Levels (Levelknotenliste) werden für den Übergang zum nächsttieferen Level die entsprechenden Nachfolgerknoten bestimmt. Dieses Vorgehen wird solange fortgesetzt, bis die Blätter des Baumes erreicht worden sind. Im Ergebnis führt die Traversierung zu einer Liste von Blattknoten, die das Wiederholungsmuster des HSP-Baums darstellen.

Allerdings unterscheidet sich die Traversierung im Vergleich zum klassischen Vorgehen aufgrund der unterschiedlichen Knotentypen des HSP-Baums. Aufgrund ihrer z.T unterschiedlichen Semantiken müssen die verschiedenen Knotentypen beim Übergang von einem Level zum nächsttieferen Level gesondert berücksichtigt werden (siehe später). Ein Unterschied zur klassischen Traversierung zeigt sich bereits im Umgang mit Blattknoten. Bei HSP-Bäumen müssen Blattknoten beim Übergang von einem Level auf den nächsttieferen Level übertragen werden. Ein solches Vorgehen ist in der klassischen Level-Order Traversierung nicht zu finden (siehe Knoten 3, 5 oder 6 in Abbildung 5.3). Die Übertragung von Blattknoten auf die tieferen Level ist notwendig, da Blattknoten eines HSP-Baums stets vom Typ `HNODE` sind und semantisch jeweils einer Zyklusposition entsprechen. Bei der Traversierung müssen diese Blattknoten über alle Levels (nach unten) mitgeführt werden, da sonst im Ergebnis eine Zyklusposition im Zyklus bzw. eine Position im resultierenden Wiederholungsmuster fehlen würde. Abbildung 5.4 zeigt das Verhalten bei Blattknoten beim Übergang von einem Level zum nächsttieferen Level für ein Beispiel. Dieses Beispiel entspricht einem HSP-Baum, der lediglich aus H-Knoten besteht und eine Chemotherapie repräsentiert, bestehend aus dem strukturierten Teilzyklus `Medikation` und Teilzyklus `Pause`. Im klassischen Level-Order Algorithmus würde keine Übertragung der Knoten $v_{31}, ..., v_{35}$ beim Übergang von Level zwei auf Level drei stattfinden. Diese würden somit im Wiederholungsmuster fehlen. Bei HSP-Bäumen hingegen, werden diese Knoten an den nächsttieferen Level mit übergeben.

Das Beispiel aus Abbildung 5.4 entspricht dem einfachsten Fall der Traversierung, d.h. einem HSP-Baum, der nur aus H-Knoten besteht. Da im vorliegenden Fall weder Auswahl- noch Projektionsebenen berücksichtigt werden müssen, entspricht die dargestellte Kno-

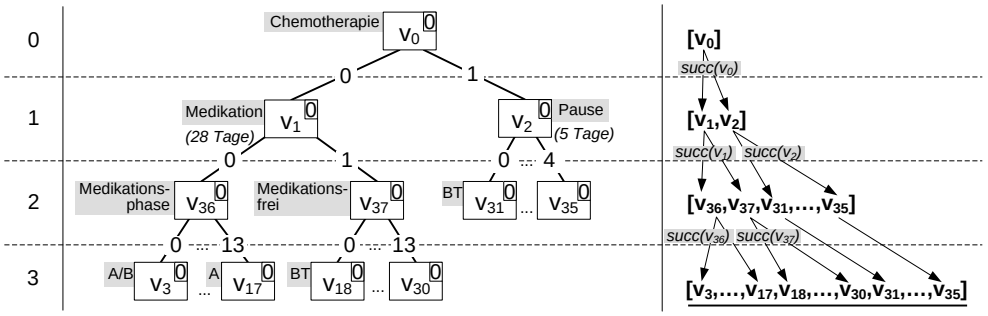


Abbildung 5.4: Beispiel für die Level-Order Traversierung bei HSP-Bäumen

tenliste auf Level drei bereits dem Ergebnis der Auswertung, d.h. die geordnete Menge von Blattknoten $L = [v_3, \ldots, v_{17}, v_{18}, \ldots, v_{30}, v_{31}, \ldots, v_{35}]$ bildet das Wiederholungsmuster des HSP-Baums. Bei HSP-Bäumen, die nur aus H-Knoten bestehen, ergibt sich das Wiederholungsmuster alleine durch die Anwendung der Nachfolgerfunktion *succ*() (siehe Tabelle 4.1).[2]

In den anderen Fällen, d.h. Levels mit Auswahl- und Projektionsebenen, müssen diese Ebenen gesondert berücksichtigt werden. Diese gesonderte Berücksichtigung findet in den Schritten 2.1 – 2.3 des Algorithmus statt und wird in den nachfolgenden Abschnitten im Detail vorgestellt.

5.2.3 Schritt 2.1: Bestimmung der Wiederholungsmuster von Auswahl- und Projektionsebenen

Die Berücksichtigung von Auswahl- und Projektionsebenen erfolgt bei einer levelbasierten Traversierung des HSP-Baums von Level zu Level. Den ersten Schritt bildet die Ableitung des individuellen Wiederholungsmusters einer Auswahl- oder Projektionsebene. Hierbei müssen jedoch die verschiedenen Grundformen von Auswahl- und Projektionsebenen pro Level berücksichtigt werden, d.h. einfache, gekoppelte und geschachtelte Auswahl- bzw. Projektionsebenen (siehe Abschnitt 4.2.2 und Abschnitt 4.2.3). Im Folgenden geben wir einen Überblick zu der bei der Traversierung zu berücksichtigen Formen. Auch wenn im Folgenden hauptsächlich Auswahlebenen im Vordergrund stehen, gelten die Betrachtungen uneingeschränkt auch für Projektionsebenen.

[2]Wir werden später zeigen, dass zur Bestimmung der individuellen Wiederholungsmuster von Auswahl- und Projektionsebenen der vorliegende Algorithmus zur Bildung von Wiederholungsmustern rekursiv aufgerufen wird. HSP-Bäume, die ausschließlich aus H-Knoten bestehen, bilden hier das Abbruchkriterium der Rekursion (siehe Abschnitt 5.7).

5.2.3.1 Level mit einer einzelnen Ebene

Den Grundfall bilden Bäume mit einer einzelnen, einfachen (d.h. nicht-geschachtelten oder gekoppelten) Auswahl- oder Projektionsebene. Abbildung 5.5 zeigt ein Beispiel für einen HSP-Baum mit einfacher Auswahlebene. Das Beispiel entspricht einer Chemotherapie mit der Auswahl "*Jede 2.Pause beginnend bei der Letzten*" für den Teilzyklus `Pause`. Bei der Auswertung muss auf Level 1 die Auswahlebene berücksichtigt werden. Ebenfalls zu erkennen ist, dass für die Zählung der Level lediglich die H-Knoten relevant sind, d.h. Auswahl- und Projektionsebenen führen nicht zu einem "neuen" Level. Im vorliegenden Beispiel sind etwa die Auswahlebene σ^s und ihrer Blätter dem Level 1 zugeordnet. Eine vereinfachte Darstellung des von der Auswahlebene betroffenen Levels für das vorliegende Beispiel zeigt Abbildung 5.6a. Der Knoten v_1 im Level 1 ist ein H-Knoten der den Teilzyklus `Medikation` darstellt. Die beiden Knoten v_2 und v_2' sind die Blätter der Auswahlebene σ^1 und stellen jeweils einen Teilzyklus `Pause` dar. Der Knoten v_2 entspricht einer `Pause` von 5 Tagen und v_2' einer `Pause` von 3 Tagen, d.h jeder zweite Teilzyklus `Pause` ist lediglich 3 Tage lang.

Semantisch stellt jede Auswahlebene im HSP-Baum selber ein individuelles Wiederholungsmuster dar, das die Reihenfolge der von einer Auswahl betroffenen Zyklen bzw. Zyklenpositionen repräsentiert. In Abbildung 5.5 etwa repräsentiert die Auswahlebene σ^s das Wiederholungsmuster der (theoretisch unendlichen) Auswahl "*Jede 2.Pause*" (siehe Abschnitt 4.2.2). Zur Bestimmung des individuellen Wiederholungsmusters wird jede Auswahlebene bzw. jeder Teilbaum T', gebildet aus Auswahlebene σ^s und ihren Blättern σ^s_{leaf}, als eigenständiger HSP-Baum verstanden. Für diesen Teilbaum T' wird (rekursiv) die entsprechende Funktion zur Ableitung von Wiederholungsmustern für HSP-Bäume aufgerufen. Im vorliegenden Fall einer einfachen Auswahlebene (ohne enthaltenen Auswahl- oder Projektionsebenen), d.h. einen (mit Ausnahme von Wurzelknoten) lediglich aus H-Knoten bestehenden eigenständigen HSP-Baum, ist die Bestimmung des Wiederholungsmusters direkt und einfach möglich (siehe Abschnitt 5.2.2). Für unser Beispiel ergibt sich das einfache Wiederholungsmuster (v_2, v_2') für die Auswahle-

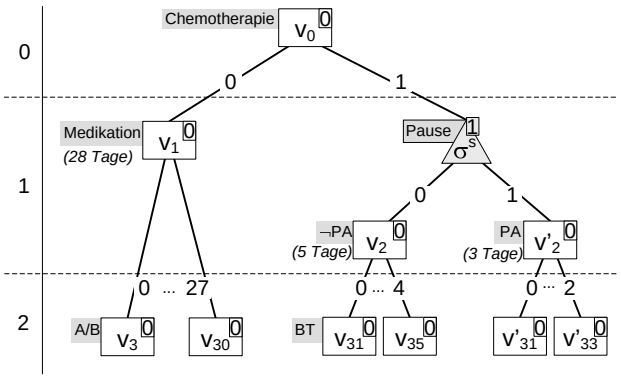
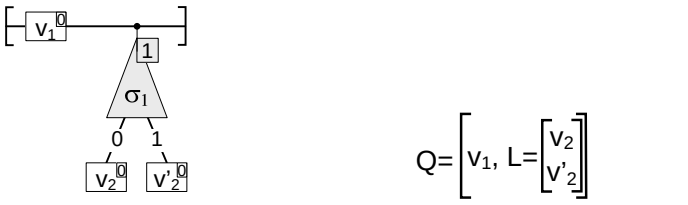


Abbildung 5.5: HSP-Baum mit einzelner einfacher Auswahlebene



(a) Vereinfachte Leveldarstellung (b) Kompakte Leveldarstellung

Abbildung 5.6: Level mit einzelner einfacher Auswahlebene

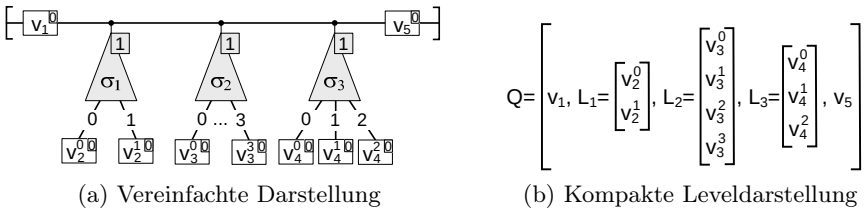
bene σ^s. Semantisch bedeutet dies, dass das erste Vorkommen einer `Pause` dem Knoten v_2 entspricht und das zweite Vorkommen dem Knoten v_2'.

Im Ergebnis führt der erste Schritt der Auswertung von Auswahlebenen zu einer *kompakten Leveldarstellung*. Das heißt, für jede Auswahlebene wird deren individuelles Wiederholungsmuster bestimmt und in Form einer Knotenliste L_i repräsentiert. Die (geordnete) Menge aller H-Knoten eines Levels, zusammen mit den individuellen Wiederholungsmustern der enthaltenen Auswahlebene, führt zu der kompakten Leveldarstellung. Die kompakte Leveldarstellung wird im Folgenden als Liste Q verstanden, die als Elemente einzelne Knoten oder Knotenlisten L_i enthält, wobei jedes L_i das Muster einer Auswahlebene darstellt. In Abbildung 5.6b ist die kompakte Leveldarstellung des Beispiels aus Abbildung 5.6a dargestellt. Das heißt, die kompakte Leveldarstellung des Level 1 entspricht der Liste Q, die den Knoten v_1 und die Knotenliste $L = [v_2, v_2']$ enthält, wobei L das Wiederholungsmuster der Auswahlebene σ_1 darstellt.[3]

5.2.3.2 Level mit mehreren Ebenen

Die Auswertung beim Vorkommen von mehreren Auswahl- und Projektionsebenen in einem Level unterscheidet sich nicht gegenüber dem zuvor vorgestellten Grundfall. Zur Auswertung wird für jede der Auswahlebenen des Levels jeweils das entsprechende individuelle Wiederholungsmuster gebildet. Im Ergebnis führt das zu einer kompakten Leveldarstellung, in der jedes Wiederholungsmuster einer Ebene durch jeweils eine Knotenliste L_i dargestellt ist. In Abbildung 5.7 ist ein entsprechendes Beispiel für einen Level mit mehreren Auswahl- und Projektionsebenen dargestellt. Analog zu Abbildung 5.6a ist nur ein einzelner Level dargestellt. Das Level besteht hierbei aus den beiden H-Knoten v_1 und v_5 am Anfang bzw. Ende des Levels, sowie aus den drei Auswahlebenen σ^1,σ^2 und σ^3 mit ihren entsprechenden Blättern.

[3] Aus Gründen der Übersicht werden die Knotenlisten L_i der kompakten Leveldarstellung in den folgenden Abbildungen stets in Vektorschreibweise aufgeführt. Das oberste Element des Vektors entspricht dem ersten Element in der Liste.



(a) Vereinfachte Darstellung (b) Kompakte Leveldarstellung

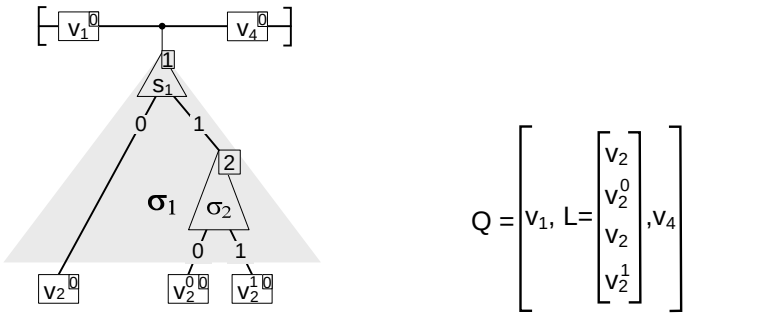
Abbildung 5.7: Level mit mehreren Auswahlebenen

5.2.3.3 Level mit geschachtelten Ebenen

Die Berücksichtigung von geschachtelten Auswahl- oder Projektionsebenen erfolgt automatisch aufgrund des rekursiven Aufrufs zur Bildung der individuellen Wiederholungsmuster. Während sich bei nicht-geschachtelten Ebenen das Wiederholungsmuster direkt ergibt, wird bei geschachtelten Ebenen der vollständige Algorithmus, wie in Abschnitt 5.2 dargestellt, durchlaufen. Die in einer geschachtelten Auswahlebene anzutreffenden Fälle entsprechen genau denen, die in dem vorliegenden Abschnitt besprochen werden. Abbildung 5.8 zeigt ein Beispiel für einen Level mit geschachtelter Auswahlebene. Wie Abbildung 5.8a zeigt, besteht der einzelne Level aus den beiden H-Knoten v_1 und v_4 sowie der geschachtelten Auswahlebene σ^1, in der die Auswahlebene σ^2 eingeschachtelt ist. Die kompakte Leveldarstellung zeigt Abbildung 5.8b. Die Knotenliste L entspricht dem individuellen Wiederholungsmuster der Ebene σ^1.

5.2.4 Schritt 2.2: Transformation

Im Schritt 2.2 bzw. bei der *Transformation* (vgl. Abbildung 5.2) finden Anpassungen der im vorherigen Schritt 2.1 ermittelten Wiederholungsmuster von Auswahl- oder Projektionsebenen statt. Solche Anpassungen werden nötig, da die ermittelten Muster der



(a) Vereinfachte Darstellung (b) Kompakte Leveldarstellung

Abbildung 5.8: Level mit geschachtelter Auswahlebene

verschiedenen Ebenen nicht in jedem Fall korrekt sind. Insbesondere ist dies bei Projektionsebenen der Fall.

5.2.4.1 Transformation der Muster von Projektionsebenen

Die Problematik bei Projektionsebenen ist, dass im HSP-Baum durch eine Projektionsebene lediglich das zur Projektion verwendete Periodizitätsschema repräsentiert wird. Das Ergebnis der Projektion, d.h. das Ergebnis einer Projektion zwischen zwei Periodizitätsschemas, wird jedoch nicht direkt im HSP-Baum abgebildet (siehe Abschnitt 4.2.3). Infolgedessen entspricht das im vorherigen Schritt 2.1 abgeleitete individuelle Wiederholungsmuster einer Projektionsebene lediglich dem Wiederholungsmuster des zu projizierenden Schemas, nicht jedoch dem Wiederholungsmuster der Projektion. Die Ermittlung des korrekten Wiederholungsmusters des Ergebnisses einer Projektion findet in dem Schritt der *Transformation* statt. Zu diesem Zweck werden, ausgehend von der kompakten Leveldarstellung, die durch den vorherigen Schritt ermittelten individuellen Wiederholungsmuster von Projektionsebenen "umgerechnet" bzw. transformiert. Das Ergebnis einer solchen Transformation ist eine kompakte Leveldarstellung, in der für jede Projektionsebene das korrekte vollständige Wiederholungsmuster enthalten ist. Abbildung 5.9 zeigt beispielhaft eine kompakte Leveldarstellung vor und nach einer solchen Transformation (siehe Abschnitt 5.4.1).

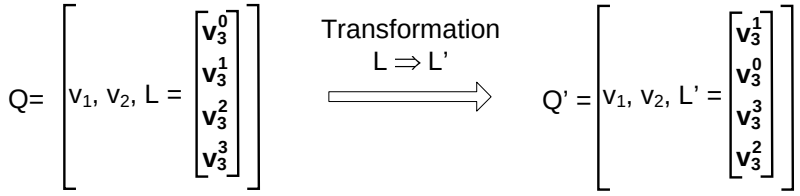


Abbildung 5.9: Transformation der Muster einer Projektionsebene

Abbildung 5.9 zeigt auch, dass die Transformation lediglich zu einem "Umsortieren" bzw. Permutieren der Knoten innerhalb des Musters L führt. Für die genaue Semantik dieser Transformation sind jedoch tiefer gehende theoretische Betrachtungen und Hintergründe notwendig, die in den Abschnitten 5.3 – 5.5 erfolgen. Hierbei wird sich unter anderem zeigen, dass die Transformation auch für eine Auswertung bestimmter Auswahlebenen übertragen und eingesetzt werden kann (siehe Abschnitt 5.5).

5.2.4.2 Berücksichtigung von Levelabhängigkeiten

Neben Projektions- und Auswahlebenen werden im Schritt 2.2. zusätzlich auch Abhängigkeiten zwischen den einzelnen Levels bei der levelbasierten Traversierung begegnet. Wie wir noch zeigen werden, können beim Entpacken eines Levels (siehe Schritt 2.3)

Abhängigkeiten entstehen, die in Form eines *Übertrags* in "tieferen" Levels "verrechnet" werden müssen. Interessanterweise basiert die Bestimmung und Verrechnung des Übertrags auf der gleichen Theorie und den gleichen Methoden, die auch zur Transformation bzw. Ermittlung der korrekten Wiederholungsmuster von Projektions- und Auswahlebenen angewendet werden. Die genauen Zusammenhänge werden im Detail in Abschnitt 5.6 erläutert.

5.2.5 Schritt 2.3: Entpacken der kompakten Leveldarstellung

Der letzte Schritt der Auswertung von Auswahl- und Projektionsebenen stellt das "Entpacken" der kompakten Leveldarstellung dar. Die hierbei enthaltenen Knotenlisten bzw. Wiederholungsmuster werden dabei linearisiert. Das heißt, im Ergebnis führt das Entpacken zu einer einfachen Knotenliste bzw. dem entsprechenden "einfachen" Wiederholungsmuster des Levels (siehe Abschnitt 5.1).

Abbildung 5.10 zeigt das Vorgehen des Entpackens anhand eines einfachen Beispiels. Die verwendete kompakte Leveldarstellung entspricht dem Beispiel aus Abschnitt 5.2.3, d.h. einer Chemotherapie mit der Auswahl "*Jede 2.Pause beginnend bei der Letzten*" für den Teilzyklus `Pause`. Die Liste $L = [v_2, v_2']$ repräsentiert das Wiederholungsmuster der (theoretisch unendlichen) Auswahl "*Jede 2.Pause*" bzw. der entsprechenden Auswahlebene. Das erste Vorkommen einer `Pause` entspricht dem Knoten v_2 und das zweite Vorkommen dem Knoten v_2'. Beim Entpacken wird der kompakte Level Q mehrfach durchlaufen und bei jedem Vorkommen von L der mit dem Vorkommen korrespondierende Knoten aus der Liste ausgewählt und in die Ergebnisliste übertragen. Die grauen Pfeile stehen jeweils für einen Durchlauf beim Entpacken des kompakten Levels. Die Ergebnisse des Entpackens bildet das einfache Wiederholungsmuster $[v_1, v_2, v_1, v_2']$, d.h. der Knoten v_1 steht für einen Medikationszyklus und die Knoten v_2 und v_2' für die Pausenzyklen. Der Knoten v_2' stellt zusätzlich den von der Auswahl betroffenen Pausenzyklus dar.

Vereinfacht ausgedrückt, wird beim Entpacken die kompakte Leveldarstellung so oft durchlaufen, bis alle Knoten der Liste L bzw. alle Knoten einer jeden Liste L_i mindestens einmal "besucht" wurden. Die hierfür notwendige Berechnung der erforderlichen Durchläufe des kompakten Levels damit alle Knoten mindestens einmal "besucht" wurden, hängt von der Anzahl der Knotenlisten in einem kompakten Level ab. Im einfachs-

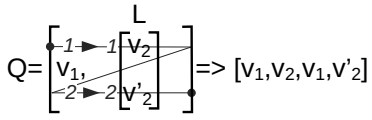


Abbildung 5.10: Entpacken der kompakten Leveldarstellung mit einzelnem Wiederholungsmuster

ten Fall, d.h. einer kompakten Leveldarstellung mit einer einzelnen Knotenliste L, gibt die Länge von L die Anzahl an Durchläufen der kompakten Leveldarstellung vor. Im obigen Beispiel muss folglich der kompakte Level genau zweimal durchlaufen werden, damit alle Knoten der Liste L mindestens einmal "besucht" wurden.

Den allgemeinen Fall stellen kompakte Leveldarstellung mit mehreren Knotenlisten dar. Hier müssen die Länge der einzelnen Knotenlisten in Beziehung gesetzt werden, um die Mindestanzahl an Durchläufen eines Levels zu ermitteln. Hierzu wird das kleinste gemeinsame Vielfache `kgV` über alle Längen der enthaltenen Knotenlisten L_i gebildet, d.h. $L_{min} := \texttt{kgV}(L_0, L_1, ..., L_n)$. Das Ergebnis L_{min} gibt die Mindestanzahl an, wie häufig der kompakte Level durchlaufen werden muss, damit alle Knoten innerhalb der Knotenlisten L_i mindestens einmal besucht wurden.[4] Im vorliegenden Beispiel etwa ergibt sich die Mindestanzahl von 12 Durchläufen ($\texttt{kgV}(3, \texttt{kgV}(2, 4) = 4) = 12$. In Abbildung 5.11 ist das Entpacken der kompakten Leveldarstellung aus Abbildung 5.7b dargestellt. Das Entpacken ist nach 12 Durchläufen von Q abgeschlossen und führt zu einem Wiederholungsmuster der Länge $60 = (12 \text{ Durchläufe} * 5 \text{ Elemente des Levels})$. Gut am Beispiel zu erkennen ist, dass beim Entpacken die Knotenlisten L_i als zirkulär betrachtet werden. Das heißt, nach dem Zugriff auf das letzte Element bzw. Knoten der Liste L_i führt ein erneuter (nachfolgender) Zugriff auf die Liste wieder zurück auf das erste Element. Im vorliegenden Beispiel ist dies beispielsweise bei der Liste L_1 im Durchlauf 3–4 der Fall.

Bei dem Entpacken der kompakten Leveldarstellung müssen enthaltene 0-Knoten in den Knotenlisten gesondert berücksichtigt werden.

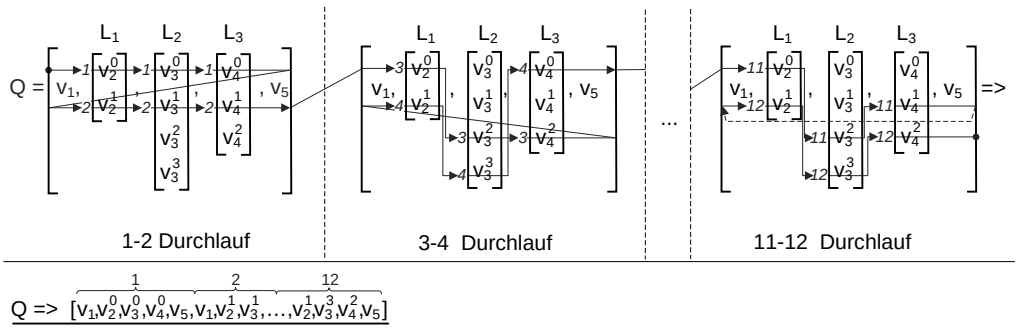


Abbildung 5.11: Entpacken der kompakten Leveldarstellung mit mehreren Wiederholungsmustern

[4] Eine einfachere Berechnung ist, die Anzahl der Durchläufe aus dem Produkt der einzelnen Längen der Knotenlisten L_i zu ermitteln.

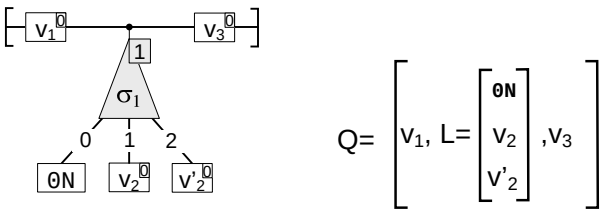
Allerdings kann diese Art der Berechnung u.U. zu zu großen Ergebnissen führen. Dies ist insbesondere dann der Fall, wenn zwei Knotenlisten nicht teilerfremd sind oder die gleichen Längen aufweisen. Dieses Vorgehen führt im Ergebnis zu Wiederholungsmustern, die zwar korrekt sind, aber weitaus länger als notwendig, weswegen in dieser Arbeit die `kgV`()-Berechnung verwendet wird.

5.2.5.1 Entpacken der Leveldarstellung mit enthaltenen 0-Knoten

Knoten vom Typ `ONODE` bzw. 0-Knoten bilden das Null-Element des HSP-Baums. Ein solches Nullelement wird benötigt, um das (periodische) Löschen von Zyklen aus einer Auswahlmenge abzubilden (siehe Abschnitt 4.2.4). Das heißt ein 0-Knoten stellt eine Art "Platzhalter" für (periodisch) gelöschte Zyklen oder Zyklenpositionen dar. Allerdings können Null-Knoten nicht "frei" im HSP-Baum vorkommen, sondern nur unterhalb von Auswahl- oder Projektionsebenen (siehe Abschnitt 4.2.4, Strukturregel SR-7). Aufgrund dessen erfolgt ihre Berücksichtigung immer im Zusammenhang mit der Auswertung ihrer entsprechenden Ebene. In Abbildung 5.12a ist ein entsprechendes Beispiel für ein Level mit einzelner Auswahlebene und einem Null-Knoten (`ON`) dargestellt, in dem jeder 3. Zyklus bzw. Zyklusposition (beginnend beim Ersten) entfällt.

Wie im allgemeinen Fall muss der kompakte Level so oft durchlaufen werden, bis alle Knoten innerhalb der Knotenliste L mindestens einmal besucht worden sind. Hierzu zählt auch der 0-Knoten. Dies ist wichtig, denn ein 0-Knoten stellt ein Zyklus der Länge=0 dar, der im Wiederholungsmuster zwar mit keiner Musterposition korreliert, aber dennoch "gezählt" werden muss, um eine korrekte Auswertung zu erhalten. Beispielsweise würde die Nichtberücksichtigung des 0-Knotens bei der Ermittlung der Durchläufe dazu führen, dass lediglich zwei Durchläufe stattfinden würden und der letzte Knoten v_2' in der Liste L fälschlicherweise nicht "besucht" wird.

Die gesonderte Berücksichtigung der 0-Knoten findet erst bei der Übertragung der Knoten von der kompakten Leveldarstellung in die Ergebnisliste statt. 0-Knoten werden zwar "gezählt", kommen aber nicht als Knoten in der Ergebnisliste vor, was ihrer Semantik als Platzhalter für gelöschte Zyklen oder Zyklenpositionen entspricht. In Abbildung 5.13 ist das Entpacken im Fall von 0-Knoten dargestellt.



(a) Vereinfachte Leveldarstellung (b) Kompakte Leveldarstellung

Abbildung 5.12: Level mit einzelner Auswahlebene und 0-Knoten

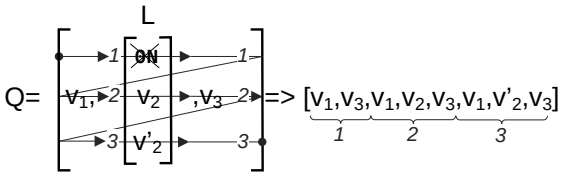


Abbildung 5.13: Entpacken der kompakten Leveldarstellung mit 0-Knoten

5.2.5.2 Entpacken von gekoppelten Auswahl- und Projektionsebenen

Eine gekoppelte Auswahl- bzw. Projektionsebene beschreibt die mehrmalige Anwendung von Auswahloperationen (siehe Abschnitte 4.2.2.2 und 4.2.3). Hierbei wird für ein bzw. mehrere Blätter der ersten Auswahlebene σ_1 eine weitere Auswahl vorgenommen. Abbildung 5.14 zeigt ein Level im HSP-Baum mit gekoppelter Auswahl. In diesem Beispiel wird für das erste Blatt der Auswahlebene σ_1 (symbolisiert mit s_1) eine weitere Auswahl vorgenommen, die zu der zweiten Auswahlebene σ_2 führt.

Die Auswertung gekoppelter Auswahl- oder Projektionsebenen in einem Level erfolgt auf Basis der in Abbildung 5.2 vorgestellten Schritte bzw. wird auf diese zurückgeführt. Genauer gesagt, wird mit der inneren Schleife des Algorithmus in Abbildung 5.2 die Auswertung von gekoppelten Ebenen berücksichtigt. Bei der Auswertung wird stets zunächst die erste Auswahlebene (σ^1) ausgewertet, im Anschluss dann die zweite (σ^2) bzw. nachfolgende gekoppelte Auswahlebenen. Die Auswertung der einzelnen Auswahlebene erfolgt analog wie zu Beginn des vorliegenden Abschnitts 5.2.5 besprochen. Das heißt, beim Entpacken der ersten Auswahlebene wird der Level solange durchlaufen bis alle Knoten mindestens einmal besucht wurden. Im Ergebnis führt das Entpacken zu einer einfachen Knotenliste des Levels. Allerdings enthält die Ergebnisliste nach Entpacken der kompakten Leveldarstellung der ersten Auswahlebene σ^1 erneut einen S-Knoten, nämlich den Wurzelknoten von Auswahlebene σ^2. An dieser Stelle wird die Ergebnisliste als "neuer" Level verstanden und eine entsprechende Auswertung dieses Levels und der enthaltenen Auswahlebenen durchgeführt. Dieses Vorgehen im Umgang mit gekoppelten Auswahlebenen wird solange fortgesetzt bis nur noch H-Knoten in der Ergebnisliste

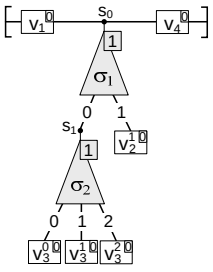


Abbildung 5.14: Level mit einer gekoppelten Auswahl

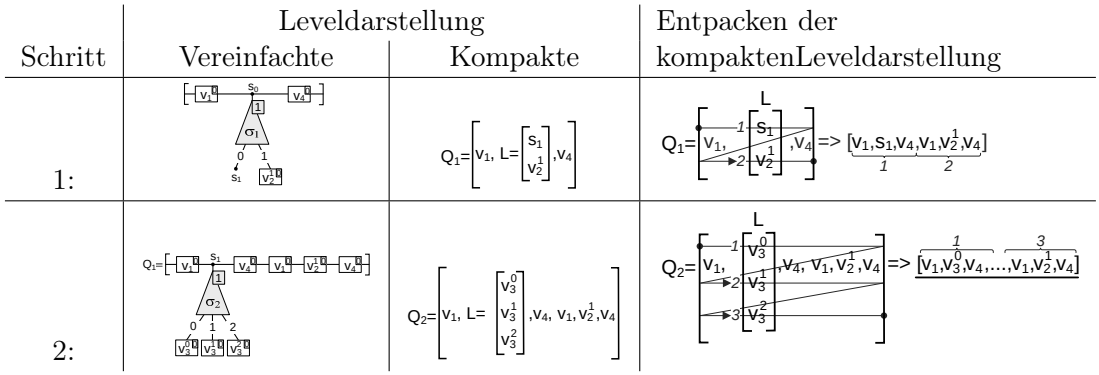


Abbildung 5.15: Auswertung eines Levels mit gekoppelter Auswahlebene

eines Levels enthalten sind.

Das vorgestellte Entpacken gekoppelter Ebenen erfolgt unabhängig von der konkreten Art der Ebenen, d.h. es findet das gleiche Vorgehen bei gekoppelten Auswahlebenen (σ_1/σ_2), gekoppelten Projektionsebenen (π_1/π_2) und bei Mischformen wie gekoppelte Auswahl- und Projektionsebene (σ/π) oder gekoppelte Projektions- und Auswahlebene (π/σ) statt. Abbildung 5.15 zeigt das Vorgehen für das Beispiel aus Abbildung 5.14. Im 1. Schritt wird zunächst lediglich die Auswahlebene σ_1 im Level ausgewertet. Im 2. Schritt wird das Ergebnis des 1.Schritts als "neues" Level verstanden, und hier dann die zweite (gekoppelte) Auswahlebene σ_2 ausgewertet.

In den vorausgehenden Abschnitten haben wir das allgemeine Vorgehen zur Ableitung von Wiederholungsmustern von HSP-Bäumen vorgestellt. Hierzu haben wir für jeden durchzuführenden Schritt dessen Semantik und Funktionsweise informell vorgestellt. Eine exakte algorithmische Beschreibung der einzelnen Schritte findet sich in Abschnitt 5.7. Abschließend zeigt Abbildung 5.16 die vollständige Traversierung bzw. Ableitung des Wiederholungsmusters des HSP-Baums aus Abbildung 5.5. In diesem Beispiel wird kein Transformationsschritt (Schritt 2.2.) benötigt. Die Knotenliste $[v_3, \ldots, v_{30}, v_{31}, \ldots, v_{35}, v_3, \ldots, v_{30}, v_{31}, v'_{32}, v'_{33}]$ auf Level 2 bildet das Ergebnis der Traversierung und entspricht damit dem Wiederholungsmuster des HSP-Baums.

Der schwierigste Schritt des Algorithmus zur Ableitung von Wiederholungsmustern bildet die Transformation von Mustern. Dazu werden im folgenden Abschnitt zunächst theoretische Grundlagen eingeführt, die anschließend auf die Auswertung des HSP-Baums übertragen werden.

5.3 Theoretische Grundlagen der Auswertung von Projektionsebenen

Im Folgenden werden die theoretischen Grundlagen bei der Auswertung von Projektionsebenen vorgestellt. Zunächst erfolgt eine Wiederholung bezüglich der Semantik von Projektionsebenen, in der das Grundproblem der Projektion im Detail dargestellt wird. Im Anschluss wird das Problem der Projektion auf klassische Wiederholungsmuster übertragen, um so die theoretischen Hintergründe der Auswertung aufzuzeigen.

5.3.1 Semantik von Projektionsebenen

Die Projektionsebenen dienen in HSP-Bäumen der Abbildung des Ergebnisses einer Projektionsauswahloperation (vgl. Abschnitt 4.2.3). So stellt eine Projektionsauswahl eine Auswahloperation dar, die die Verknüpfung zweier unabhängiger Periodizitätsschemas ermöglicht. Es werden nur die Zykluspositionen bzw. Labels eines Periodizitätsschemas PS_1 ausgewählt bzw. markiert, die zugleich im zweiten Periodizitätsschema PS_2 mit bestimmten Positionen bzw. Labels korrespondieren. So kann auf das Schema "*Jeden 3.Tag (wird ein Medikament gegeben)*" (PS_1) das Schema "*Jeden Montag und Mittwoch in einer Woche*" (PS_2) projiziert werden, um alle Gaben des Medikaments, die auf einen Montag bzw. Mittwoch fallen (Label `PA`), zu adressieren. Dieses Beispiel zeigt Abbildung 5.17. Die Projektionsebene (eingeleitet mit dem P-Knoten p) entspricht dem Schema PS_2.

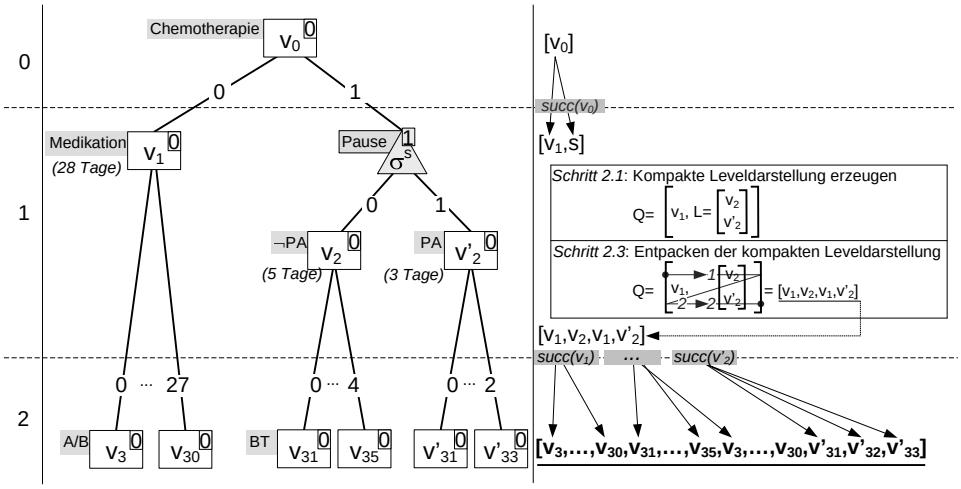


Abbildung 5.16: Traversierung eines HSP-Baums mit einfacher Auswahlebene

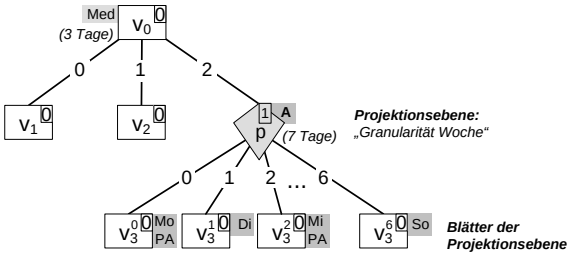


Abbildung 5.17: Beispiel für eine Projektionsebene im HSP-Baum

Im HSP-Baum wird lediglich die ursprüngliche Reihenfolge der Zyklen bzw. Zyklenpositionen des Periodizitätsschemas PS repräsentiert (siehe Abschnitt 4.2.3). So repräsentieren im obigen Beispiel die einzelnen Blätter der Projektionsebene lediglich das Vorkommen des Knotens v_3, das auf einen bestimmten Wochentag fällt. Beispielsweise entspricht der Knoten v_3^0 demjenigen Vorkommen von Knoten v_3, das auf einen Montag (Mo) fällt. Die Reihenfolge der Blätter der Projektionsebene entspricht jedoch nicht der korrekten Reihenfolge des Zusammenfallens. So fällt das erste Vorkommen von Knoten v_3 nicht, wie im HSP-Baum durch den Kantenindex 0 repräsentiert, auf einen Montag bzw. entspricht dem Knoten v_3^0. Stattdessen fällt das erste Vorkommen von v_3 auf einen Mittwoch, d.h. es entspricht dem Knoten v_3^2 (im Falle identischer Instanziierungspunkte beider Schemas). Die vollständige und korrekte Reihenfolge ist durch die folgende Knotenliste gegeben:

$$(v_3^2(Mi), v_3^5(Sa), v_3^1(Di), v_3^4(Fr), v_3^0(Mo), v_3^3(Do), v_3^6(So))$$

Im Kern ergibt sich das Problem, für jede Projektionsebene zu ermitteln, in welcher Reihenfolge welche Position von Schema PS mit den Positionen vom Schema PS_2, d.h. mit den Blättern dieser Projektionsebene, zusammenfallen. Diese Reihenfolge der Blätter bzw. die entsprechende Knotenliste bildet das korrekte Wiederholungsmuster der Projektionsebene, das es im Schritt der Transformation zu bestimmen gilt.

5.3.2 Theorie der periodischen Projektion

Das Prinzip der periodischen Projektion ist auf einfache Wiederholungsmuster, d.h. auf Muster mit äquidistanter Periodenlänge, leicht übertragbar. Im Folgenden gehen wir zunächst von jeweils identischen Instanziierungspunkten der Muster aus. Die periodische Projektion zwischen Wiederholungsmustern entspricht dem Problem, das Zusammenfallen von Musterpositionen zweier (unterschiedlich langer) Wiederholungsmuster M_1 und M_2 zu ermitteln. Genauer gesagt, gilt es zu ermitteln, bei welcher Wiederholung eine Position von Muster M_1 mit welcher Position in Muster M_2 zusammenfällt. Abbildung 5.18 zeigt die periodische Projektion zweier Wiederholungsmuster für das obige Beispiel "*Jeden 3.Tag (beginnend am Letzten wird ein Medikament gegeben)*" (PS_1) und "*Jeden Montag und Mittwoch in einer Woche*" (PS_2). Das Wiederholungsmuster

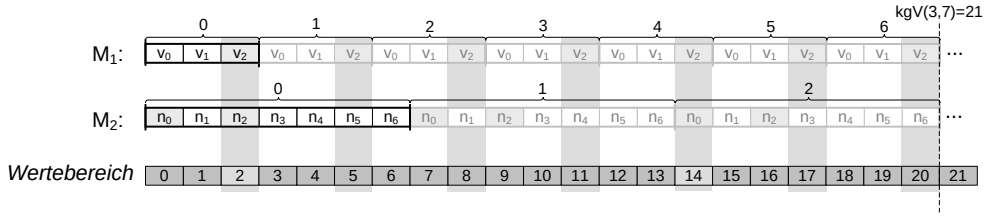


Abbildung 5.18: Problem der Projektion bei Wiederholungsmustern

für Schema PS_1 entspricht dem Muster M_1 bzw. der Knotenliste (v_0, v_1, v_2), und für Schema PS_2 den Muster M_2 bzw. der Knotenliste $(n_0, n_1, n_2, n_3, n_4, n_5, n_6)$. Die grau hinterlegten Bereiche in Abbildung 5.18 stellen das Zusammenfallen der Position v_2 von Muster M_1 mit den Positionen des Musters M_2 dar. Es ergibt sich die Reihenfolge $(n_2, n_5, n_1, n_4, n_0, n_3, n_6)$, d.h. in der ersten Wiederholung von M_1 fällt Position v_2 mit Position n_2 zusammen, in der zweiten Wiederholung mit Position n_5, etc.. Für das Beispiel ist insbesondere das Zusammenfallen von Position v_2 (Medikamentengabe in Schema PS_1) mit den Positionen n_0 und n_2 (Montag und Mittwoch in Schema PS_2) von besonderem Interesse und in der Abbildung durch die grau hervorgehobenen Werte im Wertebereich gekennzeichnet.

Abbildung 5.18 löst das Problem der periodischen Projektion bei Wiederholungsmuster durch Verwendung der kleinsten gemeinsamen Periodenlänge beider Muster. Das heißt, beide Muster werden lediglich innerhalb ihrer kleinsten gemeinsamen Periodenlänge verlängert und anschließend die jeweiligen Positionen entsprechend verglichen.[5] Allerdings kann sich die Verlängerung beider Muster innerhalb ihrer kleinsten gemeinsamen Periodenlänge als aufwändig erweisen. Es ergibt sich eine ähnliche Problematik wie bei der Realisierung von Vergleichsoperationen zwischen Mustern (siehe Abschnitt 2.3.3.1). Aufgrund des Aufwands drängt sich die Fragestellung auf, ob die periodische Projektion bzw. die Reihenfolge des Zusammenfallens von Positionen nicht direkt berechnet werden kann, d.h. ohne eine unmittelbare Verlängerung der Muster.

Um diese Frage zu klären, wird im Folgenden eine alternative Darstellungsweise des Problems der Projektion verwendet. Den Ausgangspunkt bildet eine Tabelle, die auf Basis von Muster M_1 erzeugt wird. In der Tabelle entspricht jede Spalte jeweils einer Musterposition von Muster M_1. Die Zeilen der Tabelle stehen jeweils für eine Wiederholung des Musters M_1. In die Tabelle werden nun, beginnend bei Spalte 0 und Zeile 0, die einzelnen Positionen von Muster M_2 bzw. das bezüglich des `kgV` verlängerten Musters M_2 eingefügt. Immer, wenn hierbei das (rechte) Ende der Tabelle erreicht ist, wird die Einfügung eine Zeile tiefer an der linken Seite der Tabelle fortgesetzt. Im Ergebnis enthält jede Spalte der so ausgefüllten Tabelle die Reihenfolge, in der eine Position v_i aus Muster M_1 mit den Positionen von Muster M_2 zusammenfällt. Für das Beispiel aus

[5] Die kleinste gemeinsame Periodenlänge beider Muster entspricht dem kleinsten gemeinsamen Vielfachen -`kgV`- zwischen den beiden Musterlängen (vgl. Abschnitt 2.3.3). Im Beispiel in Abbildung 5.18 ist die kleinste gemeinsame Periodenlänge = 21 ($kgv(3,7) = 21$).

Abbildung 5.18 ergibt sich folgende alternative Darstellungsweise bzw. Tabelle:

w	v_0	v_1	v_2
0	n_0	n_1	n_2
1	n_3	n_4	n_5
2	n_6	n_0	n_1
3	n_2	n_3	n_4
4	n_5	n_6	n_0
5	n_1	n_2	n_3
6	n_4	n_5	n_6

Die Spalte v_2 mit Positionen $(n_2, n_5, \ldots, n_6)$ repräsentiert die Reihenfolge, mit der die Position v_2 aus Muster M_1 mit den Positionen von Muster M_2 zusammenfällt.

Die alternative Darstellungsweise des Problems der Projektion kann dazu verwendet werden, um eine Funktion zu konstruieren, die in Abhängigkeit von einer Position v_i aus Muster M_1 und einer Wiederholungszahl w (bezüglich Muster M_1) die korrespondierende Position n_j im Muster M_2 ermittelt. Diese Funktion wird mit dem *Satz der periodischen Projektion* eingeführt. Im Beweis zu diesem Satz sind die Herleitung bzw. Konstruktion der Funktion dargestellt, die im Kern die alternative Darstellungsweise des Problems der Projektion verwendet.

Satz 5.1 (Periodische Projektion): *Gegeben seien zwei Muster $M_1 = (a_0, ..., a_{n-1})$ und $M_2 = (b_0, ..., b_{m-1})$ mit $m, n \in \mathbb{N}$. Dann kann zu jedem Element $a_i \in M_1$ sowie einer beliebigen Wiederholungszahl $w \in \mathbb{Z}$ (bezüglich Muster M_1) das korrespondierende Element $b_j \in M_2$ durch die Funktion $p(i, w) = (w * n + i) \bmod m = j$ bestimmt werden.*

Beweis: *Der Beweis für die Korrektheit der Funktion $p(i, w)$ erfolgt mit Hilfe ihrer Konstruktionsvorschrift:*

w	a_0	a_1	$\ldots$	a_{n-1}
0	$b_{0 \bmod m}$	$b_{1 \bmod m}$	$\cdots$	$b_{(n-1) \bmod m}$
1	$b_{(1*n)+0) \bmod m}$	$b_{(1*n)+1) \bmod m}$	$\cdots$	$b_{(1*n)+(n-1)) \bmod m}$
2	$b_{(2*n)+0) \bmod m}$	$b_{(2*n)+1) \bmod m}$	$\cdots$	$b_{(2*n)+(n-1)) \bmod m}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$m-1$	$b_{((m-1)*n+0) \bmod m}$	$b_{((m-1)*n+1) \bmod m}$	$\cdots$	$b_{((m-1)*n)+(n-1)) \bmod m}$

■

Der Beweis zu Satz 5.1 zeigt das Vorgehen bei der alternativen Darstellungsweise der Projektion formalisiert. In jeder Zelle wird stets mit *mod m* gerechnet, für den Fall, dass das (unverlängerte, ursprüngliche) Muster M_2 kürzer ist als Muster M_1 (was besonders für Spalte $n-1$ wichtig ist). Am Ende einer Zeile wird das nächste Element von Muster M_2 eine Zeile tiefer an der linken Seite eingefügt. Beispielsweise ist der Nachfolger von Element $b_{(n-1) \bmod m}$ (siehe Zeile 0) durch das Element $b_{n \bmod m}$ gegeben, bzw. anders ausgedrückt das Element $b_{((1*n)+0) \bmod m}$. Durch diese Vorgehensweise ergibt sich in

Zeile $m-1$ und Spalte a_{n-1} der allgemeine Fall, d.h. wir erhalten die im Satz eingeführte Funktion $p(i,w)$. Allerdings bildet die Herleitung der Funktion lediglich den ersten Schritt zur Lösung des Problems der Projektion. In weiteren Schritten gilt es zusätzlich, die Eigenschaften der Funktion und ihrer Anwendung bzw. ihrer Ergebnisse im Detail zu betrachten.

Eine Eigenschaft der alternativen Darstellungsweise ist es, dass jede Spalte der Tabelle das vollständige Wiederholungsmuster der Musterpositionen repräsentiert. Für das obige Beispiel bedeutet dies, dass in der Spalte v_2 bzw. dem Vektor $(n_2, n_5, n_1, n_4, n_0, n_3, n_6)$ alle Elemente von M_2 enthalten sind, die mit der Position v_2 des Musters M_1 korrespondieren. Diese Vollständigkeit ergibt sich trivialerweise dadurch, dass in der Tabelle im Wesentlichen das Muster M_2 erneut innerhalb des kgV der Muster M_1 und M_2 aufgeführt wird. Das kgV beider Muster ist in Zeile $(n-1)$ und Spalte $(m-1)$ erreicht. Die nachfolgende Zeile (m) und Spalte (0) kann dann wieder als Zeile 0 und Spalte 0 verstanden werden. Das heißt, die Tabelle bzw. entsprechende Spalten wiederholen sich ab da. Folgendes Korollar überträgt den Zusammenhang der Vollständigkeit auf die Funktion aus Satz 5.1. Das Korollar besagt, dass die Anwendung der Funktion $p(i,w) = j$ mit $0 \leq w \leq m-1$ zu einem Vektor führt, der dem vollständigen Wiederholungsmuster des Zusammenfallens der Musterposition a_i von Muster M_1 mit den Positionen b_j von Muster M_2 entspricht.

Korollar 5.1: *Gegeben seien zwei Muster $M_1 = (a_0, \ldots, a_{n-1})$ und $M_2 = (b_0, \ldots, b_{m-1})$ mit $m, n \in \mathbb{N}$. Vektor $v = (p(i,0), \ldots, p(i,m-1))$ enthält alle mit Element $a_i \in M_1$ korrespondierenden Elemente b_j mit $j \in v$ von Muster M_2.*

In diesem Zusammenhang gilt es, die Eigenschaften des Vektors v, d.h. die Menge korrespondierender Elemente einer Position aus Muster M_1, zu betrachten. Auffällig in den bisherigen Beispielen ist, dass stets jede Position von Muster M_1 genau einmal mit jeder Position von Muster M_2 zusammenfällt. Das mehrfache Vorkommen von Elementen von Muster M_2 war bisher nicht anzutreffen. Dies ist jedoch kein Zufall, sondern ein grundsätzliches Verhalten der periodischen Projektion, wie folgendes Korollar unterstreicht.

Korollar 5.2: *Gegeben seien zwei Muster $M_1 = (a_0, \ldots, a_{n-1})$ und $M_2 = (b_0, \ldots, b_{m-1})$ mit $m, n \in \mathbb{N}$. In dem Vektor $v = (p(i,0), \ldots, p(i,k-1))$ ist jedes Element von Muster M_2 maximal einmal enthalten, wobei für k gilt: $k = \frac{|M_2|}{ggT(m,n)}$.*

Beweis: *Siehe Anhang B.1* ∎

Das Korollar besagt, dass $v = (p(i,0), \ldots, p(i,k-1))$ stets jedes Element von Musters M_2 maximal einmal enthält. Das heißt, jede Zyklusposition von M_1 korrespondiert (maximal) einmal mit einer Zyklusposition von Muster M_2 (innerhalb der gemeinsamen Periodenlängen $kgV(m,n)$). Zusätzlich wird durch Korollar 5.2 die Länge des Vektors bzw. des Wiederholungsmusters der Positionen genau bestimmt. Allerdings müssen bezüglich Vektor v zwei Fälle unterschieden werden. Im ersten Fall sind die Muster M_1

und M_2 bzw. ihre Längen teilerfremd, d.h. der größte gemeinsame Teiler (ggT) beider Muster ist eins ($ggT(m,n) = 1$). Dann entspricht v stets dem vollständigen Musters M_2, d.h. jede Position von M_2 kommt genau einmal im Vektor v vor ($|v| = |M_2|$).

Im zweiten Fall sind die Längen beider Muster nicht teilerfremd ($ggT(m,n) \neq 1$), d.h. sie sind Vielfache voneinander. In diesem Fall enthält v nicht alle Elemente von M_2. Hierbei gilt der Zusammenhang $ggT(m,n) \neq 1 \Rightarrow |v| = \frac{|M_2|}{ggT(m,n)}$, d.h. das Wiederholungsmuster der Reihenfolgen wird bereits durch den kürzeren Vektor v vollständig beschrieben. So sind die beiden Periodizitätsangaben "*Jeden 2.Tag beginnend beim Ersten*" ($M_1 = (v_0, v_1)$) und "*Jeden 4.Tag beginnend beim Letzten*" ($M_2 = (n_0, n_1, n_2, n_3)$) bezüglich ihrer Länge nicht teilerfremd, d.h. $ggT(2,4) = 2$. Es ergibt sich die folgende alternative Darstellungsweise der Projektion:

w	v_0	v_1
0	n_0	n_1
1	n_2	n_3
2	n_0	n_1
3	n_2	n_3

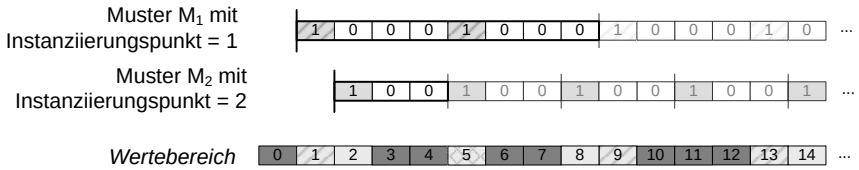
Wie an der obigen Tabelle zu erkennen ist, korrespondiert die Position v_1 lediglich mit Positionen n_1 und n_3 von Muster M_2, nicht jedoch mit Positionen n_0 und n_2. Somit können z.B. für Position v_1 die korrespondierenden Positionen in Muster M_2 bereits durch den kürzeren Vektor $v = (n_1, n_3)$ beschrieben werden (im Vergleich zur ursprünglichen Länge des Musters M_2). An dieser Stelle sollte jedoch erwähnt werden, dass die Funktion aus Satz 5.1 unabhängig von den beiden Fällen ($ggT(m,n) = 1$ vs. $ggT(m,n) \neq 1$) stets "richtig" rechnet, d.h. bezogen auf die Funktion müssen die beiden Fälle nicht unterschieden werden. Lediglich für das Ergebnis der Funktion ergeben sich die oben besprochenen Eigenschaften.[6]

Die in Satz 5.1 eingeführte Funktion sowie die in Korollar 5.1 und Korollar 5.2 beschriebenen Eigenschaften ihrer Anwendung bilden den theoretischen Hintergrund bzw. die Grundlagen der periodischen Projektion. Davon ausgehend wird im Folgenden die periodische Projektion bei Mustern mit jeweils unterschiedlichen Instanziierungspunkten vorgestellt.

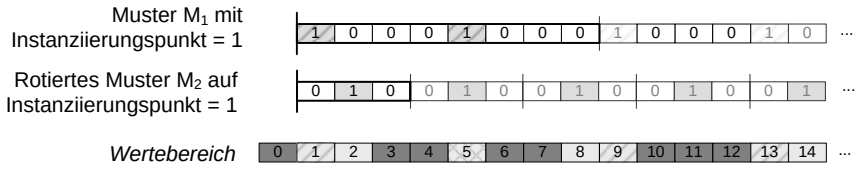
5.3.3 Umgang mit unterschiedlichen Instanziierungspunkten

Den grundsätzlichen Umgang mit unterschiedlichen Instanziierungspunkten bei Wiederholungsmustern haben wir bereits in Kapitel 2 vorgestellt. Bei Wiederholungsmustern

[6] In Kapitel 4 wurde in Abschnitt 4.3.1.3 der Begriff der Dead-Leafs bei HSP-Bäumen eingeführt (siehe Abschnitt 4.3.1.3). Diese Dead-Leafs hängen mit den in Korrolar 5.2 beschriebenen Eigenschaften der Projektion zusammen. Dead-Leafs enstehen genau dann, wenn die Länge des Wiederholungsmusters der Ebene und die Länge des Baums (in dem die Ebene enthalten ist) nicht teilerfremd sind. Die Dead-Leafs der Ebene stellen dann genau die Elemente im Muster der Ebene dar, die niemals mit einer bestimmten Position im Baumsmusters zusammenfallen.



(a) Unterschiedlicher Instanziierungspunkt



(b) Vereinheitlichter Instanziierungspunkt

Abbildung 5.19: Beispiel für Wiederholungsmuster mit unterschiedlichen Instanziierungspunkten und deren Vereinheitlichung

besteht die Möglichkeit, unterschiedliche Instanziierungszeitpunkte "herauszurechnen" bzw. zu vereinheitlichen. Ermöglicht wird dies durch eine Verschiebung auf einen gewünschten einheitlichen Instanziierungszeitpunkt bei gleichzeitiger zyklischer Rotation des Musters (jeder Wert der rechts "heraus fällt", wird links wieder eingefügt). Die Rotation ist werteäquivalent, d.h. die Wertemenge des rotierten Musters, ausgehend von dem vereinheitlichten Instanziierungspunkt, ist identisch zur ursprünglichen Wertemenge. Abbildung 5.19 zeigt das Prinzip der Rotation bei Mustern.

Diese Vorgehensweise zur Vereinheitlichung wird auch bei der Projektion von Mustern mit unterschiedlichen Instanziierungspunkten verwendet. Das heißt, Muster M_2 mit Instanziierungspunkt s_2 wird auf den Instanziierungspunkt s_1 von Muster M_1 vereinheitlicht bzw. beide Muster auf einen anderen beliebigen Punkt im Wertebereich vereinheitlicht. Hierzu wird im Folgenden zunächst die notwendige Rotation des Musters formal beschrieben. Zu diesem Zweck wird mit dem folgenden Satz eine entsprechende Funktion eingeführt, die für jede Position eines Musters, das korrespondierende Element in der rotierten Variante des Musters bestimmt. Als Rotationspunkt wird der Wert im Wertebereich bezeichnet, auf den das Muster vereinheitlicht wird.

Satz 5.2 (Rotation von Mustern): *Gegeben seien ein Muster* $M = (a_0, ..., a_{n-1})$ *mit* $n \in \mathbb{N}$*, sein Instanziierungspunkt* $s \in \mathbb{Z}$ *und ein beliebiger Rotationspunkt* $p \in \mathbb{Z}$*. Zu jedem Element* $a_i \in M_1$ *kann das bezüglich des Rotationspunktes* p *rotierte Element* $a_j \in M$ *durch die Funktion* $r(i, d) = (i + d) \bmod n = j$ *bestimmt werden, mit* $d = s - p$*.*

Beweis: *Der Beweis für die Korrektheit der Funktion* $r(i, d)$ *erfolgt mit Hilfe ihrer Konstruktionsvorschrift:*

Originalmuster	a_0	a_1	...	a_{n-1}
Rotiertes Muster	$a_{(0+d) \bmod n}$	$a_{(1+d) \bmod n}$	...	$a_{((n-1)+d)) \bmod n}$

■

Aus dem Satz 5.2 folgt das Korollar 5.3.

Korollar 5.3: *Gegeben sei ein Muster $M = (a_0, ..., a_{n-1})$ mit $n \in \mathbb{N}$, sein Instanziierungspunkt $s \in \mathbb{Z}$ und ein Rotationspunkt $p \in \mathbb{Z}$. Der Vektor $v = (r(0,d), ..., r(n-1,d))$ enthält alle rotierten Elemente a_j mit $j \in v$ von Muster M mit $d = s - p$.*

Das Korollar besagt, dass die Anwendung der Funktion $r(i,d) = j$ mit $0 \leq i \leq n-1$ zu einem Vektor führt, der dem vollständig rotierten Wiederholungsmuster von Muster $M = (a_0, ..., a_{n-1})$ entspricht. Angewandt auf das Beispiel aus Abbildung 5.19, d.h für das Muster $M_2 = (n_0, n_1, n_2, n_3)$ mit Instanziierungspunkt $s_2 = 3$, ergibt sich bei Vereinheitlichung von M_2 auf den Rotations- bzw. Instanziierungspunkt $s_1 = 1$ von M_1 das rotierte Muster (mit $d = s_1 - s_2 = -2$):

$$M_2^r = (n_{r(0,-2)=\mathbf{2}}, n_{r(1,-2)=\mathbf{3}}, n_{r(2,-2)=\mathbf{0}}, n_{r(3,-2)=\mathbf{1}})$$

Bei der Projektion von Mustern mit unterschiedlichen Instanziierungspunkten wird diese Funktion zur Rotation von Mustern zusätzlich verwendet. Ihre Anwendung lässt sich am besten anhand der alternativen Darstellung erklären. Ausgangspunkt bilden Muster $M_1 = (a_0, ..., a_{n-1})$ (mit Instanziierungspunkt s_1) und Muster $M_2 = (b_0, ..., b_{m-1})$ (mit Instanziierungspunkt s_2) sowie die Differenz $d = s_1 - s_2$. Ohne Beschränkung der Allgemeinheit wird im Folgenden zunächst stets die Projektion und erst anschließend die Rotation durchgeführt. Dies ist grundsätzlich möglich und führt zum gleichen Ergebnis.[7]

Im ersten Schritt wird zunächst der Instanziierungspunkt vernachlässigt und die Projektion mit Hilfe der entsprechenden Funktion $p(i,w)$ vorgenommen. Für den allgemeinen Fall kann dies durch die folgende Tabelle dargestellt werden, in der anstelle der Positionen von Muster M_2 lediglich die entsprechenden Funktionsaufrufe enthalten sind.

w	a_0	...	a_{n-1}
0	$b_{p(0,0)}$	...	$b_{p(n-1,0)}$
1	$b_{p(0,1)}$	...	$b_{p(n-1,1)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$m-1$	$b_{p(0,m-1)}$	...	$b_{p(n-1,m-1)}$

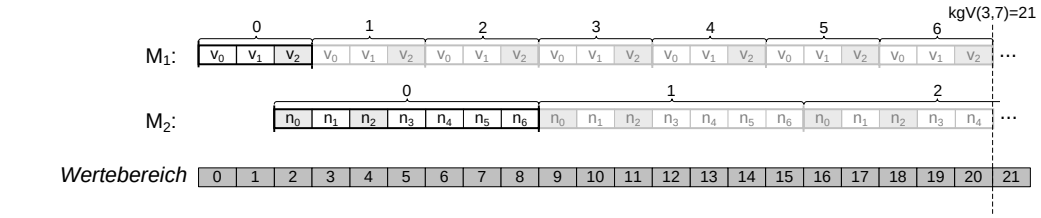
[7] Grundsätzlich spielt es keine Rolle, ob erst eine Rotation vorgenommen und anschließend das rotierte Muster verlängert wird (was bei der Projektion durch die alternative Darstellungsweise letztlich, wenn auch indirekt, gemacht wird) oder erst das Muster verlängert und anschließend das verlängerte Muster rotiert wird. An dieser Stelle ist die Wahl der Reihenfolge lediglich durch eine daraus resultierende verständlichere Darstellung der Aufrufe der eingeführten Funktionen begründet.

Im zweiten Schritt wird nun nachträglich für jede Zelle bzw. Musterposition eine entsprechende Rotation vorgenommen bzw. jeweils die Funktion $r(i, d)$ aufgerufen. Übertragen auf obige Funktionsaufrufe kann der allgemeine Fall der Projektion von Mustern mit unterschiedlichen Instanziierungspunkten durch folgende Tabelle bzw. alternative Darstellung beschrieben werden:

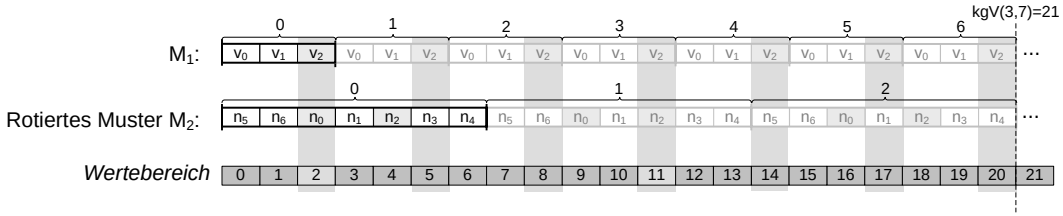
w	a_0	$\dots$	a_{n-1}
0	$b_{r(p(0,0),d)}$	$\dots$	$b_{r(p(n-1,0),d)}$
1	$b_{r(p(0,1),d)}$	$\dots$	$b_{r(p(n-1,1),d)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$m-1$	$b_{r(p(0,m-1),d)}$	$\dots$	$b_{r(p(n-1,m-1),d)}$

Über den "inneren" Aufruf der Funktion $p(i, w)$ wird das bezüglich eines Elementes $a_i \in M_1$ und einer Wiederholungszahl w korrespondierende Elemente b_j ermittelt. Nachträglich wird über den "äußeren" Aufruf $r(j, d)$ mit Differenz d das korrespondierende Element von b_j im rotierten Muster bestimmt. Das Wiederholungsmuster einer Projektion zwischen Mustern M_1 und M_2 für Knoten $a_i \in M_1$ (mit unterschiedlichen Instanziierungspunkten) wird durch den Vektor $v = (b_{r(p(i,0),d)}, ..., b_{r(p(i,m-1),d)})$ repräsentiert. Das heißt, Vektor v stellt die Reihenfolge des Zusammenfallens der Position a_i von Muster M_1 mit den Positionen von Muster M_2 dar. Abschließend ist in Abbildung 5.20 das eingangs verwendete Beispiel (siehe Abbildung 5.18) mit unterschiedlichen Instanziierungspunkten ($M_1 = 0$, $M_2 = 2$) dargestellt. Als Ergebnis für die periodische Projektion auf Muster M_1 bzw. Knoten v_2 mit Muster $M_2 = (n_0, ..., n_6)$ ergibt sich der folgende Vektor $v = (n_{r(p(2,0),d)}, ..., n_{r(p(2,6),d))}) = (n_0, n_3, n_6, n_2, n_5, n_1, n_4)$.

Mit dieser Beschreibung, wie mit unterschiedlichen Instanziierungspunkten bei Mustern umgegangen wird, sind unsere Betrachtungen zu den theoretischen Grundlagen und Hintergründen der periodischen Projektion abgeschlossen. Im folgenden Abschnitt werden diese theoretischen Grundlagen auf die Auswertung von HSP-Bäumen bzw. auf den Schritt der Transformation von Projektionsebenen übertragen.



(a) Muster vor der Rotation



(b) Projektion nach der Rotation

Abbildung 5.20: Projektion mit unterschiedlichen Instanziierungspunkten

5.4 Transformation: Auswertung von Projektionsebenen

Die in Abschnitt 5.3 vorgestellte Theorie der periodischen Projektion ermöglicht es, das Ergebnis einer Projektionsauswahloperation unmittelbar auf Grundlage weniger Parameter zu berechnen. Zur Berechnung werden lediglich die Längen der beiden individuellen Wiederholungsmuster von Schema P_1 und P_2, die Position des zu projizierenden Schemas P_2 innerhalb des Wiederholungsmusters von Schema P_1 sowie die beiden Instanziierungspunkte der Schemas benötigt. Für den Algorithmus zur Ableitung von Wiederholungsmustern bedeutet dies, dass lediglich diese Grundwerte bei der Transformation (Schritt 2.2 des Algorithmus) zur Verfügung gestellt werden müssen, um das korrekte Wiederholungsmuster einer Projektionsebene im HSP-Baum zu berechnen bzw. zu transformieren. Das heißt, für die im Schritt 2.1 ermittelten individuellen Wiederholungsmuster einer Projektionsebene können mit Hilfe der Projektion und Rotation die korrekten Wiederholungsmuster abgeleitet werden (siehe Abschnitt 5.2.4). Hierzu werden zwei Funktionen, die Projektionsfunktion $\mathcal{P}$ und die Rotationsfunktion $\mathcal{R}$, eingeführt, die die im Schritt 2.1 ermittelten Wiederholungsmuster bzw. Knotenlisten L einer Projektionsebene entsprechend transformieren. Auf Basis der Funktionen wird das individuelle Wiederholungsmuster der Projektionsebene bzw. die Knotenliste L in das entsprechend korrekte Wiederholungsmuster L' transformiert. Die Parameter beider Funktionen sowie deren (informelle) Semantik zeigt Tabelle 5.1.[8] Die Projektionsfunktion $\mathcal{P}$ agiert positionsabhängig, d.h. sie ermittelt das korrekte Ergebnismuster L'

[8] Die genaue Funktionsweise bzw. Definition beider Funktionen zeigt Algorithmus A.6 und Algorithmus A.7 in Anhang A.

Funktion	Beschreibung
$\mathcal{P}(L, n, m, pos) = L'$	L: Individuelle Wiederholungsmuster des Schemas P_2 (als Knotenliste)
	n: Länge des Wiederholungsmusters von Schema P_1
	m: Länge des Wiederholungsmusters von Schema P_2
	pos: Menge von Musterposition der Projektion im Wiederholungsmuster von P_1
	L': Korrekte Wiederholungsmuster von Schema P_2 (als Knotenliste)
$\mathcal{R}(L, r, z) = L'$	L: Wiederholungsmuster von Schema P (als Knotenliste)
	r: Rotationspunkt $r \in \mathbb{Z}$
	z: Instanziierungspunkt von Schema P $z \in \mathbb{Z}$
	L': Rotierte Wiederholungsmuster von Schema P (als Knotenliste)

Tabelle 5.1: Übersicht der verwendeten Basisfunktionen im HSP-Baum

einer Projektion in Abhängigkeit von der übergebene Menge an Positionen *pos* und den Längen m, n der beiden Wiederholungsmuster der Schemas P_1 und P_2. Die Rotationsfunktion $\mathcal{R}$ ermittelt das Muster bzw. die Knotenliste L', die sich bei Rotation des Wiederholungsmusters bzw. der Liste L mit Instanziierungspunkt z bezüglich des (übergebenen) Rotationspunkt r (z.B. Instanziierungspunkt eines zweiten Musters) ergibt.

Bei Anwendung der Funktionen und ihrer Parameter gilt es, die Besonderheiten des HSP-Baums zu berücksichtigen. Genauer gesagt, dürfen die benötigten Funktionsparameter nicht zum Zeitpunkt der Auswertung ermittelt werden, sondern werden durch die Projektionsparameter einer Ebene zur Verfügung gestellt. Wie in Abschnitt 4.2.5 gezeigt, ist im HSP-Baum die Situation zum Zeitpunkt der Auswahloperation wesentlich und nicht die Situation zum Zeitpunkt der Auswertung. Das bedeutet zum Beispiel, dass zum Zeitpunkt der Auswertung nicht die vorliegende Musterlänge von Schema P_1 verwendet werden darf, sondern die Länge zum Zeitpunkt der Projektionsoperation zugrundezulegen ist. Die Länge des Musters von Schema P_1 kann zu beiden Zeitpunkten identisch sein, muss aber nicht. Wie in Abschnitt 4.2.5 vorgestellt, erfolgt die "Sicherung" der Situation zum Operationszeitpunkt durch die Auswahl- bzw. Projektionsparameter, die jeder Ebene zugewiesen werden.

Durch Einführung der Theorie der periodischen Projektion ergibt sich unmittelbar die Semantik der Projektionsparameter $SP^n = (z, l_P, l_A, Pos)$, $NT^n =$ `PNODE`. Tabelle 5.2 zeigt diese in einer Übersicht. Der Parameter z repräsentiert den Instanziierungspunkt des zu projizierenden Schemas P_2. Der zweite benötigte Instanziierungspunkt von Schema P_1 wird zum Zeitpunkt der Auswertung ermittelt bzw. festgelegt. Die Parameter l_P und l_A repräsentieren die Länge der Wiederholungsmuster der Schemas P_1 und P_2

zum Zeitpunkt der Projektionsoperation. Der Parameter Pos repräsentiert die Menge an Positionen des Musters von Schema P_2 innerhalb des Musters von P_1. Hierbei gibt die Länge des Musters von P_1 das Gültigkeitsintervall der Positionen vor, d.h. es gilt $Pos := \{x \,|\, 0 \leq x < l_P\}$.

Bei Anwendung der Parameter für die Projektions- und Rotationsoperation gilt es zwei Fälle zu unterscheiden: einelementige Positionsmengen ($|Pos| = 1$) und mehrelementige Positionsmengen ($|Pos| > 1$).

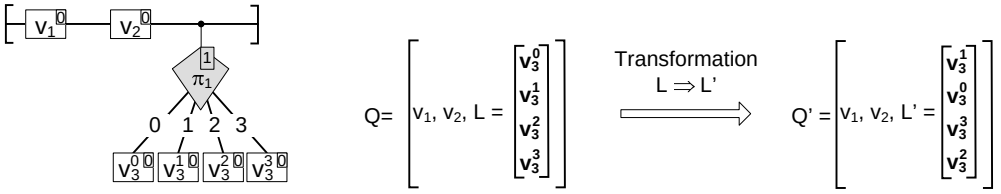
5.4.1 Einelementige Positionsmenge

Eine einelementige Positionsmenge stellt den einfachsten Fall der Verwendung der Projektionsparameter für die Projektions- und Rotationsfunktion dar. In diesem Fall können das Wiederholungsmuster und die Projektionsparameter einer entsprechenden Ebene unmittelbar auf die Funktionen übertragen werden. Das heißt, ausgehend von dem individuellen Wiederholungsmuster $L = [v_0, ..., v_n]$ der Projektionsebene π^n, den Projektionsparametern $SP^n = (z, l_P, l_A, Pos = \{p\})$ und dem Instanziierungspunkt z_1 des dem HSP-Baum entsprechenden Periodizitätsschemas, ergibt sich das korrekte Wiederholungsmuster L' durch die Funktionsaufrufe: $\mathcal{P}(L, l_P, l_A, p) = L'_1$ und $\mathcal{R}(L'_1, z_1, z) = L'$.

Abbildung 5.21 zeigt den Schritt der Transformation anhand eines Beispiels. Ausgangspunkt bildet die vereinfachte Leveldarstellung des Beispiels aus Abbildung 5.21a. Die für diesen Level im Schritt 2.1 erzeugte kompakte Leveldarstellung (Q), mit den individuellen Wiederholungsmustern der Projektionsebene, ist in Abbildung 5.21b dargestellt. Für das individuelle Wiederholungsmuster L der Projektionsebene π^n, mit Projektionsparametern $SP^n = (0, 3, 4, \{2\})$ und Instanziierungspunkt $z_1 = 0$, werden die Projektionsfunktion $\mathcal{P}(L, 3, 4, 2) = L_1$ und die Rotationsfunktion $\mathcal{R}(L_1, 0, 1) = L'$ angewendet. Es resultiert die kompakte Leveldarstellung Q' auf der rechten Seite in Abbildung 5.21b mit dem korrekten Wiederholungsmuster L' der Projektionsebene.

Parameter	Semantik
z	Instanziierungspunkt des Projektionsschemas P_2
l_P	Länge des Wiederholungsmuster von P_1
l_A	Länge des Wiederholungsmuster von P_2
$Pos = \{p_1, ..., p_m\}$	Positionen von P_2 innerhalb des Musters von P_1

Tabelle 5.2: Semantische Zuordnung der Projektionsparameter



(a) Vereinfachte Leveldarstellung (b) Transformation der kompakten Leveldarstellung

Abbildung 5.21: Ableitung des korrekten Wiederholungsmusters einer Projektionsebene

5.4.2 Mehrelementige Positionsmengen

Mehrelementige Positionsmengen bei Projektionsparametern sind das Ergebnis vorhandener 0-Knoten, d.h. Knoten vom Typ `ONODE`, in einem Level zum Zeitpunkt der Anwendung der Projektionsoperation. Wie zuvor gezeigt, hängt die Projektion eines Musters M_2 auf das Muster M_1 von der Länge beider Muster und insbesondere von der Position der Projektion innerhalb des Musters M_1 ab. Allerdings kann aufgrund der Existenz von 0-Knoten diese Position teilweise nicht eindeutig angegeben werden. Stattdessen variiert sie, was durch die Menge an Positionsangaben für eine Projektionsebene dargestellt wird.

5.4.2.1 Ursache von mehrelementigen Mengen an Positionsangaben

Um den Effekt der mehrelementigen bzw. schwankenden Positionsangaben näher zu betrachten, ist in Abbildung 5.22 zunächst die Situation in einem Level ohne 0-Knoten zum Zeitpunkt der Projektionsoperation dargestellt. Auf der linken Seite ist der Level dargestellt, in der Mitte das Entpacken der kompakten Leveldarstellung und auf der rechten Seite das resultierende Wiederholungsmuster. Bei Anwendung einer Projektionsoperation für den Knoten v_3 (das konkrete Projektionsschema bzw. dessen Länge l_A spielen zunächst keine Rolle) zeigt sich, dass dieser Knoten im Ergebnismuster an den Positionen zwei und sechs vorkommt. Allerdings kann jeder Durchlauf beim Entpacken der kompakten Leveldarstellung als Wiederholung eines (äquidistanten) Musters der Länge drei verstanden werden. Je nach Wiederholung bzw. Durchlauf werden z.T. zwar andere Knoten verwendet (siehe Knoten v_2^0 und v_2^1 in Abbildung 5.22), jedoch bleibt die Situation pro Durchlauf gleich. Das heißt, jeder Durchlauf führt zu einem (Teil-) Muster der Länge drei, in dem der Knoten v_3 stets die Position drei (bzw. zwei bei Zählung von 0) aufweist - unabhängig davon, wieviele weitere Wiederholungen oder Durchläufe stattfinden. Beim Einfügen einer Projektionsebene als Ergebnis der Projektionsoperation auf Knoten v_3 ist es folglich ausreichend, für die Projektionsparameter

als Musterlänge des Periodizitätsschemas $l_p = 3$ und für die Position = 2 ($Pos = \{2\}$) anzugeben.

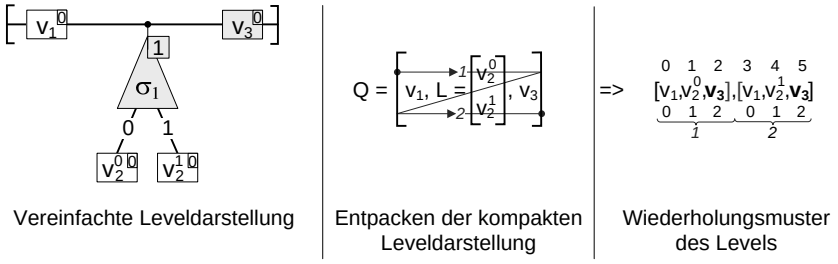


Abbildung 5.22: Vereinfachte Darstellung eines Levels ohne 0-Knoten zum Zeitpunkt der Projektionsoperation

Vereinfacht ausgedrückt, kann in einem Level ohne 0-Knoten jeder Durchlauf beim Entpacken als Wiederholung eines Musters verstanden werden, das die gleiche Länge wie der Level aufweist. Die Positionen aller Knoten und Ebenen bleiben bei jedem Durchlauf gleich. Das Ergebnismuster (rechte Seite) ist in seiner Länge somit stets ein ganzzahliges Vielfaches der Länge des Levels (hier z.B.: Länge = $6 \equiv 3{*}2$ Durchläufe).

Durch 0-Knoten in einem Level wird der direkte Zusammenhang zwischen Level (-länge) und Muster (-länge) jedoch aufgehoben. Abbildung 5.23 zeigt ein entsprechendes Level zum Zeitpunkt der Projektionsoperation mit 0-Knoten (`0N`). Auf der linken Seite ist der Level dargestellt. In der Mitte das Entpacken der kompakten Leveldarstellung und auf der rechten Seite das resultierende Wiederholungsmuster.

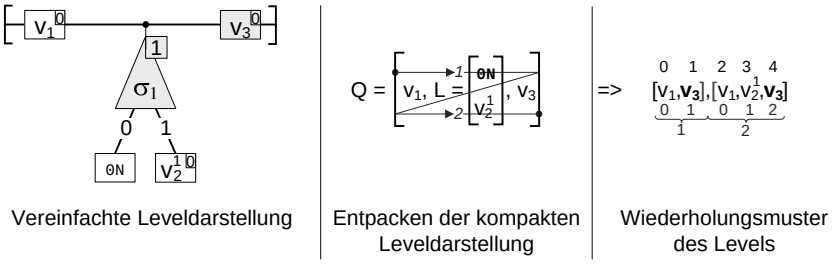


Abbildung 5.23: Vereinfachte Darstellung eines Levels mit 0-Knoten zum Zeitpunkt der Projektionsoperation

Im Vergleich zum vorherigen Beispiel zeigt sich, dass der Knoten v_3 an den Positionen eins und vier vorkommt. Zusätzlich ändert sich die Länge des Musters pro Durchlauf. Im ersten Durchlauf ist die Länge = 2 und im zweiten Durchlauf gilt Länge = 3. Dies führt dazu, dass ein Durchlauf im Level nicht als "Wiederholung" eines Musters

verstanden werden kann. Stattdessen muss bei der Projektion das vollständige Ergebnismuster berücksichtigt werden, und es ist nicht ausreichend (wie im vorherigen Fall), lediglich die pro Durchlauf erzeugten (Teil-) Muster zu betrachten. Das heißt, im vorliegenden Beispiel muss die Projektion für ein Muster der Länge = 5 ($l_P = 5$) stattfinden und nicht wie zuvor auf einem Muster der Länge = 3 ($l_P = 3$).

Allerdings wird auch im vorliegenden Fall nur eine einzelne Projektionsebene im Level als Ergebnis der Projektionsoperation auf Knoten v_3 angelegt. Es kann in den Projektionsparametern zwar die Länge des mit dem Level korrespondierenden Wiederholungsmusters durch den einzelnen Wert $l_p = 5$ beschrieben werden, die Position der Projektion hingegen ist nur über die mehrelementige Menge $Pos = \{1, 4\}$ beschreibbar. Erst durch vollständige Auflistung aller Positionen einer Ebene im Muster kann die korrekte Auswertung einer Projektionsebene bei solchen Levels gewährleistet werden. Zum Zeitpunkt der Auswertung müssen mehrelementige Mengen an Positionsangaben aber gesondert berücksichtigt werden.

5.4.2.2 Umgang mit mehrelementigen Mengen von Positionsangaben

Für den Umgang mit mehrelementigen Mengen von Positionsangaben gehen wir zunächst von dem einfachen Fall der einelementigen Menge an Positionsangaben aus. Ausgangspunkt bildet das in Abbildung 5.24 dargestellte Beispiel eines Levels zum Zeitpunkt der Auswertung. Es handelt sich hierbei um das gleiche Beispiel wie in Abbildung 5.22, allerdings mit eingefügter Projektion auf Knoten v_3 bzw. entsprechender Projektionsebene.

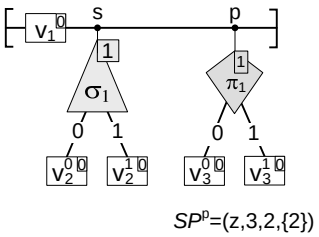


Abbildung 5.24: Vereinfachte Darstellung eines Levels ohne 0-Knoten zum Zeitpunkt der Auswertung

Im ersten Schritt betrachten wir zunächst die Situation bei der Projektion für alle Positionen des vollständigen Ergebnismusters M_1 mit Länge 6, wie in Abbildung 5.22 auf der rechten Seite dargestellt. Wir gehen an dieser Stelle bewusst nicht von der verkürzten Betrachtung des Musters aus, d.h. von einem Muster der Länge drei, das sich lediglich wiederholt (siehe oben). Des Weiteren gehen wir von dem Projektionsschema bzw. Muster $M_2 = [n_1, n_2]$ aus, dass letztlich durch die Projektionsebene π_1, wenn auch nur für eine Position im Level, repräsentiert wird. Es gilt zu beachten, dass die Auswahl- und

Projektionsebenen bzw. deren Muster gleich lang sind, wodurch das vollständige Wiederholungsmuster des Levels erneut die Länge sechs aufweist (siehe Abschnitt 5.2.5). Es ergibt sich folgende (vollständige) Darstellung der Projektion von Muster M_2 auf alle Positionen von M_1 (siehe Abschnitt 5.3.2):

w	v_1	v_2^0	v_3^0	v_1	v_2^1	v_3^1
0	n_1	n_2	$\mathbf{n_1}$	n_2	n_1	$\mathbf{n_2}$

$\equiv$

w	v_1	v_2^0/v_2^1	v_3^0/v_3^1
0	n_1	n_2	$\mathbf{n_1}$
1	n_2	n_1	$\mathbf{n_2}$

Auf der linken Seite sind die Projektion des Musters M_1 (oberhalb) und die mit seinen Positionen bzw. Knoten korrespondierenden Positionen bzw. Knoten von Muster M_2 dargestellt (unterhalb). Da M_1 und M_2 nicht teilerfremd sind, korreliert jeder Knoten von M_1 genau einmal mit einem Knoten von M_2 (Korollar 5.2). Auf der rechten Seite ist die Situation der Projektion mit dem kürzeren Muster der Länge drei dargestellt. Bei der Projektion auf Knoten v_3 (Projektionsebene π_1 in Abbildung 5.22) ergibt sich das Ergebnismuster $[n_1, n_2]$ bzw. das Muster $[v_3^0, v_3^1]$.[9] In diesem Fall ist das Ergebnismuster der Projektion identisch zum Projektionsmuster M_2 (siehe fett dargestellte Knoten). Wesentlich ist, dass das Ergebnismuster der Projektionsebene π auf Basis der Theorie der periodischen Projektion direkt berechnet werden kann. Das heißt, ausgehend vom individuellen Wiederholungsmuster der Projektionsebene $L = [v_3^0, v_3^1]$ und dem Funktionsaufruf $\mathcal{P}(L, z, 3, 2, 2)$ (mit Länge $M_1 = 3$, Länge $M_2 = 2$ und Position in $M_1 = 2$) ergibt sich das (gleiche) Ergebnismuster $L' = [v_3^0, v_3^1]$.

Die direkte Berechnung des Ergebnismusters einer Projektionsebene ist bei mehrelementigen Mengen von Positionsangaben nicht mehr ohne weiteres gegeben. Ausgangspunkt der folgenden Betrachtungen bildet das Beispiel aus Abbildung 5.25. Es handelt sich um dasselbe Beispiel wie in Abbildung 5.23, allerdings mit eingefügter Projektionsebene für Knoten v_3.

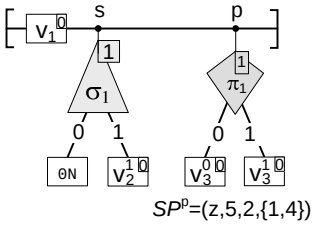


Abbildung 5.25: Vereinfachte Darstellung eines Levels mit 0-Knoten zum Zeitpunkt der Auswertung

[9] Wie in Abschnitt 4.2.3 dargestellt, werden die Blattknoten des Projektionsschemas durch Kopien des Knotens (der Knoten) ersetzt, der (die) im entsprechenden HSP-Baum von der Projektion betroffen ist (sind). Das führt dazu, dass der Knoten n_1 im HSP-Baum durch den Knoten v_3^0 und der Knoten n_2 durch den Knoten v_3^1 dargestellt wird.

Auch in diesem Fall betrachten wir zunächst die Situation einer Projektion mit Projektionsschema bzw. Muster $M_2 = [n_1, n_2]$ für alle Positionen des vollständigen Ergebnismusters M_1 mit der Länge 5. Es ergibt sich folgende Situation:

w	v_1	v_3^0	v_1	v_2^1	v_3^1
0	n_1	$\mathbf{n_2}$	n_1	n_2	$\mathbf{n_1}$
1	n_2	$\mathbf{n_1}$	n_2	n_1	$\mathbf{n_2}$

$\equiv$

w	v_1	v_2^1	v_3^0/v_3^1
0	n_1	$-$	$\mathbf{n_2}$
1	n_1	n_2	$\mathbf{n_1}$
2	n_2	$-$	$\mathbf{n_1}$
3	n_2	n_1	$\mathbf{n_2}$

Auf der linken Seite ist die Projektion für alle Positionen von Muster M_1 dargestellt. Es ist gut erkennbar, dass das resultierende Muster der Projektion für die beiden Vorkommen des Knotens v_3 verschieden ist. Das heißt, an Position 1 ergibt sich das Muster $[n_2, n_1]$ und an Position 4 das Muster $[n_1, n_2]$. Das vollständige Wiederholungsmuster der Projektion, aus Sicht von Knoten v_3 bzw. der Projektionsebene π_1, ist auf der rechten Seite dargestellt und entspricht den Mustern $[n_2, n_1, n_1, n_2]$ bzw. $[v_3^1, v_3^0, v_3^0, v_3^1]$. Wie leicht zu sehen ist, kann dieses Wiederholungsmuster der Projektionsebene nicht durch die Projektionsfunktion $\mathcal{P}$ berechnet werden bzw. es wird nicht durch die Theorie der periodischen Projektion abgedeckt.

Allerdings ist es möglich, das vollständige Wiederholungsmuster der Projektion zu konstruieren. Hilfreich ist hierbei die Darstellung der Projektion auf der linken Seite (siehe oben). Bei näherer Betrachtung zeigt sich, dass man das vollständige Muster der Projektionsebene π auf der rechten Seite erhält, wenn man die beiden einzelnen Muster der zugehörigen Positionen eins und vier auf der linken Seite, wie in Abbildung 5.26 gezeigt, "durchläuft".

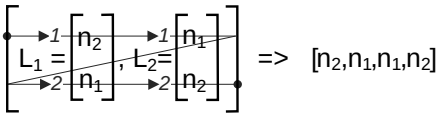


Abbildung 5.26: Beispiel für die Konstruktion des Ergebnismusters der Projektion

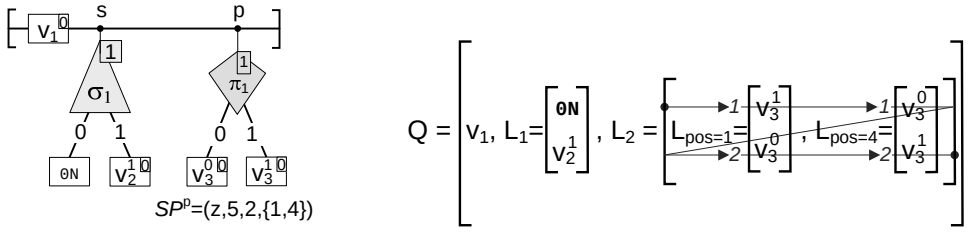
Die "kleinen" Muster pro Position werden lediglich entsprechend ihrer Reihenfolge bzw. die enthaltenen Knoten entsprechend ihres Vorkommens durchlaufen und in das "große" Muster übertragen. Diese Form des "Durchlaufens" ist jedoch aus Schritt 2.3 (siehe Abschnitt 5.2.5) bekannt, d.h. dem Entpacken der kompakten Leveldarstellung. Das Vorgehen beim Entpacken der kompakten Leveldarstellung ist analog dem Durchlaufen der einzelnen Muster zur Konstruktion des "großen" Musters der Projektion.

Der Zusammenhang zwischen Projektionsmuster pro Position und deren Durchlaufen zur Konstruktion des vollständigen Musters, ermöglicht den Umgang mit mehrelementigen Mengen von Positionsangaben. Genauer gesagt wird dieser Fall auf den Fall einelementiger Mengen und dem Entpacken der kompakten Leveldarstellung zurückgeführt.

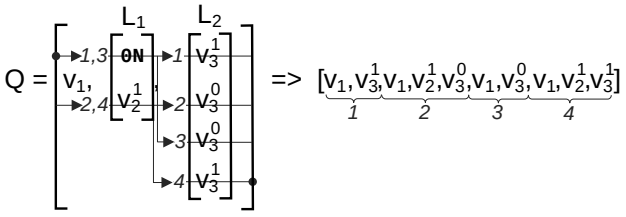
Ausgehend von den Projektionsparametern $SP^v = (z, l_P, l_A, \{p_0, ..., p_n\})$ der Projektionsebene π^v, dem individuellen Wiederholungsmuster $L = [v_0, ..., v_n]$ und den Instanziierungspunkt z_1 kann das Vorgehen bei mehrelementigen Mengen an Positionsangaben wie folgt angegeben werden ($Pos = \{p_0, ..., p_n\}$):

1. Für jede Position $p_i \in Pos$ wird das korrekte Wiederholungsmuster L_i der Projektionsebene für diese Position durch den Funktionsaufruf $\mathcal{R}(\mathcal{P}(L, l_P, l_A, p_i), z, z_1)$ gebildet.
2. Das Ergebnis des ersten Schritts bildet die kompakte Darstellung des Projektionsmusters $L = [L_0, ..., L_n]$ mit den einzelnen Wiederholungsmustern L_i pro Position.
3. Die kompakte Darstellung des Projektionsmusters wird entpackt (siehe Abbildung 5.26) und es ergibt sich das vollständige und korrekte Wiederholungsmuster L' der Projektionsebene π^v.

Das Vorgehen der Transformation bei mehrelementigen Mengen von Positionsangaben für Projektionsebenen ist bezogen auf unser Beispiel aus Abbildung 5.25 in Abbildung 5.27 dargestellt.



(a) Vereinfachte Leveldarstellung (b) Kompakte Leveldarstellung und Entpacken der kompakten Darstellung des Projektionsmusters L_2



(c) Entpacken der kompakten Leveldarstellung mit vollständigen Muster der Projektionsebene L_2

Abbildung 5.27: Transformation von Projektionsebenen bei mehrelementigen Positionsmengen

Interessanterweise stellt der Fall der mehrelementigen Positionsangaben den allgemei-

nen Fall dar, der den einfachen Fall der einelementigen Positionsmenge beinhaltet. Im Algorithmus muss lediglich die Vorgehensweise bei der mehrelementigen Positionsmenge implementiert werden. Es resultiert dieselbe Lösung wie für den einelementigen Fall.

5.5 Transformation: Auswertung von Auswahlebenen

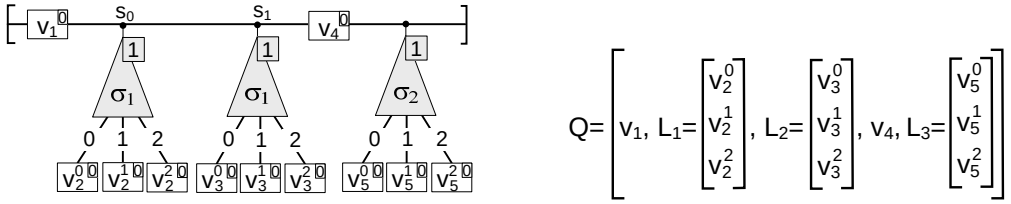
In den bisherigen Betrachtungen zur Ableitung von Wiederholungsmustern haben wir den Schritt der Transformation von Wiederholungsmustern lediglich in Verbindung mit Projektionsebenen besprochen. Bei Auswahlebenen hingegen sind wir stets davon ausgegangen, dass diese nicht von dem Transformationsschritt betroffen sind. Allerdings gilt dies lediglich für einen bestimmten Fall von Auswahlebenen, der bisher auch immer in den Beispielen zugrundegelegt worden ist. Im Allgemeinen muss auch für Auswahlebenen bzw. deren Wiederholungsmuster ein entsprechender Transformationsschritt vorgenommen werden.

Im Folgenden werden zunächst die Fälle, die einen zusätzlichen Transformationsschritt benötigen, vorgestellt. Anschließend zeigen wir, dass die Theorie der periodischen Projektion auch in diesen Fällen anwendbar bleibt. Abschließend werden das konkrete Vorgehen zur Transformation von Auswahlebenen und die Verwendung der entsprechenden Auswahlparameter vorgestellt.

5.5.1 Abhängige Auswahlebenen und nicht-auswahlerhaltende Operationsreihenfolgen

Die bisherigen Betrachtungen zu Auswahlebenen sind indirekt stets davon ausgegangen, dass die Ebenen innerhalb eines Levels unabhängig voneinander sind. Anders ausgedrückt, jede Auswahlebene war das Ergebnis jeweils einer einzelnen Auswahloperation. Die bei der Operation anzugebende Auswahlmenge von Knoten entsprach hierbei immer einer einelementigen Menge.[10] Eine Auswahloperation mit mengenwertiger Auswahlmenge, d.h. eine einzelne Auswahloperation für eine Menge von Knoten, führt zu *abhängigen Auswahlebenen* (siehe Abschnitt 4.2.2). Abbildung 5.28 zeigt ein Level mit zwei abhängigen Auswahlebenen $(\sigma^{s_0}, \sigma^{s_1})$. Die Abhängigkeit beider Auswahlebenen voneinander wird in dem Beispiel durch die gleiche Bezeichnung der Auswahlebenen σ^{s_0} und σ^{s_1} als Auswahlebene σ_1 dargestellt. Von "außen" ist die Abhängigkeit der Auswahlebenen nicht direkt erkennbar. Wie jedoch in Abschnitt 4.2.2 gezeigt, wird die Abhängigkeit über die Auswahlparameter der jeweiligen Auswahlebenen repräsentiert (siehe später). Zusätzlich enthält der Level in Abbildung 5.28 noch die unabhängige Auswahlebene σ_2, um den Unterschied zur Auswertung abhängiger Auswahlebenen besser darstellen zu können.

[10] Wie in Abschnitt 4.2.2 gezeigt, entspricht die Auswahlmenge der Menge an Knoten, für die zum Zeitpunkt der Auswahloperation eine entsprechende Auswahlebene angelegt wird.



(a) Vereinfachte Darstellung (b) Kompakte Leveldarstellung

Abbildung 5.28: Level mit mehreren abhängigen Auswahlebenen

Da die Abhängigkeit von Auswahlebenen von "außen" nicht direkt ablesbar ist, wird mit diesen Ebenen im ersten Schritt des Algorithmus (Schritt 2.1.) wie bisher umgegangen. Das heißt, für jede der Auswahlebenen wird das individuelle Wiederholungsmuster bestimmt, und es ergibt sich die in Abbildung 5.28b gezeigte kompakte Leveldarstellung.

Allerdings zeigt sich die Besonderheit abhängiger Auswahlebenen beim Versuch die entsprechende kompakte Leveldarstellung zu entpacken. Bei unabhängigen Auswahlebenen wird mit jedem Durchlauf des kompakten Levels der Reihe nach die Knoten aus der entsprechenden Knotenliste L ausgewählt. Im vorliegenden Beispiel wird somit beim ersten Durchlauf der erste Knoten der (unabhängigen) Auswahlebene σ^2 bzw. aus der Knotenliste L_3 ausgewählt, beim zweiten Durchlauf der zweite Knoten aus L_3 usw.. Bei abhängigen Auswahlebenen ändert sich dieses Vorgehen jedoch. Aufgrund der Semantik abhängiger Auswahlebenen (siehe Abschnitt 4.2.2.3) muss beim ersten Durchlauf des kompakten Levels beim ersten Vorkommen einer abhängigen Ebene (σ^{s_0}) der erste Knoten aus der Liste (L_1) ausgewählt werden. Das Vorkommen der nachfolgenden abhängigen Ebene (σ^{s_1}) muss im ersten Durchlauf jedoch global wie das zweite Vorkommen der Ebene bewertet und folglich der zweite Knoten v_3^1 ausgewählt werden. Vereinfacht ausgedrückt, die Menge an abhängigen Auswahlebenen in einem Level müssen bei der Auswertung wie eine einzelne Auswahlebene behandelt werden. Dies logische Vorgehen des Entpackens zeigt Abbildung 5.11 für das Beispiel aus Abbildung 5.28. Die Abbildung zeigt auch, dass die unabhängige Auswahlebene σ^2 "normal", d.h. die Knotenliste L_3 der Reihenfolge nach, ausgewertet wird.

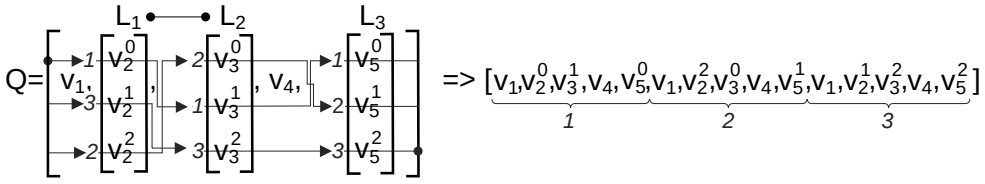


Abbildung 5.29: Logische Auswertung der kompakten Leveldarstellung eines Levels mit mehreren abhängigen Auswahlebenen

Die logische Vorgehensweise zum Entpacken der kompakten Leveldarstellung bei abhän-

gigen Auswahlebenen ist semantisch korrekt, d.h. sie spiegelt die Semantik abhängiger Auswahlebenen korrekt wider. Allerdings weist diese Vorgehensweise ein entscheidendes Problem auf. Wie in Abschnitt 4.2.5 dargelegt, wird bei HSP-Bäumen die Situation zum Zeitpunkt der Anwendung einer Auswahloperation "eingefroren" und durch Auswahl- oder Projektionsparameter abgebildet. Grund dafür ist, dass nach einer Auswahloperation weitere algebraische Operationen zur Beschreibung eines Periodizitätsschemas folgen können; diese wiederum können die Auswahl bzw. die Repräsentation der Auswahl im HSP-Baum unmittelbar beeinflussen. Trotz Anwendung solch nicht-auswahlerhaltender Operationsreihenfolgen gilt es, die ursprüngliche Semantik einer Auswahl bzw. entsprechenden Auswahlebene zu erhalten. Hierzu werden die Projektions- und Auswahlparameter solcher Auswahlebenen bei der Auswertung verwendet und nicht die zum Zeitpunkt der Auswertung vorliegende Situation im HSP-Baum bzw. in den entsprechenden Levels. Die vorgestellte Vorgehensweise des Entpackens abhängiger Auswahlebenen erweist sich insbesondere im Zusammenhang mit nicht-auswahlerhaltenden Operationsreihenfolgen als problematisch.

Bei unabhängigen Auswahlebenen werden beim Entpacken der kompakten Leveldarstellung das Wiederholungsmuster einer Auswahlebenen bzw. die Knoten in der Liste L entsprechend ihrer Reihenfolge (einfach oder mehrfach) durchlaufen (siehe Muster L_3 in Abbildung 5.29). Bei abhängigen Auswahlebenen hingegen werden die voneinander abhängigen Muster bzw. Listen wie einzelne Listen behandelt. Dies wiederum führt dazu, dass die Auswahl eines Knotens aus einer Liste L_i stets in Abhängigkeit von der Position in der vorherigen (abhängigen) Liste L_{i-1} erfolgen muss. In Abbildung 5.29 ist dies für die Listen L_1 und L_2 der Fall. Hier müssen im ersten Durchlauf des kompakten Levels bei Liste L_2 bereits der zweite Knoten ausgewählt werden und beim zweiten Durchlauf der dritte Knoten von L_1 sowie der vierte Knoten von L_2.

Ausgehend davon, dass der Abbildung 5.29 dargestellte Level seit dem Zeitpunkt der Anwendung der Auswahloperation unverändert ist, stellt das obige Vorgehen beim Entpacken noch kein Problem dar. Allerdings wird es problematisch, wenn durch eine nachfolgende, nicht-auswahlerhaltende Operation, beispielsweise, die zweite (abhängige) Auswahlebene L_2 gelöscht worden wäre. Abbildung 5.30 zeigt den resultierenden Level nach dem Löschen der Auswahlebene L_2. Gemäß der erwähnten Semantik im HSP-Baum müssten die verbliebene erste Ebene bzw. das Muster L_1 weiterhin wie bisher ausgewertet bzw. deren Knoten in der entsprechenden Reihenfolge durchlaufen werden. Das Ergebnis der Auswertung wäre dann das rechts dargestellte Wiederholungsmuster. Im Vergleich unterscheiden sich die beiden Ergebnismuster in Abbildung 5.29 und Abbildung 5.30 lediglich darin, dass die Knoten der Liste L_2 im letzteren Muster fehlen. Die Reihenfolge der Knoten von Liste L_1 im Ergebnismuster ist jedoch weiterhin identisch.

Die Ableitung eines solchen Ergebnismusters, welches der Semantik im Umgang mit nicht-auswahlerhaltenden Operationen entspricht, erweist sich mit dem vorgestellten Vorgehen zum Entpacken als unmöglich. Da die zweite Liste L_2 fehlt bzw. gelöscht ist, kann die Auswahl eines Knotens aus Liste L_1 nicht mehr in Abhängigkeit derselben erfolgen. Anders ausgedrückt, das vorgestellte Entpacken der kompakten Leveldarstellung

würde bei abhängigen Auswahlebenen und nicht-auswahlerhaltenden Operationsreihenfolgen bzw. deren Ergebnisse im HSP-Baum unmittelbar beeinflusst.

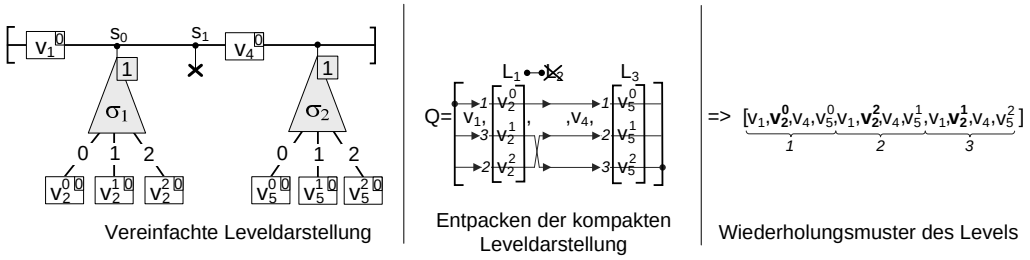


Abbildung 5.30: Semantisch korrekte Auswertung eines Levels nach dem Löschen einer der abhängigen Auswahlebenen

Ziel muss es daher sein, ein Vorgehen zu entwickeln, dass auch im Fall abhängiger Auswahlebenen und nicht-auswahlerhaltender Operationsreihenfolgen eine korrekte Auswertung gemäß der vorgegebenen Semantik ermöglicht. Wie wir im Folgenden zeigen werden, kann hierfür die Theorie der periodischen Projektion (vgl. Abschnitt 5.3.2) verwendet werden. Auf Grundlage der periodischen Projektion können abhängige Auswahlebenen auch bei nicht-auswahlerhaltender Operationsreihenfolge korrekt ausgewertet werden.

5.5.2 Abhängige Auswahlebenen und die periodische Projektion

Ausgangspunkt unserer Betrachtungen bildet die Semantik einer Auswahloperation (siehe Abschnitt 3.4.3.1). Auswahloperationen ermöglichen es dem Anwender, basierend auf einem Auswahlschema, bestimmte Zyklen, Teilzyklen oder Zyklusspositionen eines Periodizitätsschemas auszuwählen bzw. zu markieren. Grundlage bildet die Angabe einer Auswahlmenge, d.h. einer Menge an Elementen des Periodizitätschemas, aus der durch das Auswahlschema periodisch bestimmte Elemente ausgewählt werden. Aus formaler Sicht wird die Auswahlmenge als ein (unendlich) diskreter Wertebereich verstanden, in dem ein einzelner Wert einem Teilzyklus oder einer Zyklusposition aus der Auswahlmenge entspricht. Durch die Auswahloperation wird das Auswahlschema einfach über den Elementen der Auswahlmenge interpretiert, d.h. es erfolgt dieselbe Interpretation wie bei Periodizitätschemas über einem diskreten Wertebereich. Abbildung 5.31 zeigt ein Beispiel für die Interpretation des Auswahlschemas über der Auswahlmenge. Der Wertebereich entspricht der Menge aller Medikationsteilzyklen mit Label `Med` (etwa einer Chemotherapie). Das Auswahllabel `PA` (**P**eriodische **A**uswahl) bezeichnet bei entsprechender Interpretation des Auswahlschemas über der Auswahlmenge die auszuwählenden Medikationsteilzyklen, d.h. "*Jeden 2. Medikationsteilzyklus beginnend beim Ersten*".

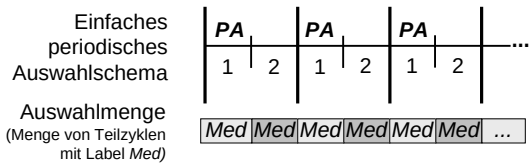


Abbildung 5.31: Beispiel einer periodischen Auswahl

In HSP-Bäumen können, im Gegensatz zum allgemeinen Modell, sowohl das Auswahlschema als auch die Auswahlmenge in endlicher Form repräsentiert werden. Die Auswahlmenge wird hierbei durch eine endliche Anzahl an Knoten repräsentiert und das Auswahlschema wird durch die resultierende Auswahlebene(n) dargestellt (siehe Abschnitt 4.2.2). Dies wiederum ermöglicht es sowohl die endliche Auswahlmenge als auch das Auswahlschema in Form von Wiederholungsmustern darzustellen. Abbildung 5.32 zeigt die Übertragung des Beispiels aus Abbildung 5.31 auf endliche Wiederholungsmuster. Die endliche Repräsentation der Auswahlmenge entspricht einem Wiederholungsmuster (M_1) in dem alle Musterpositionen belegt sind. Das heißt, sie entspricht einem Muster, das alle Werte des grundlegenden Wertebereichs, d.h. alle Elemente der (theoretisch) unendlichen Auswahlmenge, repräsentiert. Die Länge des Wiederholungsmusters für die Auswahlmenge ergibt sich aus dem Umfang der endlichen Auswahlmenge im HSP-Baum. Im Beispiel sind wir davon ausgegangen, dass die Auswahlmenge im HSP-Baum drei Knoten umfasst und folglich das Muster ebenfalls die Länge 3 aufweist. Das Auswahlschema wird durch das Muster M_2 repräsentiert, wobei die Belegung der Position m_0 dem Label `PA` entspricht.

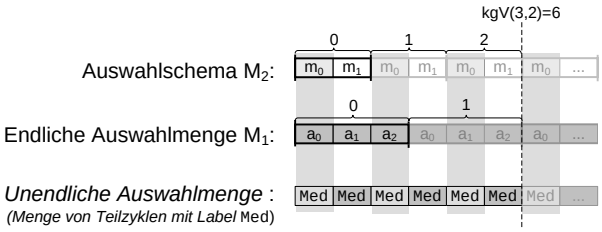


Abbildung 5.32: Darstellung der periodischen Auswahl als Wiederholungsmuster

Ausgehend von dieser Darstellungsform der Auswahl (Abbildung 5.32) zeigt sich, dass bei Auswertung voneinander abhängiger Auswahlebenen im Kern das gleiche Problem wie bei der Projektion zu lösen ist. Das heißt, auch hier werden nur genau die Elemente aus der Auswahlmenge ausgewählt, die mit bestimmten Positionen (m_0) bzw. Labels (`PA`) des Auswahlschemas korrespondieren (siehe Abbildung 5.18). Folglich kann die Theorie der Projektion analog auch für die Auswertung abhängiger Auswahlebenen erfolgen, was durch folgende, alternative Darstellungsform der Projektion verdeutlicht wird. Die Projektion erfolgt dabei "aus Sicht" der Auswahlmenge bzw. des Musters M_1.

Das heißt, die Auswahlmenge entspricht dem Wiederholungsmuster $M_1 = (a_0, a_1, a_2)$, und das Auswahlschema wird durch das Muster $M_2 = (m_0, m_1)$ repräsentiert:

w	a_0	a_1	a_2
0	m_0	m_1	m_0
1	m_1	m_0	m_1

Die Reihenfolge der Elemente in jeder Spalte a_i $(i = 1, 2, 3)$ entspricht der Reihenfolge bzw. dem Wiederholungsmuster, in welcher die einzelnen Elemente der Auswahlmenge mit den Elementen des Auswahlschemas zusammenfallen. Für unser Beispiel bedeutet dies, dass für das Element a_1 der Auswahlmenge die Reihenfolge des Zusammenfallens mit den Positionen des Auswahlschemas bzw. Musters M_2 durch den Vektor $v = (m_1, m_0)$ beschrieben wird (bzw. dieser Vektor mit Hilfe der Projektion ermittelt werden kann).

Übertragen auf den HSP-Baum bedeuten diese Überlegungen, dass ausgehend von der Menge voneinander abhängiger Auswahlebenen $\sigma = [\sigma_0, ..., \sigma_m]$ für jede Auswahlebene σ_i das korrekte Wiederholungsmuster separat und unabhängig von den anderen (eigentlich abhängigen) Auswahlebenen bestimmt werden kann. Hierzu müssen lediglich für jede abhängige Auswahlebene der Umfang m der Auswahlmenge (z.B. $|M_1| = 3$), die Länge des Auswahlschemas (z.B. $|M_2| = 2$) sowie die Position i innerhalb der Auswahlmenge (z.B. $i \in \{0, 1, 2\}$) bekannt sein. Auf Basis dieser Grundwerte kann durch die Anwendung der periodischen Projektion das korrekte Wiederholungsmuster pro Auswahlebene berechnet werden. Im Gegensatz zur Projektion wird jedoch keine Rotation bzw. Berücksichtigung von Instanziierungspunkten benötigt.

5.5.3 Auswahlebenen und Auswahlparameter

Der Vorteil bei der Übertragung der periodischen Projektion auf Auswahlebenen liegt darin, dass beim Entpacken der kompakten Leveldarstellung die Abhängigkeit von Auswahlebenen nicht mehr berücksichtigt werden muss. Stattdessen wird zuvor für jede abhängige Auswahlebene bzw. deren individuelles Wiederholungsmuster das korrekte Wiederholungsmuster mit Hilfe der Projektionsfunktion $\mathcal{P}$ gebildet. Die korrekten Wiederholungsmuster können dann mit der bereits beschriebenen Vorgehensweise in Schritt 2.3 entpackt werden.

Vergleichbar zu Projektionsebenen werden auch bei Auswahlebenen die für die Projektionsfunktion notwendigen Parameter nicht zum Zeitpunkt der Auswertung ermittelt. Stattdessen werden (alle) diese Parameter zum Zeitpunkt der Auswahloperation bestimmt und in den Auswahlparametern hinterlegt. Wie gezeigt, ist es für die Auswertung von abhängigen Auswahlebenen ausreichend, für jede Ebene die Position i innerhalb der Auswahlmenge, den Umfang der Auswahlmenge sowie die Länge des Auswahlschemas zu

kennen. Das heißt für die Auswahlparameter $SP^n = (z, l_P, l_A, Pos)$ von Auswahlebenen ($NT^n = \texttt{SNODE}$) ergibt sich die in Tabelle 5.3 dargestellte Semantik bzw. Zuordnung.

Parameter	Semantik
z	Instanziierungspunkte
l_P	Umfang der Auswahlmenge
l_A	Länge des Auswahlschemas
$Pos = \{i\}$	Position i innerhalb der Auswahlmenge

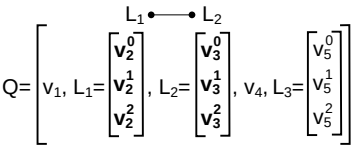
Tabelle 5.3: Semantische Zuordnung der Auswahlparameter

Der Parameter z bzw. der Instanziierungspunkt ist nur aus Gründen der Vollständigkeit aufgeführt und wird bei der Auswertung nicht berücksichtigt. Der Parameter l_P repräsentiert den Umfang der Auswahlmenge und der Parameter l_A die Länge des Wiederholungsmusters des Auswahlschemas. Der Parameter $Pos := \{x \mid 0 \leq x < l_P\}$ enthält die Positionen einer jeden Ebene innerhalb der Auswahlmengen bzw. der Menge der abhängigen Auswahlebenen. Im Gegensatz zu Projektionsparametern gilt bei Auswahlparametern stets $|Pos| = 1$. Dies ergit sich dadurch, dass bei Auswahlebenen bzw. bei deren Bestimmung die konkrete Eigenschaft des Levels keine Rolle spielt. Es stehen nur die Auswahlmenge und das Auswahlschema im Vordergrund. Variierende Positionsangaben für Auswahlebenen (wie etwa bei Projektionsebenen) sind nicht möglich und daher auch keine mehrelementige Menge an Positionsangaben.

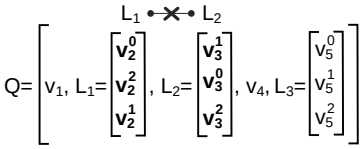
Ausgehend von den Auswahlparametern $SP^v = (z, l_P, l_A, Pos = \{i\})$ einer (abhängigen) Auswahlebene σ^v und seinem individuellen Wiederholungsmuster L wird dessen korrektes Wiederholungsmuster L' durch den Funktionsaufruf $\mathcal{P}(L, z, l_P, l_A, i) = L'$ gebildet (Funktion $\mathcal{P}$ siehe Abschnitt 5.4). Abbildung 5.33 skizziert für das einleitende Beispiel aus Abbildung 5.28 dieses Vorgehen bei abhängigen Auswahlebenen. Abbildung 5.33a zeigt die kompakte Leveldarstellung der beiden abhängigen Auswahlebenen σ^{v_2} und σ^{v_3} bzw. deren individuellen Wiederholungsmustern L_1 und L_2. In dem Beispiel weist die Ebene σ^{v_2} die Auswahlparameter $SP^{v_2} = (0, 2, 3, Pos = \{0\})$ und die Ebene σ^{v_3} die Auswahlparameter $SP^{v_3} = (0, 2, 3, Pos = \{1\})$ auf, d.h. die Auswahlmenge umfasst zwei Positionen ($l_P = 2$) und die Länge des Auswahlschemas = 3 ($l_A = 3$). Mit Hilfe der Funktionsaufrufs $\mathcal{P}(L_1, 2, 3, 0)$ bzw. $\mathcal{P}(L_1, 2, 3, 1)$ kann das korrekte Wiederholungsmuster für Muster L_1 bzw. L_2 gebildet werden. Ebenfalls dargestellt ist das im Schritt 2.3. folgende Entpacken der transformierten kompakten Leveldarstellung.

5.5.3.1 Abhängige Auswahlebenen und Verkürzung von Mustern

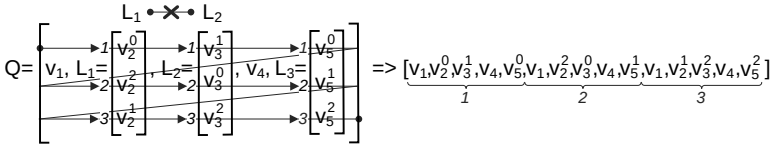
Aufgrund der vorgestellten Auswertung von abhängigen Auswahlebenen gelten die gleichen Eigenschaften der periodischen Projektion wie in der Theorie beschrieben. Insbesondere gilt dies auch für Korollar 5.2, das u.a. besagt, dass im Fall nichtteilerfremder



(a) Vor Anwendung der Projektionsfunktion



(b) Nach Anwendung der Projektionsfunktion



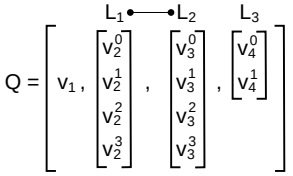
(c) Entpacken der transformierten kompakten Leveldarstellung

Abbildung 5.33: Transformation der kompakten Leveldarstellung mit abhängigen Auswahlebenen

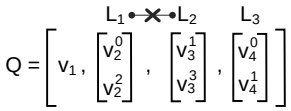
Musterlängen das Ergebnis der Projektion kürzer ist als das ursprüngliche bei der Projektion verwendete Muster. Übertragen auf die Auswertung abhängiger Auswahlebenen bedeutet dies, falls der Umfang der Auswahlmenge und die Länge des Musters des Auswahlschemas nicht teilerfremd sind, dass das korrekte Wiederholungsmuster einer jeden (abhängigen) Auswahlebenen kürzer als sein individuelles Muster ist. Abbildung 5.34 zeigt ein entsprechendes Beispiel. Als Auswahlparameter für die Auswahlebene bzw. dessen Muster L_1 ergibt sich $SP = (0, l_P = 2, l_A = 4, Pos = \{0\})$ bzw. für das Muster L_2 erhalten wir die Parameter $SP = (0, 2, 4, Pos = \{1\}$, wobei jeweils gilt $\texttt{ggT}(l_p, l_A) \neq 1$. Daraus folgt, dass das Ergebnis für Muster L_1 durch ein kürzeres Wiederholungsmuster L' repräsentiert wird: $|L'| = \frac{l_A}{\texttt{ggT}(l_P, l_A)} = 2$ (siehe Korollar 5.2). Diese Kürzung des Musters ergibt sich aufgrund der Implementierung der Projektionsfunktion $\mathcal{P}()$ automatisch. Das heißt, die Anwendung der Funktion für L_1 bzw. L_2 ergibt $\mathcal{P}(L_1, 2, 4, 0) = [v_2^0, v_2^2]$ bzw. $\mathcal{P}(L_2, 2, 4, 1) = [v_3^1, v_3^3]$. Abbildung 5.34b zeigt die kompakte Leveldarstellung nach erfolgter Projektion für das Beispiel aus Abbildung 5.34a. Abbildung 5.34c zeigt das Entpacken der kompakten Leveldarstellung aus Abbildung 5.34b.

5.5.3.2 Unabhängige Auswahlebenen und Auswahlparameter

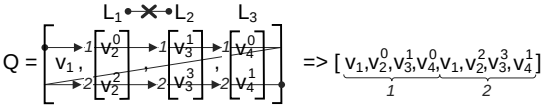
Im Zusammenhang mit Auswahlparametern gilt es anzumerken, dass entsprechende Parameter nicht nur für abhängige sondern auch für unabhängige Auswahlebenen hinterlegt werden. Dies erspart es im Algorithmus zur Transformation von Auswahlebenen (siehe Abschnitt 5.7), zwischen abhängigen und unabhängigen Auswahlebenen unterscheiden zu müssen. Hierbei wird ein kleiner "Trick" angewendet, der darin besteht, für unabhängige Auswahlebenen für den Parameter l_P (Umfang der Auswahlmenge) den Wert $= 1$ und für $Pos = \{0\}$ anzugeben. Wie mit der Theorie der periodischen



(a) Vor Anwendung der Projektionsfunktion



(b) Nach Anwendung der Projektionsfunktion



(c) Entpacken der transformierten kompakten Leveldarstellung

Abbildung 5.34: Transformation der kompakten Leveldarstellung mit abhängigen Auswahlebenen und verkürzten Ergebnismustern

Projektion leicht nachvollzogen werden kann, liefert eine Projektion eines beliebigen Musters M_2 auf ein Muster M_1 mit Länge 1 wieder M_2. Für die Projektionsfunktion folgt daher $\mathcal{P}(L, z, l_P = 1, l_A, 0) = L$, d.h. für unabhängige Auswahlebenen kann die Funktion aufgerufen bzw. die Projektion durchgeführt werden, als Ergebnis resultiert jedoch wieder das übergebene Muster L. Somit kann für jede Auswahlebene im Algorithmus einfach die Projektionsfunktion aufgerufen werden, ohne eine entsprechende Unterscheidung in abhängig oder unabhängig vornehmen zu müssen..

Wie in den vorausgegangen Abschnitten gezeigt, stellt die Theorie der periodischen Projektion ein solides Fundament für die Transformation von Projektions- und Auswahlebenen dar. Wesentlich ist, dass zur Auswertung der Ebenen bzw. zur Bestimmung der Wiederholungsmuster lediglich zwei (drei) Grundparameter notwendig sind: (die Instanziierungspunkte), die Länge des ersten Wiederholungsmusters M_1 (Muster des Periodizitätsschemas oder der Auswahlmenge) und die Länge des zweiten Wiederholungsmusters M_2 (Muster des Auswahlschemas oder des zu projizierenden Schemas). Auf Basis dieser Grundwerte kann für jede Position aus Muster M_1 die Reihenfolge bzw. das Wiederholungsmuster ermittelt werden, wie jede Position von M_1 mit den Positionen eines zweiten Musters M_2 zusammenfallen.

5.6 Transformation: Berücksichtigung von Levelabhängigkeiten

In den bisherigen Betrachtungen haben wir den Umgang mit Auswahl- und Projektionsebenen innerhalb eines einzelnen Levels berücksichtigt. Noch nicht betrachtet haben wir dagegen den bei der Traversierung entstehenden Übergang von Level zu Level. Wie wir noch sehen werden, können bei Auswertung eines einzelnen Levels jedoch Abhängigkeiten zu anderen (tieferliegenden) Levels entstehen, die es zu berücksichtigen gilt. Im Folgenden werden diese Abhängigkeiten zwischen den Levels im Fokus stehen. Hierzu werden wir im ersten Schritt zeigen, wie und wo entsprechende Abhängigkeiten überhaupt entstehen können. Im zweiten und dritten Schritt wird der Umgang mit den Abhängigkeiten in Form eines *Levelübertrags* sowie dessen Erzeugung vorgestellt. Wir werden zeigen, dass die Theorie der periodischen Projektion auch hier eine unverzichtbare Basis für den Umgang mit Levelabhängigkeiten darstellt.

5.6.1 Entstehung von Levelabhängigkeiten

Die Entstehung von Levelabhängigkeiten steht in unmittelbarem Zusammenhang mit dem Entpacken eines Levels in Schritt 2.3. Im Fall von Auswahl- oder Projektionsebenen wird beim Entpacken die kompakte Leveldarstellung zum Teil "mehrfach" durchlaufen und die entsprechenden Knoten in die Ergebnisliste übertragen. Aufgrund des mehrfachen Durchlaufens können einzelne Knoten bzw. Kopien dieser Knoten ebenfalls mehrfach in dem resultierenden Wiederholungsmuster eines Levels enthalten sein.

Zur Bestimmung des nächst tieferen Levels werden für alle Knoten eines Wiederholungsmusters eines Levels die Nachfolgerknoten bestimmt (siehe Abschnitt 5.2.2). Dies erfolgt auch für mehrfach enthaltene Knoten bzw. kopierte Knoten im Muster eines Levels. Durch das Kopieren und das Bestimmen der Nachfolger pro Kopie wird der gesamte nachfolgende Teilbaum indirekt mit kopiert. Im weiteren Verlauf der Auswertung wird jeder dieser indirekt kopierten Teilbäume ebenfalls von Level zu Level mit traversiert. Abbildung 5.35 zeigt diesen Zusammenhang. In der Ergebnisliste des 1. Levels ist der Knoten v_1 mehrfach enthalten. Durch Kopieren des Knotens und das jeweilige Bestimmen der Nachfolger wird indirekt der nachfolgende Teilbaum, d.h. die Auswahlebene σ_2, mit kopiert.

Die Erzeugung der indirekten Kopien der (Teil-) Bäume führt zu Abhängigkeiten zwischen unterschiedlichen Levels. Das Problem dabei sind die in den kopierten Teilbäumen enthaltenen Auswahl- oder Projektionsebenen. Beispielsweise ist dies im oberen Beispiel der Fall. Unterhalb des Knotens v_1 befindet sich in einem Teilbaum die einzelne Auswahlebene σ_2. Durch das Kopieren des Teilbaums auf dem 1. Level wird bei jeder Kopie des Knotens v_1 indirekt auch die Auswahlebene σ_2 kopiert sowie später (im tieferen 2. Level) traversiert und ausgewertet.

Die indirekt kopierten Auswahlebenen dürfen beim Entpacken der (kompakten) Leveldarstellung nicht unabhängig voneinander ausgewertet werden, da diese Kopien von

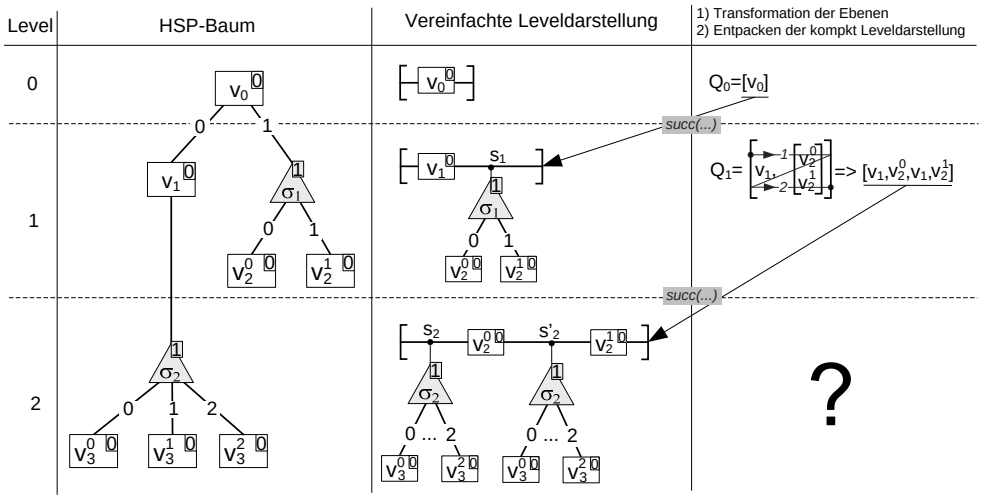


Abbildung 5.35: Darstellung des indirekten Kopierens von Teilbäumen

Auswahlebenen abhängig voneinander sind. Das heißt, in Abbildung 5.35 korrespondiert die Auswahlebene σ_2 im HSP-Baum auf dem 2. Level semantisch mit den einzelnen Vorkommen von Knoten v_1 auf dem 1. Level. Somit korrespondiert das erste Vorkommen von Knoten v_1 im 2.Level mit den Knoten v_3^0 bzw. das zweite Vorkommen von Knoten v_1 mit den Knoten v_3^1.[11] Durch die Kopien des Knotens v_1 im 1. Level, ist diese Form des "Abzählens" des Vorkommens von Knoten v_1 im 2. Level auf die beiden Auswahlebenen σ^{s_2} und $\sigma^{s'_2}$ verteilt wurden (siehe Abbildung 5.35). Die letztere Auswahlebene $\sigma^{s'_2}$ ist eine Kopie der Auswahlebene σ^{s_2}, die sich aufgrund des Kopierens von Knoten v_1 auf dem 1. Level ergibt. Vergleichbar zu abhängigen Auswahlebenen (siehe Abschnitt 5.5.1) müssen zur Berücksichtigung von Levelabhängigkeiten die einzelnen Kopien von Auswahlebenen zusammen wie eine Auswahlebene behandelt werden. Beispielsweise ist in Abbildung 5.36 das semantisch korrekte Entpacken der kompakten Leveldarstellung des 2. Level aus Abbildung 5.35 mit den beiden abhängigen Listen L_0 und L_1 dargestellt. Die Liste L_0 bzw. L_1 bildet das Wiederholungsmuster der Auswahlebene σ^{s_2} bzw. das Muster der Kopie der Auswahlebene $\sigma^{s'_2}$. Im ersten Durchlauf beim Entpacken ergibt sich bei Liste L_0 der erste Knoten v_3^0. Im selben Durchlauf darf in Liste L_1 jedoch nicht ebenfalls der Knoten v_3^0 ausgewählt werden, sondern bereits der zweiten Knoten v_3^1. Dieser Knoten v_3^1 korrespondiert auf dem 1. Level mit dem zweiten Vorkommen des Knotens v_2.

Das dargestellte Verhalten entspricht der Situation, die wir in Verbindung mit abhängigen Auswahlebenen (siehe Abschnitt 5.5.1), bereits vorgefunden haben. Im Kern liegt auch hier das Kopieren von Auswahlebenen vor. Bei abhängigen Auswahlebenen ent-

[11] Dieser Zusammenhang wird besonders deutlich, wenn wir uns die Auswahlebene σ_1 "wegdenken" würden. In diesem Fall wird nur durch die Auswahlebene σ_2 (indirekt) jedes Vorkommen von Knoten v_1 "abgezählt".

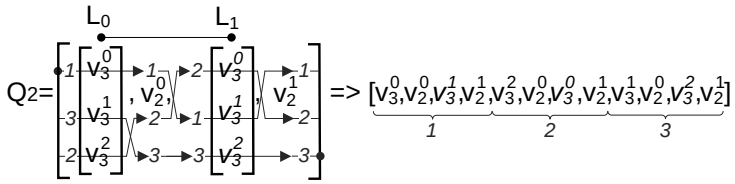


Abbildung 5.36: Semantisch korrektes Entpacken eines Levels bei bestehenden Levelabhängigkeiten

stehen die Abhängigkeiten dadurch, dass wir eine Auswahl für eine mehrelementige Auswahlmenge vornehmen und für jedes Element dieser Auswahlmenge eine Kopie einer Auswahlebene erzeugen. Bei Levelabhängigkeiten hingegen werden entsprechende Kopien bei der Auswertung indirekt und in tieferliegenden Levels erzeugt. Im Resultat führt dies jedoch zu den gleichen Abhängigkeiten. Der einzige Unterschied besteht darin, dass von Levelabhängigkeiten zusätzlich auch Projektionsebenen betroffen sind (siehe später).

5.6.2 Berücksichtigung von Levelabhängigkeiten

Die Berücksichtigung der Levelabhängigkeiten basiert prinzipiell auf der gleichen Idee wie bei abhängigen Auswahlebenen (siehe Abschnitt 5.5.2). Das heißt, die Abhängigkeit zwischen zwei Auswahlebenen kann auf Basis der periodischen Projektion bzw. der entsprechenden Permutation des Wiederholungsmusters aufgelöst werden. Im Kern wird nur die Anzahl bzw. Umfang der (geordneten) Menge voneinander abhängiger Ebenen und deren individuelle Position innerhalb der Menge benötigt (siehe Abschnitt 5.5.2).

Übertragen auf Levelabhängigkeiten können wir analog verfahren. Das heißt, die Abhängigkeit von Ebenen kann durch Anwendung der Projektion aufgehoben werden. Hierzu muss lediglich die Anzahl bzw. Menge an Kopien der Auswahlebene und die entsprechende Position einer jeden Ebene in der Menge bekannt sein. Anders ausgedrückt, muss für jede Ebene bekannt sein, wie viele (indirekte) Kopien dieser Ebene existieren und an welcher Position in dieser Menge sie sich befindet. Für das Beispiel aus Abbildung 5.35 bedeutet dies, dass es sich um eine zweielementige Menge $(\sigma^{s_2}, \sigma^{s'_2})$ an abhängigen Auswahlebenen handelt, mit der Position 0 für σ^{s_2} und die Position 1 für $\sigma^{s'_2}$. Die folgende alternative Darstellungsform der Projektion verdeutlicht diesen Zusammenhang. Die Projektion erfolgt "aus Sicht" der Menge kopierter Auswahlebenen. Das heißt, die Menge an Kopien entspricht dem Wiederholungsmuster $M_1 = (k_0, k_1)$ und die ursprüngliche Auswahlebene σ_2 wird durch das Muster $M_2 = (v_3^0, v_3^1, v_3^2)$ repräsentiert:

w	k_0	k_1
0	v_3^0	v_3^1
1	v_3^2	v_3^0
2	v_3^1	v_3^2

Die Reihenfolge der Elemente in Spalte k_i $(i = 0, 1)$ entspricht dem Wiederholungsmuster einer Kopie der Auswahlebene an seiner entsprechenden Positionen in der Menge der Kopien. Das heißt, die erste Spalte steht für das Muster der Auswahlebene σ^{s_2}, die zweite Spalte für das Wiederholungsmuster der Kopie $\sigma^{s'_2}$.

Zur Berücksichtigung von Levelabhängigkeiten muss vor dem Entpacken der kompakten Leveldarstellung lediglich eine Transformation bzw. eine entsprechende Projektion des Wiederholungsmusters vorgenommen werden. Hierzu muss für jede Ebene ein Tupel $(\#k, pos)$ bekannt sein, bestehend aus der Anzahl an Kopien $(\#k)$ und der jeweiligen Position der Ebene in der (geordneten) Menge an Kopien (pos). Dieses Tupel wird als "Übertrag" bezeichnet. Mit dem Funktionsaufruf $\mathcal{P}(L, \#k, n, pos)$ kann für das Muster L mit Länge n der Übertrag $(\#k, pos)$ berücksichtigt werden, d.h. eine entsprechende Transformation des Musters vorgenommen werden. Für obiges Beispiel bedeutet dies, dass die Anwendung der Projektion auf die entsprechenden Listen L_0 mit Übertrag $(2, 0)$ und L_1 mit Übertrag $(2, 1)$ zu dem Ergebnis $\mathcal{P}(L_0, 2, 3, 0) = [v_3^0, v_3^2, v_3^1]$ bzw. $\mathcal{P}(L_1, 2, 3, 1) = [v_3^1, v_3^0, v_3^2]$ führt.

Nach erfolgter Transformation der einzelnen Muster zur Bereinigung von Levelabhängigkeiten, kann die kompakte Leveldarstellung wie "gewohnt" entpackt werden (siehe Abbildung 5.37). Das Ergebnismuster entspricht dem Muster des in Abbildung 5.36 dargestellten (semantisch korrekten) Entpackens von Ebenen bei Levelabhängigkeiten.

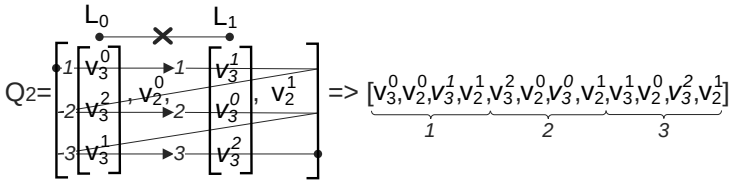


Abbildung 5.37: Entpacken der kompakten Leveldarstellung nach der Bereinigung von Levelabhängigkeiten

Auch wenn bisher Levelabhängigkeiten lediglich bei Auswahlebenen betrachtet worden sind, gelten die getroffenen Aussagen auch für Projektionsebenen. Beispielsweise könnte die im vorherigen Beispiel verwendete Auswahlebene σ_2 durch eine Projektionsebene ersetzt werden. Auch in diesen Fall würden durch das Kopieren des Knotens v_1 auf dem 1. Level indirekt Kopien der Projektionsebene auf dem 2. Level erzeugt werden. Die dadurch entstehenden Abhängigkeiten können dann analog wie oben dargestellt berücksichtigt werden.

In den vorherigen Abschnitten vorgestellten Transformationen von Auswahl- und Projektionsebenen fand die Bestimmung der entsprechenden Auswahl- und Projektionsparameter zum Zeitpunkt des "Erzeugen" der Ebene statt. Im Unterschied dazu, wird in der vorliegenden Transformation die Parameter bzw. der eingeführte Tupel "Übertrag" erst zum Zeitpunkt der Auswertung ermittelt. Im Folgenden werden die Berechnung und die Übergabe des Übertrags von Level zu Level näher betrachtet.

5.6.3 Umgang mit den Übertrag von Level zu Level

Für die Bereinigung von Levelabhängigkeiten ist die Bestimmung des Übertrags eines Levels wesentlich. Die Berechnung des Übertrages erfolgt stets nach dem Entpacken der kompakten Leveldarstellung und vor Bestimmung der Knoten des nächst tieferen Levels, in Bezug auf den Auswertungsalgorithmus somit nach Schritt 2.3 und vor Schritt 2.1 (siehe Abbildung 5.2) des nachfolgenden Schleifendurchlaufs.

Weitergabe des Übertrags an den nächst tieferen Level: Der berechnete Übertrag eines Knotens auf Level i wird an alle Nachfolgerknoten auf dem nächst tieferen Level $i+1$ weiter gegeben. Grundsätzlich ist der Übertrag lediglich für Auswahl- oder Projektionsebenen relevant bzw. wird nur bei deren Auswertung berücksichtigt. Aus Gründen der Vereinfachung kann die Übergabe des Übertrags jedoch für jeden Nachfolgerknoten erfolgen – unabhängig vom konkreten Knotentyp. Dies erspart und eine Fallunterscheidung im entsprechenden Auswertungsalgorithmus. Abbildung 5.38 zeigt ein Beispiel. Aufgrund der Auswertung von Auswahlebene σ_1 wird im 1. Level der Knoten v_1 kopiert, d.h. der Knoten v_1' im Levelmuster E_1 ist eine Kopie des Knotens v_1. Daraus folgt für den Knoten v_1 der Übertrag $(2,0)$ und für dessen Kopie v_1' der Übertrag $(2,1)$. Zur Bestimmung des (nächst tieferen) 2. Levels werden für alle im Wiederholungsmuster des 1. Levels enthaltenen Knoten die Nachfolgerknoten bestimmt. Das bedeutet, dass sowohl für den Knoten v_1 als auch für dessen Kopie v_1' die Nachfolgerknoten bestimmt werden. Die ermittelten (unterschiedlichen) Überträge von v_1 und v_1' werden an deren Nachfolger auf dem 2. Level übergeben, d.h der Übertrag $(2,0)$ von v_1 an die Knoten v_4, s_2 (σ_2), v_5 und der Übertrag $(2,1)$ von v_1' an die Knoten v_4', s_2', v_5' (die Kopien der Knoten v_4, s_2, v_5 darstellen).

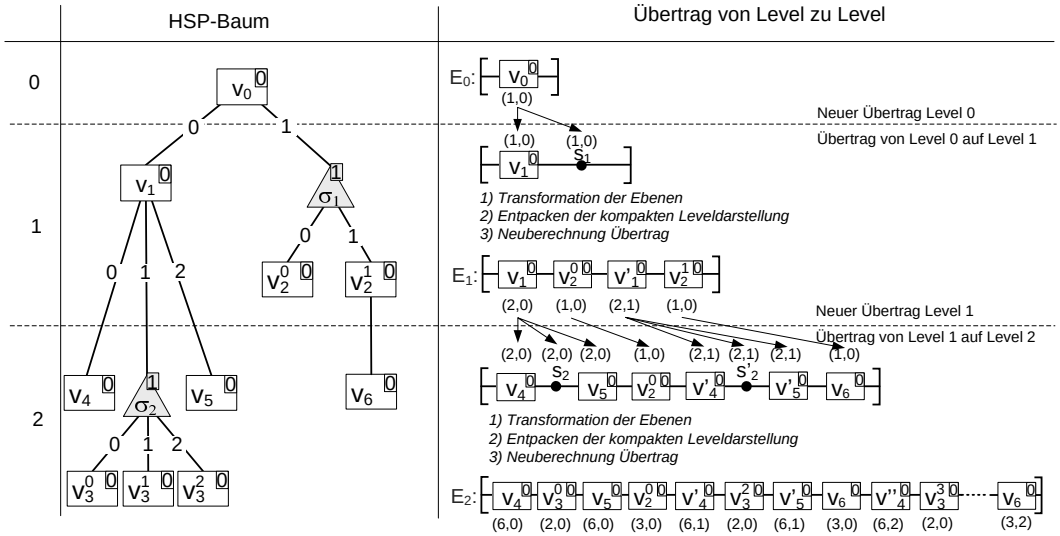


Abbildung 5.38: Übergabe des Übertrags auf den nächst tieferen Level

Neuberechnung des Übertrags: Nach Übergabe des Übertrags an das nächst tiefere Level $i+1$ wird der Übertrag bei Auswahl- oder Projektionsebene wie zuvor dargestellt "verrechnet". Das heißt, es wird eine entsprechende Transformation der Wiederholungsmuster der Ebenen vorgenommen. Der vom höheren Level erhaltene Übertrag wird in einem Level jedoch nur für die Verrechnung verwendet und nicht an den nächst tieferen Level weitergegeben. Stattdessen wird eine Neuberechnung des Übertrags, basierend auf dem Wiederholungsmuster des vorliegenden Levels, durchgeführt. Der neu berechnete Übertrag wird an den tieferen Level weitergegeben. Die Neuberechnung wird aufgrund der Semantik des Übertrags notwendig. Das heißt, nach Berücksichtigung von Levelabhängigkeiten bzw. entsprechender Transformationen in einem Level kann durch das Entpacken der kompakten Leveldarstellung erneut Kopien von Knoten erzeugt werden. Diese neue Anzahl an Kopien ist für den nachfolgenden Level relevant und führt dazu, dass der Übertrag, d.h. die Anzahl an Kopien identischer Knoten und ihre jeweilige Position in der Menge, neu berechnet werden muss. Lediglich im Fall von Levels, die keine Auswahl- oder Projektionsebenen enthalten, können keine neuen Kopien erzeugt werden und (theoretisch) der Übertrag 1:1 aus dem vorherigen Level an den jeweils nächst tieferen Level weitergegeben werden. Dieses Vorgehen zur Neuberechnung des Übertrags ist in Abbildung 5.38 gut zu erkennen.

Wie in den vorausgehenden Abschnitten gezeigt, führt das indirekte "Kopieren" von Auswahl- oder Projektionsebenen bei der Auswertung des HSP-Baums zu Levelabhängigkeiten. Wir konnten jedoch zeigen, dass die Theorie der periodischen Projektion auch hier eine solide Basis für die Berücksichtigung solcher Abhängigkeiten darstellt.

5.7 Algorithmus zur Ableitung von Wiederholungsmustern

Im Folgenden wird eine Implementierung des in vorhergehenden Abschnitten vorgestellten Algorithmus zur Ableitung von Wiederholungsmustern vorgestellt. Hierbei wird die Funktion $getPattern(T, z_T) = L$ eingeführt, die ausgehend von einem HSP-Baum T und einem Instanziierungspunkt z_T das entsprechende Wiederholungsmuster L des Baumes erzeugt. Für die Realisierung der Funktionen bzw. der notwendigen logischen Schritte werden entsprechende (Hilfs-) Funktionen eingeführt. Eine Übersicht zeigt Abbildung 5.39. Die Spezifikation der Funktionen $getPattern()$ und ihrer Hilfsfunktionen findet sich in Anhang A.

Allerdings unterscheidet sich die Implementierung von dem vorgestellten logischen Vorgehen des Algorithmus im Zusammenhang mit der kompakten Leveldarstellung. In der vorliegenden Implementierung wird die kompakte Leveldarstellung nicht explizit erzeugt. Stattdessen werden für jede Ebene die Schritte 2.1 (Bestimmung des individuellen Wiederholungsmusters) und 2.2 (Transformation) in Kombination verwendet, d.h. das ermittelte Wiederholungsmuster wird unmittelbar transformiert. Für ein Level wird somit nur die transformierte kompakte Leveldarstellung erzeugt und im Schritt 2.3 entsprechend entpackt. Diese Form der Implementierung erweist sich algorithmisch als

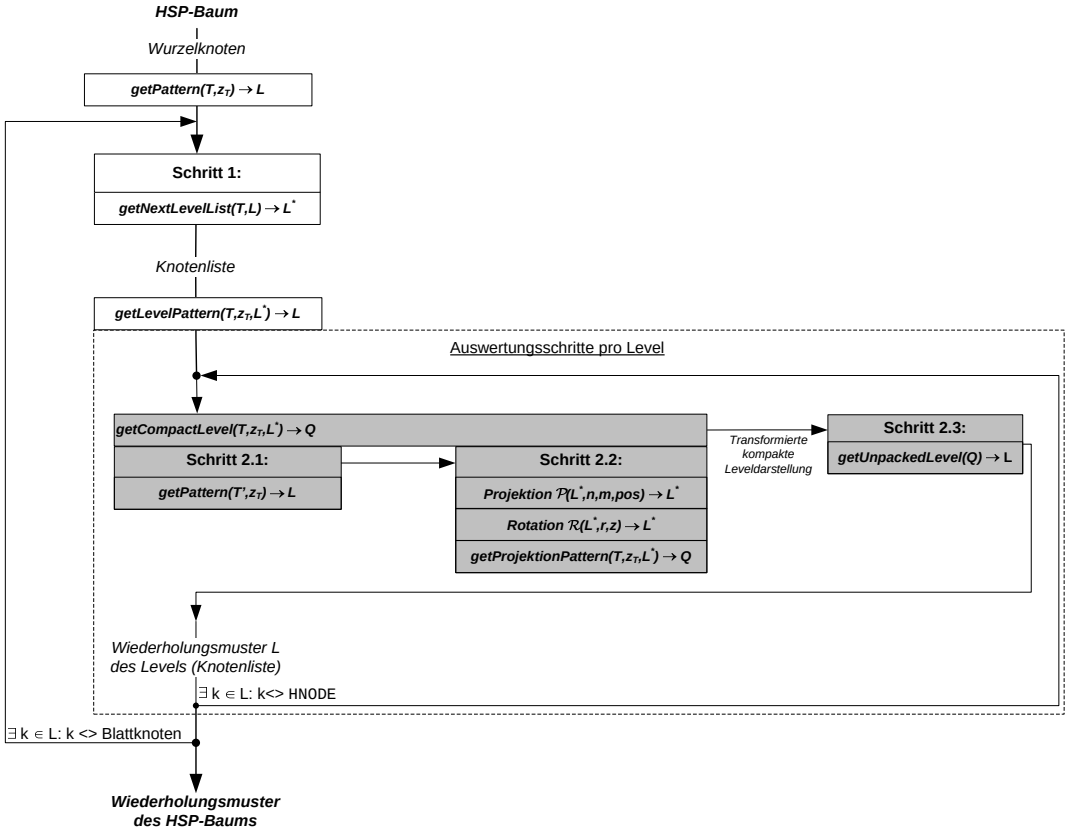


Abbildung 5.39: Übersicht der logischen Schritte des Algorithmus und die korrespondierenden Funktionen

einfacher und vermeidet das mehrmalige Durchlaufen der Leveldarstellung (Erzeugung der kompakten Leveldarstellung und nachträgliches Durchlaufen zur Transformation der Wiederholungsmuster).

5.8 Beispiel

In diesem Abschnitt wird der Algorithmus zur Auswertung von HSP-Bäumen anhand eines konkreten Beispiels dargestellt. Als Grundlage dient unter anderem der ebenfalls im Rahmen dieser Arbeit entwickelte Prototyp (siehe Kapitel 7). Abbildung 5.40 zeigt den HSP-Baum des Beispiels. Es handelt sich um eine Chemotherapie, bestehend aus den zwei Teilzyklen `Medikation` (Länge=4) und `Pause` (Länge=2), wobei zu jeder zweiten `Pause` am letzten Tag ein Bluttest `BT` stattfindet (repräsentiert durch Auswahlebene $\sigma^{S_{13}}$). Im Teilzyklus `Medikation` wird am 1. und 2. Tag das Medikament A (`MedA`) verabreicht. Hierbei besteht der zusätzliche Zusammenhang, dass die Gabe von Medikament A (`MedA`) und die Gabe von Medikament B (`MedB`) einer anderen Therapie besonders

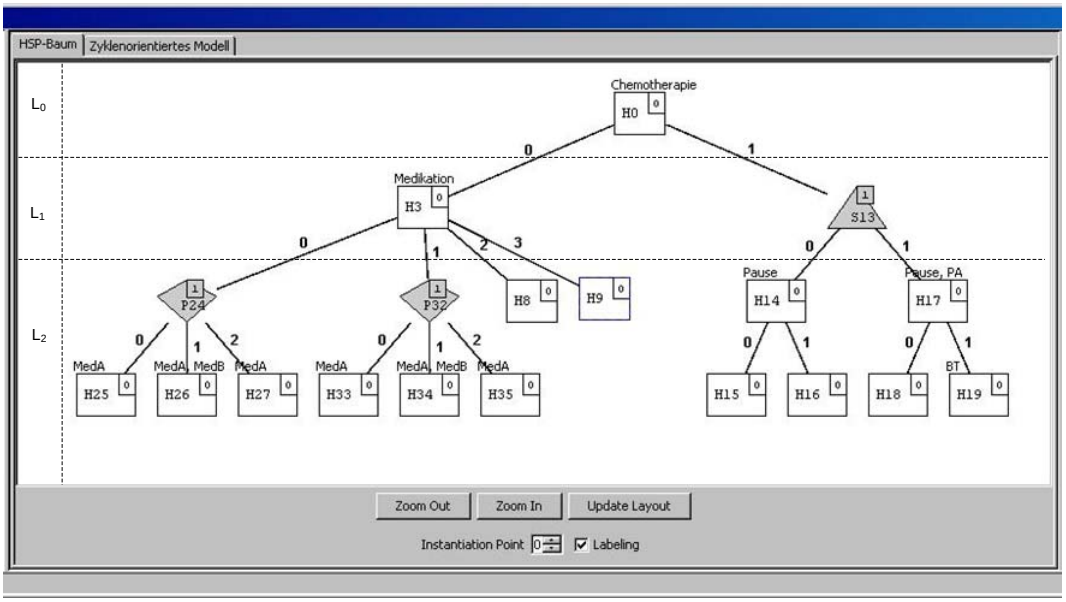


Abbildung 5.40: HSP-Baum des Auswertungsbeispiels

berücksichtigt werden muss. In der entsprechenden Therapie wird das Medikament B am 2.Tag in einem Zyklus von 3 Tagen verabreicht. Mittels einer Projektionsauswahl wurden die Gaben von Medikament A ausgewählt, die mit den Gaben von Medikament B der anderen Therapie zusammenfallen. Im Beispiel wird dies durch die beiden Projektionsebenen $\pi^{P_{24}}$ und $\pi^{P_{32}}$ im HSP-Baum repräsentiert.

Im Folgenden werden wir den Algorithmus und seine Zwischenergebnisse von Level zu Level erklären, d.h. pro Level die Schritte 2.1-2.3 im Detail erläutern (siehe Abbildung 5.39). Hierbei sind in Abbildung 5.41 die im Algorithmus erzeugten Zwischenergebnisse zu sehen. Das Ergebnis der Auswertung pro Level bzw. der gesamten Auswertung zeigt Abbildung 5.42 anhand des Prototypen.

Level L_0: Ausgangspunkt des Algorithmus ist der Wurzelknoten des HSP-Baum, der dem Level L_0 bzw. dem Levelmuster $M_0 = [H_0]$ entspricht (siehe Abbildung 5.41 und 5.42).

Level L_1: Ausgehend von Level L_0 werden im Schritt 1 des Algorithmus alle direkten Nachfolger des Levelmusters M_0 ermittelt ($getNextLevelList()$). Die resultierende vereinfachte Leveldarstellung von Level L_1 zeigt Abbildung 5.41. Der Level L_1 enthält die Auswahlebene $\sigma^{S_{13}}$.

Im **Schritt 2.1** des Algorithmus wird (rekursiv) das Wiederholungsmuster der Auswahlebene $\sigma^{S_{13}}$ ermittelt, was zu der Knotenliste $L = \begin{bmatrix} H_{14} \\ H_{17} \end{bmatrix}$ bzw. zu der kompakten

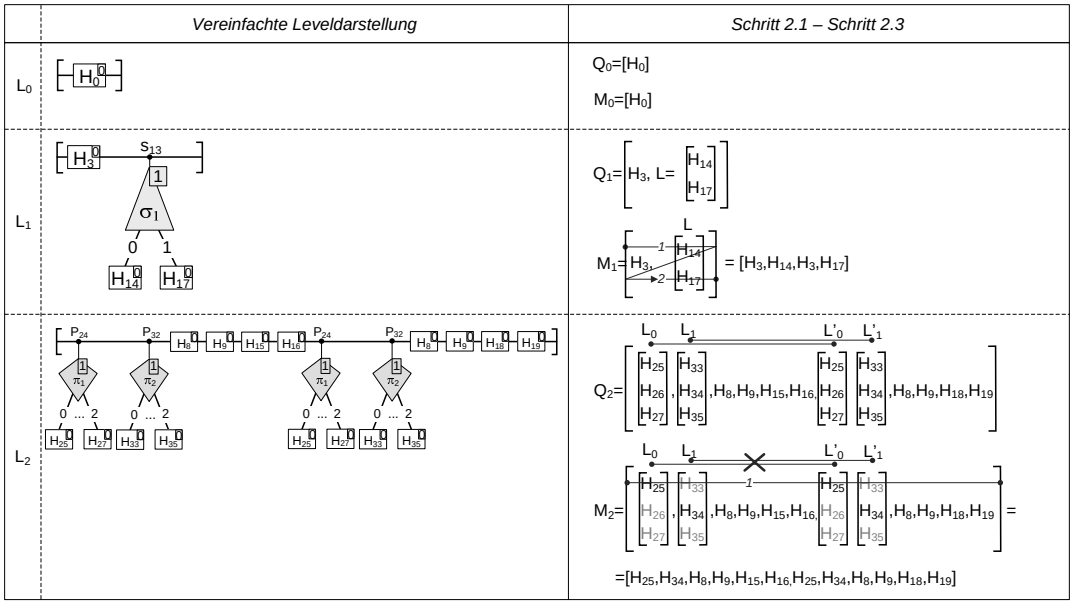


Abbildung 5.41: Darstellung der Zwischenergebnisse der Auswertung pro Level

Leveldarstellung Q_1 von Level L_1 führt (siehe Abbildung 5.41).

Im **Schritt 2.2** des Algorithmus werden die einzelnen Wiederholungsmuster von Auswahl- bzw. Projektionsebenen transformiert. Im vorliegenden Fall handelt es sich bei der Auswahlebene $\sigma^{S_{13}}$, um eine unabhängige Auswahlebene. In diesem Fall wird eigentlich keine Transformation zur Ermittlung des korrekten Wiederholungsmusters benötigt. Wie jedoch in Abschnitt 5.5 gezeigt, wird zur Vereinfachung des Algorithmus dennoch eine solche vorgenommen, wenn auch mit "neutralen" Selektionsparametern. Die Transformation des Musters bzw. Knotenliste $L = \begin{bmatrix} H_{14} \\ H_{17} \end{bmatrix}$ der Auswahlebene $\sigma^{S_{13}}$ führt aufgrund der "neutralen" Selektionsparametern $\sigma^{S_{13}} = (0, n = 1, m = 2, \{0\})$ zu der identischen Liste (siehe Abschnitt 5.5.3.2).

Im **Schritt 2.3** erfolgt das Entpacken der kompakten Leveldarstellung Q_1 (siehe Abschnitt 5.2.5), was zu dem Levelmuster $M_1 = [H_3, H_{14}, H_3, H_{17}]$ führt (siehe Level L_1 in Abbildung 5.41 und 5.42). Abschließend wird für jeden Knoten des Levelmusters M_1 der Levelübertrag ermittelt. Wie gut zu erkennen ist, wurde der Knoten H_3 bei der Auswertung vervielfältigt bzw. ist mehrfach im Levelmuster M_1 enthalten. Wie Abschnitt 5.6 zeigt, führt dies zu dem Übertrag $(2, 0)$ für das erste Vorkommen von H_3 und zum Übertrag $(2, 1)$ für das zweite Vorkommen von H_3. Alle weiteren Knoten erhalten den neutralen Übertrag $(1, 0)$. Abbildung 5.42 zeigt die Überträge für die einzelnen Knoten.

Level L_2: Ausgehend von Level L_1 werden im Schritt 1 alle direkten Nachfolger des Levelmusters M_1 ermittelt ($getNextLevelList()$). Die resultierende vereinfachte Leveldarstellung von Level L_2 zeigt Abbildung 5.41. Gut zu erkennen ist, dass aufgrund der Vervielfältigung von Knoten H_3 in Level L_1 die beiden Projektionsebenen $\pi^{P_{24}}$ (π_1) und $\pi^{P_{32}}$ (π_2) nun jeweils zweimal im Level L_2 enthalten sind.

Im **Schritt 2.1** werden die entsprechenden Wiederholungsmuster der Projektionsebenen ermittelt. Für die Projektionsebene π_1 führt dies zu dem Muster bzw. zu den Knotenlisten $L_0 = L'_0 = \begin{bmatrix} H_{25} \\ H_{26} \\ H_{27} \end{bmatrix}$ bzw. für die Ebene π_2 zu den Listen $L_1 = L'_1 = \begin{bmatrix} H_{33} \\ H_{34} \\ H_{35} \end{bmatrix}$. Aufgrund des Übertrags sind die beiden Vorkommen von π_1 bzw. von π_2 abhängig voneinander (siehe Abschnitt 5.6.2), was bei der nachfolgenden Transformation besonders berücksichtigt werden muss.

Im **Schritt 2.2** findet die bei Projektionsebenen zwingende Transformation statt, um das korrekte Wiederholungsmuster der Ebene zu bestimmen. Die erste Projektionsebene π_1 weist die Selektionsparameter $\pi^{P_{24}} = (0, l_a = 12, l_p = 3, pos = [0, 6])$ auf (siehe Abschnitt 5.4). Das heißt, zum Zeitpunkt der Projektion (nach der Zyklusauswahl auf Teilzyklus `Pause`) wies das Wiederholungsmuster der Therapie mit Medikament A eine Länge von 12 (l_a) auf. Das Projektionsschema (Therapie mit Medikament B) wiederrum eine Länge von 3 (l_p). Die erste Gabe von Medikament A (Zyklusposition 1 im Teilzyklus `Medikation`) entspricht im Muster den Positionen 0 und 6 ($pos = [0, 6]$). Somit ergibt sich die untenstehende Darstellung der periodischen Projektion, wie wir Sie auch in der Vorstellung der theoretischen Grundlagen verwendet haben (vgl. Abschnitt 5.3.2). Im oberen Bereich sind die Positionen des Wiederholungsmuster der Therapie mit Medikament A abgetragen und im unteren Bereich das Muster bzw. Knoten der Therapie mit Medikament B. Hervorgehoben sind die Positionen an denen Gaben von Medikament A an Zyklusposition 1 im Teilzyklus `Medikation` stattfinden.

pos	**0**	1	2	3	4	5	**6**	7	8	9	10	11
0	$\mathbf{H_{25}}$	H_{26}	H_{27}	H_{25}	H_{26}	H_{27}	$\mathbf{H_{25}}$	H_{26}	H_{27}	H_{25}	H_{26}	H_{27}

Aufgrund der Tatsache das die Periodenlängen l_p und l_a der beiden Therapien nicht teilerfremd sind, sondern Vielfache voneinander, ergibt sich als korrektes Muster der Projektionsebene π_1 das Muster $[H_{25}]$. Das heißt, von der Projektionsebene π_1 fällt nur der Knoten H_{25} bzw. die entsprechende Gabe von Medikament B mit einer Gabe von Medikament A (an Zyklusposition 1 im Teilzyklus `Medikation`) zusammen. Die beiden anderen Knoten H_{26} und H_{27} korrelieren niemals mit einer Gaben von Medikament A.[12] Analog verhält es sich mit der Projektionsebene π_2 und ihren Selektionsparametern $SP^{P_{32}} = (0, 12, 3, [1, 7])$, die nach der Transformation zu dem Muster $[H_{34}]$ führt.

[12] Dieser Zusammenhang ist analog zu der Problematik von Dead-Leafs, die wir in Kapitel 4 in Abschnitt 4.3.1.3 vorgestellt haben.

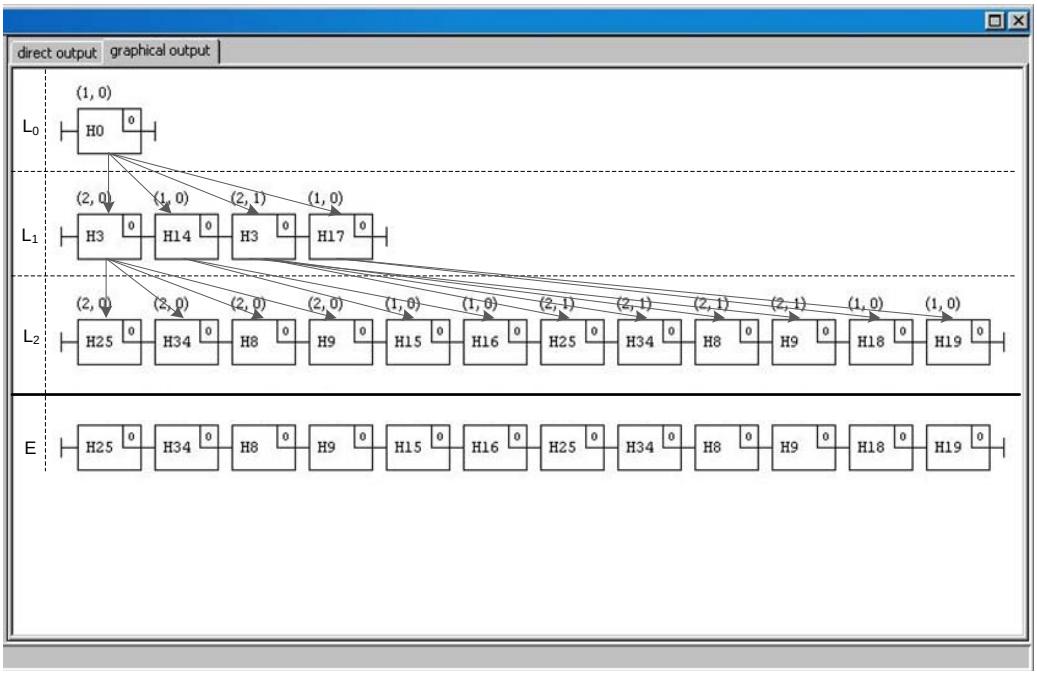


Abbildung 5.42: Ausgabe des Prototyps mit den Endergebnissen pro Level

Nachdem die korrekten Wiederholungsmuster der einzelnen Projektionsebenen ermittelt wurden, muss zusätzlich der entsprechende Übertrag aus dem vorherigen Level berücksichtigt werden. Allerdings ist dies im vorliegenden Fall trivial. Beide Wiederholungsmuster der Projektionsebenen sind lediglich einelementig und die Berücksichtigung des Übertrags führt jeweils wieder zu dem identischem einelementigen Muster (siehe Abschnitt 5.6).

Im **Schritt 2.3** erfolgt das Entpacken der im vorherigen Schritt ermittelten kompakten Leveldarstellung Q_2 statt (siehe Level L_2 in Abbildung 5.41). Die in der Abbildung grau hinterlegten Knoten sind jene Knoten, die bei der Transformation der Muster der Projektionsebenen gelöscht wurden. Es ergibt sich das Levelmuster $L_3 = [H_{25}, H_{34}, H_8, H_9, H_{15}, H_{16}, H_{25}, H_{34}, H_8, H_9, H_{18}, H_{19}]$, dass gleichzeitig auch das Endergebnis der Auswertung, d.h. das Wiederholungsmuster des HSP-Baums darstellt (siehe Abbildung 5.42).

5.9 Zusammenfassung

In diesem Kapitel haben wir die Auswertung des HSP-Baums vorgestellt. Sie ermöglicht die Ableitung von Wiederholungsmustern. Diese Ableitbarkeit stellt eine Mindestanforderung an die Auswertbarkeit sowohl des zyklen-orientierten Modells (siehe Kapitel 3) als auch des HSP-Baums dar, als dessen kompakte und berechenbare Repräsentationsform.

Es zeigt sich, dass die semantisch reichhaltige Repräsentation von Wiederholungsmustern durch HSP-Bäume, den Anwendern zwar eine intuitiv verständliche und nachvollziehbare Beschreibung ermöglicht, für die operationale Unterstützung jedoch ungeeignet ist. Stattdessen werden "einfachere" Wiederholungsmuster benötigt, bei denen horizontale und vertikale Strukturierungen sowie Labelzuweisungen keine bzw. eine untergeordnete Rolle spielen. Die Auswertung von HSP-Bäumen führt zu einer geordneten Menge von Blattknoten, die als ein solches "einfaches" Wiederholungsmuster verstanden werden kann.

Die Grundlage der Auswertung bildete eine angepasste Level-Traversierung des HSP-Baums. Hierbei wird schrittweise für jeden Level ein Levelmuster erzeugt, dass die Grundlage für die Bestimmung des Musters des nächst tieferen Levels bildet. Es zeigt sich hierbei, dass Auswahl- und Projektionsebenen die eigentliche Herausforderung bei der Auswertung darstellen, d.h. die Wiederholungsmuster dieser Ebene zu ermitteln. So zeigt sich, dass bei naiven Ansätzen der Auswertung solcher Ebenen bzw. ihrer Wiederholungsmuster eine ähnliche Problematik wie bei der Realisierung von Vergleichsoperationen zwischen Mustern entsteht. Hieraus ergibt sich die Fragestellung nach effizienteren Ansätzen, um der Problematik bei der Auswertung von Projektions- und Auswahlebenen begegnen zu können.

Wir haben gezeigt, dass die eingeführte Theorie der *periodischen Projektion* einen Ansatz darstellt, der eine effiziente Auswertung von Projektions- und Auswahlebenen ermöglicht. Diese Theorie ermöglicht die effiziente "Berechnung" der Wiederholungsmuster solcher Ebenen auch bei unterschiedlichen Instanziierungszeitpunkten. Darüber hinaus zeigt sich, dass die Theorie auch eine hervorragende Grundlage für eine effiziente Auswertung von (abhängigen) Auswahlebenen darstellt. Die Theorie ermöglicht einen einheitlichen Umgang mit Auswahl- und Projektionsebenen bei der Auswertung. Ausgehend von den eingeführten theoretischen Grundlagen haben wir einen Algorithmus beschrieben, der eine effiziente Realisierung der Auswertung ermöglicht.

Die Ableitbarkeit von Wiederholungsmustern aus HSP-Bäumen bildet die Grundlage für die nachfolgenden Kapitel. In diesen Kapiteln wird eine neue Repräsentationsform vorgestellt, die eine effiziente operationale Unterstützung für solche Muster bzw. ihrer korrespondierenden Periodizitätsschemas ermöglicht.

6 Die Bitletrepräsentation: Eine musterbasierte Realisierung von Vergleichsoperationen

Für den praktischen Einsatz ist eine effiziente operationale Unterstützung von periodischen Spezifikation unumgänglich. In Kapitel 2 haben wir hierfür verschiedene Klassen von Operationen identifiziert und vorgestellt. Hierbei zeigt sich, dass insbesondere die Vergleichsoperationen eine große Herausforderung darstellen. Allerdings zeigt sich auch, dass eine prinzipielle Realisierbarkeit solcher Operationen alleine nicht ausreicht. Insbesondere im Fall einer großen Menge an periodischen Spezifikationen muss auch die Effizienz der resultierenden Implementierung berücksichtigt werden. Wie in Kapitel 2 gezeigt, konnte hier nur eine musterbasierte Realisierung den Anforderungen der operationalen Unterstützung grundsätzlich begegnen. Allerdings fehlt es an adäquaten Ansätzen einer musterbasierten Realisierung für Wiederholungsmustern mit sehr langen Periodenlängen. Aufgrund des Repräsentationsumfangs und dem damit verbundenen operationalen Aufwand sind vorliegende Vorschläge für den praktischen Einsatz weniger geeignet (siehe Abschnitt 2.3.3). Kompaktere Repräsentationen – unter Beibehaltung der Vergleichbarkeit – sind sowohl bezüglich des Speicherbedarfs des Einzelmusters als auch bezüglich der Effizienz der operationalen Unterstützung für den praktischen Einsatz unverzichtbar.

In diesem Kapitel stellen wir mit der *Bitletrepräsentation* einen neuen Lösungsansatz vor, der einen ersten Schritt darstellt, hin zu einer kompakten Repräsentation von Wiederholungsmustern und effizienter Realisierung von Vergleichsoperationen zwischen diesen – auch bei langen Periodenlängen und großer Anzahl. Hierzu wird in Abschnitt 6.1 zunächst die Ausgangssituation der operationalen Unterstützung von musterbasierten Repräsentationsformen im Detail vorgestellt. In Abschnitt 6.2 wird die allgemeine Theorie der Bitletrepräsentation eingeführt und in Abschnitt 6.3 Möglichkeiten zur effzienten Realisierung der Bitlet-Transformation vorgestellt. Die auf der Bitletrepräsentation basierende Realisierung der operationalen Unterstützung wird in Abschnitt 6.4 eingeführt. Im Anschluss werden in Abschnitt 6.5 die Ergebnisse zur Bitletrepräsentationen im Vergleich zu alternativen Repräsentationsformen diskutiert. Abschließend werden in Abschnitt 6.6 ein Ausblick auf zukünftige Arbeiten gegeben und in Abschnitt 6.7 die wesentlicher Ergebnisse und Erkenntnisse zusammengefasst.

6.1 Ausgangssituation der operationalen Unterstützung mit Wiederholungsmustern

Die Anforderungen an eine operationale Unterstützung von periodischen Spezifikationen wurden bereits in Kapitel 2 vorgestellt. Es zeigt sich, dass neben Ableitungs- und Überprüfungsoperationen den generischen Vergleichsoperationen eine Schlüsselrolle in der operationalen Unterstützung zukommt. Vergleichsoperationen dienen der Bestimmung von gemeinsamen oder verschiedenen Werten periodischer Spezifikationen mit gleichen oder verschiedenen Periodizitätsschema. Diese Operationen werden benötigt, um Aussagen zu Konflikten in entsprechenden Anwendungen treffen zu können (siehe Abschnitt 2.1.3). Die spezifischen Eigenschaften sowie die konkrete Realisierung solcher Vergleichsoperationen werden in den folgenden Abschnitten vorgestellt bzw. vertieft.

6.1.1 Semantik von Vergleichsoperationen

Vergleichsoperationen zwischen periodischen Spezifikationen können in ihrer formalen Semantik als Mengenoperationen (z.B. $\cap, \cup$) zwischen den durch die beiden Spezifikationen repräsentierten (theoretisch unendlichen) Wertemengen verstanden werden. Solche Grundoperationen erlauben es, beliebige mengentheoretische Zusammenhänge bzw. Mengenbeziehungen zu spezifizieren, die beim Vergleich der beiden Spezifikation überprüft werden sollen. Somit können in der konkreten Anwendung durch Überprüfung der spezifizierten Mengenbeziehungen potentielle oder konkrete Konflikte zwischen periodischen Spezifikationen erkannt werden. Die konkrete Umsetzung einer solchen Vergleichsoperation in Mengenoperationen sowie die Interpretation der "Treffermenge" (z.B. als "positiv" oder "negativ") sind natürlich anwendungsabhängig.

Unabhängig von der konkreten Semantik sind für Vergleichsoperationen zwischen periodischen Spezifikationen zwei Aspekte kennzeichnend. Zum einen das *Auswertungsergebnis* und zum anderen der *Auswertungszeitraum* von Vergleichsoperationen (siehe Abschnitt 2.1.3).

- *Auswertungsergebnis:* Bei periodischen Spezifikation wird eine *quantitative Vergleichsauswertung* benötigt. Diese erlaubt es nicht nur, periodische Spezifikation bzw. deren Wertemengen auf die spezifizierten Mengenbeziehungen hin zu überprüfen, sondern es können im Fall, dass diese nicht gelten, auch die konkreten verstoßenden Werte identifiziert werden. Die Überprüfung der Mengenbeziehung beruht hierbei auf dem Test der leeren Ergebnismenge. Das heißt, das Ergebnis eines Vergleichs bei quantitativer Auswertung ist entweder die leere Menge oder die Menge der nicht erfüllenden Werte.
- *Auswertungszeitraum:* Semantisch entsprechen Vergleiche zwischen periodischen Spezifikationen dem Vergleich zweier (möglicherweise unendlich) großer Wertemengen. Konflikte zwischen periodischen Spezifikationen können daher erst "sehr spät" in deren Verlauf auftreten, die in der Praxis dann meist nicht mehr relevant

sind. Im Gegensatz zu einer solchen *unbeschränkten Auswertung* ist es häufig ausreichend und sinnvoller, den Auswertungszeitraum eines Vergleichs einzuschränken. Grundlage für eine solche *eingeschränkte Auswertung* bildet ein *Ereignishorizont,* der den Gültigkeitszeitraum einer Konfliktaussage anwendungsabhängig festlegt. Der Ereignishorizont kann konstant sein oder aber von Vergleich zu Vergleich variieren.

Die Betrachtungen zu möglichen Realisierungen von Vergleichsoperationen periodischer Spezifikationen ergab, dass lediglich musterbasierte Repräsentationsformen den Anforderungen an Vergleichsoperationen begegnen können (siehe Abschnitt 2.3). Bei dieser Repräsentationsform wird die (theoretisch unendliche) Wertemenge periodischer Spezifikation in Form eines Wiederholungsmuster repräsentiert. Auf Basis entsprechender Mengenoperationen zwischen Wiederholungsmustern können so die spezifizierten Vergleiche durchgeführt bzw. die mengentheoretischen Zusammenhänge überprüft werden. Allerdings können für Wiederholungsmuster verschiedene Repräsentationsformen mit unterschiedlichen Auswirkungen auf die Realisierung der Vergleichsoperationen unterschieden werden. In den folgenden Abschnitten werden diese Zusammenhänge näher betrachtet.

6.1.2 Varianten der linearen Musterrepräsentation

Durch eine musterbasierte Repräsentation wird das Wiederholungsmuster einer periodischen Spezifikation abgebildet. Für jede periodische Spezifikation kann ein solches Wiederholungsmuster angegeben werden, dass die (theoretisch unendliche) Wertemenge der Spezifikation in endlicher Form repräsentiert.[1] Ein Wiederholungsmuster stellt den kleinsten Zeitraum – die Periode – einer periodischen Spezifikation dar, nach dem sich die möglichen Abstände zwischen Werten sowie deren Reihenfolgen wiederholen. Ein Wiederholungsmuster "verläuft" über einen (diskreten) Wertebereich, wobei die Positionen des Musters mit den Werten des Wertebereichs korrespondieren. Die konkreten Belegungen der Positionen, d.h. die Stützstellen des Musters, treffen Aussagen, ob der zur Position korrespondierende Wert in der Spezifikation enthalten ist (oder nicht). Die Länge des Musters, d.h. die Periodenlänge entspricht der kleinsten Periode der Spezifikation.[2]

Wie in Abschnitt 2.3.3 gezeigt, stellen lineare Musterrepräsentationen, die Repräsentationsform dar, die eine operationale Unterstützung von Wiederholungsmustern grundsätzlich erlaubt. Hierbei unterscheiden wir zwei Varianten der lineare Musterrepräsentation: *dichtbesetzte und dünnbesetzte.*

[1] Beispielsweise haben wir in Kapitel 5 entsprechende Algorithmen zur Erzeugung von Wiederholungsmustern vorgestellt, die dies für im zyklenorientierten Modell beschriebene bzw. durch HSP-Bäume repräsentierte periodische Spezifikationen ermöglicht.

[2] Im Folgenden werden wir den Aspekt der unterschiedlichen Instanziierungspunkte vernachlässigen. Wie Kapitel 2 bzw. Kapitel 5 (Abschnitt 5.3.3) zeigen, erlaubt es eine entsprechende Rotation der Muster, diesem Problem zu begegnen.

- Bei einer *dichbesetzten* Musterrepräsentation wird jede Stützstelle des Musters (Musterposition) unabhängig von seiner konkreten Belegung repräsentiert. Die Bitvektordarstellung einer periodischen Spezifikation (die wir in den vorherigen Abschnitten bereits häufiger verwendet haben) ist Beispiel für eine solche dichtbesetzte Repräsentationsvariante. So kann etwa das Wiederholungsmuster einer Therapie mit einer Zykluslänge von 28 Tagen (Periodenlänge = 28) und Gaben von Medikament A vom 1. Tag bis zum 14. Tag durch den folgenden Bitvektor repräsentiert werden. Die Einsen des Vektors stellen die Gabe von Medikament A an dem mit der Position im Vektor korrespondierenden Tag dar.

$$(1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

- Bei einer *dünnbesetzten* Repräsentation werden nur die Stützstelle des Musters (Musterposition) repräsentiert, die eine bestimmte Belegung aufweisen. Zusätzlich wird die Periodenlänge des Musters explizit hinterlegt. Für das vorherige Beispiel ergibt sich etwa die folgende dünnbesetzte Repräsentationform (siehe unten). Der erste Wert des Tupels gibt die Periodenlänge an. Der zweite Wert gibt die Menge an Stützstellen bzw. deren Position im Muster an, die eine Belegung $\neq 0$ aufweisen und hier die Gaben von Medikament A darstellen. Entsprechende Belegungen werden auch als die *Nicht-Null-Belegungen* eines Musters bezeichnet.

$$(28, \{1,2,3,...,14\})$$

Es hängt von den spezifischen Eigenschaften eines Wiederholungsmusters ab, welche der beiden Repräsentationsformen sich in Bezug auf den Umfang der Repräsentation als vorteilhaft erweist. In Mustern mit wenigen Nicht-Null-Belegungen und langen Periodenlänge kann sich etwa die dünnbesetzte Repräsentation gegenüber der dichtbesetzten als vorteilhafter erweisen. Im Folgenden konzentrieren wir uns lediglich auf die Eigenschaften der beiden Repräsentationsformen im Zusammenhang mit der operationalen Realisierung, genauer gesagt, auf die Eigenschaften bei der Realisierung von Vergleichsoperationen.

6.1.3 Musterbasierte Realisierung von Vergleichsoperationen

Bei der Realisierung von Vergleichsoperationen auf Basis von Wiederholungsmustern gilt es zwei Fälle zu unterscheiden:[3] Muster mit identischer und mit unterschiedlicher Periodenlänge.

Bei identischer Periodenlänge können die beiden Wiederholungsmuster unmittelbar miteinander verglichen werden. Das heißt, die jeweiligen Stützstellen des Wiederholungsmusters werden gemäß der Semantik des Vergleichs direkt miteinander verglichen. Für

[3] Wir konzentrieren uns an dieser Stelle zunächst auf die unbeschränkte Auswertung von Vergleichsoperationen. Die Eigenschaften der eingeschränkten Auswertung von Wiederholungsmustern im Zeitbereich werden später in Abschnitt 6.5.2 näher betrachtet.

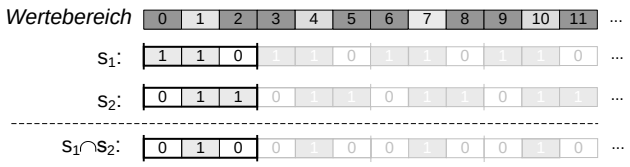


Abbildung 6.1: Schnittoperation von Wiederholungsmustern mit gleicher Periodenlänge

die quantitative Vergleichsauswertung kann hierbei einfach auf die leere Menge getestet werden bzw. ein entsprechendes Ergebnismuster angegeben werden, dass die verstoßenden Werte repräsentiert (siehe Abschnitt 2.1.3). In Abbildung 6.1 ist dies für eine Bitvektordarstellung, d.h. für eine dünnbesetzte Repräsentationsvariante von Wiederholungsmustern mit Periodenlänge 3 dargestellt. Als Vergleichsoperation wurde die Schnittoperation zwischen zwei Mustern verwendet. Bei einer dünnbesetzten Repräsentationsvariante entspricht das Beispiel in Abbildung 6.1 den beiden Tupeln $s_1 = (3, \{1, 2\})$ und $s_2 = (3, \{2, 3\})$. Die Realisierung von Vergleichsoperationen, wie etwa der Schnittoperation, wird bei dieser Repräsentationsvariante durch Mengenoperationen zwischen den jeweiligen Mengen an Stützstellen abgebildet. In dem Beispiel führt dies zu dem Ergebnistupel $s_{s_1 \cap s_2} = (3, \{2\})$ mit $s_1 \cap s_2 = \{1, 2\} \cap \{2, 3\} = \{2\}$ sowie der Periodenlänge $=$ 3, die unverändert übernommen wird.

Für Wiederholungsmuster mit unterschiedlicher Periodenlänge wird, unabhängig von der konkreten Repräsentation, stets das gleiche Vorgehen verwendet. Es wird die *kleinste gemeinsame Periodenlänge* (least common period - LCP) der beiden Wiederholungsmuster gebildet. Das LCP entspricht dem kleinsten gemeinsamen Vielfachen der Periodenlängen der Muster und ist der Zeitbereich, den beide Muster nach ganzzahliger Anzahl von Wiederholungen durchlaufen haben. Die Muster werden innerhalb des LCP erweitert bzw. explizit aufgeführt. Ziel ist es, die unterschiedlichen Periodenlängen so zu verlängern, dass erneut identische Periodenlängen entstehen. Nach der Verlängerung kann die gleiche Realisierung wie im ersten trivialen Fall verwendet werden. Eine solche Verlängerung bzw. Schnittoperation bei einer dünnbesetzten Repräsentationsvariante zeigt Abbildung 6.2 mit den zwei Mustern mit Periodenlänge 3 und Periodenlänge 2 sowie dem Ergebnismuster der Schnittoperation mit Periodenlänge $6 = LCP(2, 3)$. In der dünnbesetzten Repräsentationvariante findet ein analoges Vorgehen statt. Da heißt, die beiden Tupel werden innerhalb des LCP verlängert und die Stützstellen entsprechen aufgeführt. Für das Tupel $s_2 = (3, \{2, 3\})$ führt die Verlängerung zu den Tupel $s'_2 = (6, \{2, 3, 5, 6\})$ bzw. für das Tupel $s_3 = (2, \{1\})$ zu den Tupel $s'_3 = (6, \{1, 3, 5\})$ und im Ergebnis zu den Tupel $s_{s'_2 \cap s'_3} = (6, \{3, 5\})$.

Beide Repräsentationsvarianten ermöglichen es, die geforderte operationale Unterstützung grundsätzlich zu gewährleisten. Allerdings erweist sich die bei unterschiedlichen Periodenlänge notwendige Verlängerung der Wiederholungsmuster als problematisch. Genauer gesagt führt diese dazu, dass sowohl die Anzahl der zur Darstellung notwen-

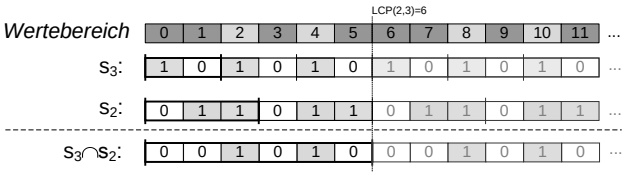


Abbildung 6.2: Schnittoperation von Wiederholungsmustern mit unterschiedlicher Periodenlänge

digen Stützstellen als auch die Anzahl an (paarweisen) Einzelvergleichen einer Vergleichsoperation ansteigt. Für die beiden Repräsentationsvarianten zeigt Tabelle 6.1 die resultierende Anzahl an notwendigen Stützstellen bzw. paarweisen Einzelvergleichen (im Worst-Case Fall, d.h. $LCP(n, m) = n \cdot m$). Die Parameter β_n und β_m stellen den *Besetzungsgrad* eines Musters dar, d.h. das normierte Verhältnis der Stützstellen zur Periodenlänge mit $0 < \beta \leq 1$, die nur bei der dünnbesetzten Repräsentationvariante relevant sind.

Repräsentationsvariante	#Einzelvergleiche
dichtbesetzt	$n \cdot m$
dünnbesetzt	$(\beta_n + \beta_m) \cdot (n \cdot m)$

Tabelle 6.1: Einzelvergleiche der beiden Repräsentationsvarianten

In einer dichtbesetzten Repräsentationvariante werden $n \cdot m$ Einzelvergleiche im Worst-Case benötigt, um eine Vergleichsoperation, wie etwa eine Schnittoperation, zwischen zwei Mustern mit unterschiedlicher Periodenlänge durchzuführen. Hierbei spielt die konkrete Belegung der einzelnen Stützstellen keine Rolle. Im Gegensatz dazu muss bei einer dünnbesetzten Repräsentationsform lediglich die Anzahl der Nicht-Null-Belegungen $(\beta_n + \beta_m)$ der Muster innerhalb des LCPs verglichen werden. Auffällig in beiden Repräsentationvarianten ist jedoch der Faktor $n \cdot m$. Dieser Faktor bzw. die damit verbundene Erhöhung von Einzelvergleichen ist es, der bei diesen Repräsentationsvarianten bzw. deren Realisierung von Vergleichsoperationen das Hauptproblem darstellt.

Die Problematik der Erhöhung und der damit verbundene Aufwand zeigt sich bereits bei einfachen praktischen Beispielen. Das Muster für *"Jeden 1.Tag im Monat"* weist eine Periodenlänge von 400 Jahren (= 146.097 Tage) auf. Dies entspricht in der dünnbesetzten Repräsentationsform 4800 Einzelwerten bzw. in der dichtbesetzten Repräsentationsform einem Bitvektor der Länge 146.097 Bits ($\approx$ 4500 Einzelwerten bei 32bit Integerpräzision). Ein Chemotherapie mit Periodenlänge = 28 Tage und Medikamentengabe vom 1. bis zum 14.Tag führt zu einem Bitvektor der Länge 28 Bits bzw. zu einer dünnbesetzten Repräsentation mit 14 Einzelwerten. Der Vergleich der Chemotherapie mit dem ersten Beispiel führt zu der Periodenlänge $lcp(28, 146.097) = 584.388$ Tagen. Dies bedeutet dass, die erste Angabe auf $4 \cdot 4800 = 19.200$ Werte ($\frac{584.388}{146.097} = 4$) und die zweite Anga-

be auf $14 \cdot 20.871 = 290.934$ Werte ($\frac{584.388}{28} = 20.871$) erweitert wird. Dies führt dazu, dass etwa in der dünnbesetzten Repräsentationsform 19.200 Werte der ersten Angaben mit 290.934 Werten der Chemotherapie verglichen werden müssen. In einer dichtbesetzten Repräsentationsform gilt es zwei Bitvektoren der Länge 584.388, was der Anzahl an bitweisen Einzelvergleichen entspricht, zu vergleichen. Die Situation verschärft sich deutlich, falls die Chemotherapie eine Länge von 29 Tagen aufweist, da hier das LCP den Wert $lcp(29, 146.097) = 4.236.813$ aufweist.

Das obige Beispiel zeigt, dass der resultierende Aufwand bei dieser Repräsentationsform von Wiederholungsmuster und Realisierung von Vergleichsoperationen für einen effizienten praktischen Einsatz weniger geeignet ist. Besonders deutlich wird dies, wenn man bedenkt, dass jede zusätzliche Ausnahme (meist) zwangsläufig zu einer entsprechenden Vergrößerung der Periodenlänge führt und somit auch zu einer Erhöhung des Umfangs der Musterrepräsentation und der Anzahl an Einzeloperationen bei einer Vergleichsoperation.

Das Kernproblem an dieser Stelle bildet die notwendige Verlängerung des Wiederholungsmusters und betrifft in gleicher Weise dichtbesetzte und dünnbesetzte Repräsentationsvarianten. Während sich bei Ersteren etwa die Länge des Bitvektors entsprechend vergrößert, erhöht sich bei Zweiteren die Anzahl der Nicht-Null-Belegungen. Bei beiden führt dies jedoch zu den oben beschriebenen starken Anstieg von Einzelvergleichen.

Aus dieser Situation ergibt sich die Fragestellung, ob ein solch starker Anstieg des Aufwands vermieden werden kann bzw. alternative Repräsentationformen existieren, die davon weniger betroffen sind. Im Folgenden stellen wir mit der *Bitletrepräsentation* eine solche Repräsentationsform vor.

6.2 Allgemeine Theorie der Bitletrepräsentation

Im folgenden Abschnitt wird die allgemeine Theorie der Bitlet-Repräsentation vorgestellt. Hierbei stehen zunächst die theoretischen Grundlagen und Eigenschaften im Vordergrund. Der Aspekt der effizienten Realisierbarkeit wird erst anschließend in Abschnitt 6.3 im Detail betrachtet. Wir beginnen mit der Grundidee der Bitletrepräsentation.

6.2.1 Grundidee der Bitletrepräsentation

Die generelle Idee der Bitlet-Repräsentation kann mit der Fourier-Transformation verglichen werden. Vereinfacht ausgedrückt, bei einer Fourier-Transformation wird eine Funktion im Zeitbereich in seine "Bestandteile" im Frequenzbereich "zerlegt" [13, 47, 44]. Durch diese Bestandteile wird die Funktion im Frequenzbereich repräsentiert. Allgemeiner ausgedrückt, bei jeder Transformation, wie die Fourier-Transformation oder andere wie Kosinus- und Wavelet-Transformation, wird eine Funktion im Zeitbereich in einen Wertebereich bestehend aus einfachen Basisfunktionen zerlegt. Durch Rekombinieren der entsprechenden Basisfunktionen kann die ursprüngliche Funktion jedoch wiederhergestellt bzw. in den Zeitbereich zurückgeführt werden. Bei der Fourier-Transformation werden (einfache) sinusförmige Basisfunktionen verwendet bzw. bei der Wavelet-Transformation skalierte und verschobene Kopien einer (endlichen) oszillierenden Wellenform [13, 33, 56]. Die weite Verbreitung entsprechender Transformationen ergibt sich aufgrund der Eigenschaft, dass aufwändige Operationen zwischen Funktionen im Zeitbereich sich im transformierten Wertebereich häufig als deutlich einfacher und effizienter realisierbar erweisen (z.B. Faltungsoperation im Frequenzbereich).

Für die *Bitletrepräsentation* bzw. bei der *Bitlet-Transformation* wird analog wie oben skizziert verfahren. Das heißt, wir transformieren ein Wiederholungsmuster mit langer Periodenlänge und komplexen periodischen Verhalten in einen Wertebereich bestehend aus Basismustern kurzer Periodenlänge und einfachem periodischen Verhalten. Diese Basiswiederholungsmuster können als die Bestandteile des komplexen Wiederholungsmusters verstanden werden. Wir bezeichnen die Basiswiederholungsmuster als *Bitlets* (aufgrund ihrer Ähnlichkeit mit Wavelets) und den korrespondierenden Wertebereich als *Bitletbereich*. Unter der *Bitletrepräsentation* wird die Abbildung des Wiederholungsmusters im Bitletbereich verstanden. Es zeigt sich, dass die Realisierung von Vergleichsoperationen sich im Bitletbereich als deutlich effizienter gegenüber Realisierungen im Zeitbereich erweist. Das heißt, das Problem der Verlängerung von Mustern unterschiedlicher Länge und dem damit verbundenen hohen Aufwand besteht bei der entsprechenden Realisierung im Bitletbereich nicht bzw. nicht in dieser Stärke. Die generelle Idee der Bitlet-Transformation und seine Beziehung zur Fourier-Transformation zeigt Abbildung 6.3.

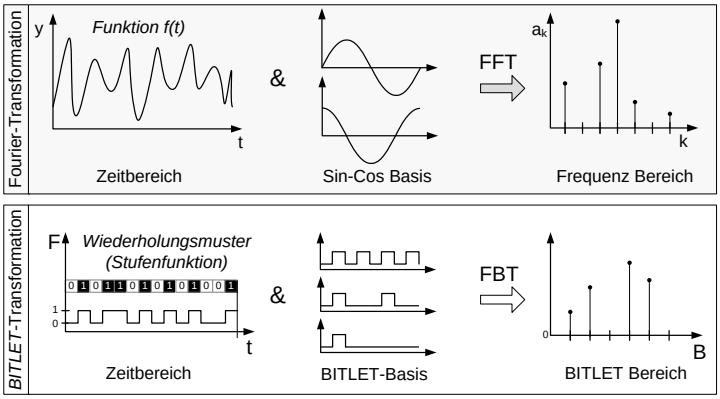


Abbildung 6.3: Schematische Darstellung des Zusammenhangs von Bitlet- und Fourier-Transformation

6.2.2 Grundlegende Begriffe und Eigenschaften

Die folgenden Begriffe und Eigenschaften entsprechen Standardbegriffen im Bereich der linearen Algebra und werden im Folgenden lediglich zum besseren Verständnis der Bitletrepräsentation kurz vorgestellt. Auf eine umfassende Einführung wird an dieser Stelle verzichtet und auf die entsprechende Standardliteratur verwiesen (z.B. [40]).

6.2.2.1 Vektorräume, lineare Unabhängigkeit und Basen

Ein **Vektorraum** V über einem Körper $(K, +, \cdot)$ oder kurz K-Vektorraum ist eine algebraische Struktur, deren Elemente sich addieren $(+)$ und skalarmultiplizieren $(\cdot)$ lassen, wobei die Summe der Elemente und das Vielfache eines Elements wieder Elemente der Struktur sind. Die Elemente so eines Vektorraumes werden als Vektoren bezeichnet.

Eine Menge von Vektoren $\{v_1, ..., v_m\} \subseteq V$ eines Vektorraums wird als **linear unabhängig** bezeichnet, falls für $k_1 \cdot v_1 + \cdots + k_m \cdot v_m = 0$ $(k_i \in K)$ folgt, dass $k_1 = k_2 = \cdots = k_m = 0$ ist. Das heißt, die Menge von Vektoren ist linear unabhängig, wenn sich keiner der Vektoren als Linearkombination der anderen Vektoren darstellen lässt. Nicht linear unabhängige Mengen heißen **linear abhängig** und es gibt eine Darstellung $k_1 \cdot v_1 + \ldots + k_m \cdot v_m = 0$ in der mindestens ein k_i nicht Null ist.

Eine Menge von linear unabhängigen Vektoren $B = \{v_1, \ldots, v_n\} \subseteq V$ wird als **Basis** des Vektorraums V bezeichnet, falls gilt, dass $B \cup \{w\}$ für jeden Vektor $w \in V$ linear abhängig ist. Das heißt, die Basis eines Vektorraums erlaubt es, jeden Vektor w des Vektorraums durch genau eine Linearkombination der Basisvektoren $w = k_1 \cdot v_1 + \ldots + k_n \cdot v_n$ darzustellen. Die Basis eines Vektorraumes ist nicht eindeutig bestimmt, d.h. jeder Vektorraum kann verschiedene Basen aufweisen.

Die Linearfaktoren $(k_1, \ldots, k_n)$ der Linearkombination von Basisvektoren heißen **Koordinaten** des Vektors bezüglich der Basis und sind Elemente des zugrundeliegenden Körpers $k \in K$. Wird mit Vektoren gerechnet, so wird stets mit deren Koordinaten gerechnet.

Die Anzahl m der Basisvektoren eines Vektorraums V wird **Dimension** genannt. Sie ist unabhängig von der Wahl der Basis und kann auch unendlich sein.

6.2.2.2 Lineare Abbildung

Eine lineare Abbildung ist eine Funktion f, die einen Vektor des Vektorraums V in einem anderen (beliebigen) Vektorraum abbildet. Das heißt, die Abbildung $f : V \mapsto W$ mit $f(v) = w$ bestimmt für den Vektor $v \in V$ dessen Vektor bzw. Bildvektor w im Vektorraum W. Der Abbildung f bzw. dem "Transformieren" eines Vektors in Abhängigkeit von der ausgewählten Basis kommt im Zusammenhang mit Bitletrepräsentation (bzw. auch bei allen anderen Transformationen) eine besondere Rolle zu. Aufgrund dessen ist die Abbildung bzw. ihre Eigenschaften im Folgenden genauer definiert:

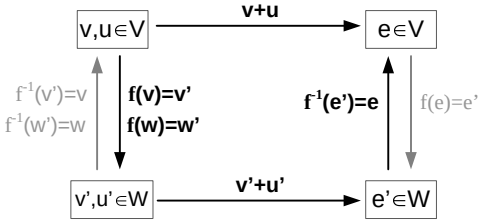
Definition 6.1 (Lineare Abbildung): *Seien V und W K-Vektorräume. Eine Abbildung $f : V \mapsto W$ von V in W wird als lineare Abbildung bezeichnet, falls für beliebige $v, u \in V$ und $k \in K$ gilt:*

$$f(v + u) = f(v) + f(u)$$

$$f(v - u) = f(v) - f(u)$$

$$f(k \cdot v) = k \cdot f(v)$$

Das Besondere der linearen Abbildung ist, dass sie sich gegenüber den Operationen im Vektorraum, d.h. die Addition $(+)$, die Subtraktion $(-)$ und die skalare Multiplikation $(\cdot)$, als *verträglich* verhält. Es spielt keine Rolle, ob wir etwa zunächst zwei Vektoren des Vektorraums V addieren $(v+u)$ und anschließend in den Raum W "überführen" bzw. abbilden $(f(v + u))$ oder die zwei Elemente einzeln in den Raum W abbilden $(f(v), f(u))$ und anschließend addieren $(f(v) + f(u))$. Das Ergebnis der Addition bleibt jeweils identisch. Das Gleiche gilt für die skalare Multiplikation und Subtraktion. Die Eigenschaft gilt jedoch nicht für eine (vorstellbare) Multiplikation zwischen Vektoren, d.h. $f(v \cdot u) \neq f(v) \cdot f(u)$. Zur Übersicht ist im folgenden Diagramm der Zusammenhang der linearen Abbildung und der Addition dargestellt:



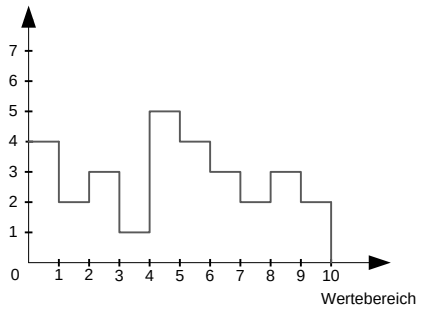


Abbildung 6.4: Beispiel einer Funktion mit diskretem Wertebereich

Sind V und W endlichdimensionale Vektorräume, so kann jede lineare Abbildung durch eine Matrix realisiert bzw. dargestellt werden. Die Matrix wird als Abbildungsmatrix oder Darstellungsmatrix bezeichnet. Für den Vektor $v \in V$ erhält man die Koordinaten des Bildvektors $f(v) = w$, d.h. den Koordinatenvektor im Raum W, durch Multiplikation der Darstellungsmatrix M mit dem Vektor v, d.h. $f(v) = M \cdot v = w$. Bilden die Spaltenvektoren der Darstellungsmatrix M eine Basis von W, so ist die lineare Abbildung bijektiv, d.h. $f^{-1}(w) = M^{-1} \cdot w = v$.

6.2.2.3 Vektorräume und Funktionen

Den Ausgangspunkt bei Transformationen, wie Fourier-Transformation oder auch der Bitletrepräsentation, bildet die zentrale Überlegung, dass eine (beliebige) Funktion mit Länge n als ein Koordinatenvektor eines n-dimensionalen K-Vektorraums verstanden werden kann.[4] Genauer gesagt die Werte im Wertebereich der Funktion werden als Koordinaten im Körper K verstanden. Die Basisvektoren des Raums "ersetzen" den ursprünglichen Definitionsbereich der Funktion, d.h. jeder Wert des Definitionsbereichs wird jeweils durch einen einzelnen Basisvektor dargestellt. Abbildung 6.4 zeigt ein Beispiel einer (Stufen-) Funktion mit Länge $n = 11$ und diskretem Wertebereich. Die Betrachtung der Stufenfunktion bzw. ihrer Werte als eine Koordinate in einem 11-dimensionalen Raum V^{11} führt zu den Vektor $v = (4, 2, 3, 1, 5, 4, 3, 2, 3, 2) \in V^{11}$.

Wie im vorherigen Abschnitt gezeigt, ist es mittels einer (linearen) Abbildung möglich, den Vektor v in einen anderen beliebigen Vektorraum W (gleicher oder verschiedener Dimension) zu überführen, d.h. $f(v) = w$ mit $w \in W$. Der "Trick" an dieser Stelle ist es, dass der neue Raum genau die Eigenschaften aufweist, die im ursprünglichen Raum V etwa für eine effiziente Realisierung von Operationen nicht zur Verfügung stehen.

[4] Auch wenn wir im Folgenden stets von diskreten und endlichen Funktionen, Basen und Vektorräumen ausgehen, so sind die vorgestellten Grundlagen allgemeingültig und nicht auf diskrete Elemente beschränkt. Allerdings, wenn eine Funktion nicht nur "in Gedanken" in eine Vektordarstellung übertragen wird, sondern konkret, so wird eine Diskretisierung mit Blick auf den numerischen Aufwand unumgänglich. Dies gilt sowohl für die Funktion selber als auch für die Wahl der Basen und den zugrundegelegten Vektorraum.

Bei einer Fourier-Transformation wird die Funktion bzw. deren Koordinatenvektor in einen Vektorraum überführt, der sinusförmige Funktionen als Basis aufweist. Aufgrund der Eigenschaften des neue Raums bzw. des Fourier Bereichs ist beispielsweise eine effiziente Realisierung der Faltungsoperation möglich, die sich im ursprünglichen Raum (Zeitbereich) als sehr aufwändig erweist [13, 47].

Die Eigenschaften eines Vektorraums und der Realisierung von Operationen wird unmittelbar durch die ausgewählte Basis bestimmt. Die Wahl der Basis des Vektorraums ist somit das entscheidende Kriterium bei entsprechenden Transformationen, die es nachträglich erlaubt, etwa Operationen effizienter realisieren zu können. Wie wir im Folgenden zeigen werden, ist dies auch bei der Bitletrepräsentation der Ansatzpunkt, der eine effizientere Realisierung von Vergleichsoperationen ermöglicht.

6.2.3 Wiederholungsmuster und Vektorräume

Im folgenden Abschnitt zeigen wir, dass Wiederholungsmuster als Koordinatenvektoren von Vektorräumen verstanden werden können. Dies ermöglicht es, die eigentlichen Ursachen des Hauptproblems im Umgang mit Muster unterschiedlicher Periodenlänge genauer mathematisch zu charakterisieren, aber auch hierfür eine entsprechende Lösungsidee zu entwickeln.

6.2.3.1 Vom Wiederholungsmuster zum Koordinatenvektor

Aus mathematischer Sicht kann ein Wiederholungsmuster als eine Funktion verstanden werden, in der die Positionen des Musters den Definitionsbereich bilden und die konkreten Belegungen des Musters den Wertebereich. Das heißt, durch ein Muster wird jedem Wert im Definitionsbereich sein Wert im Wertebereich zugeordnet. Abbildung 6.5 zeigt ein Beispiel für die Betrachtung eines Musters als Stufenfunktion.

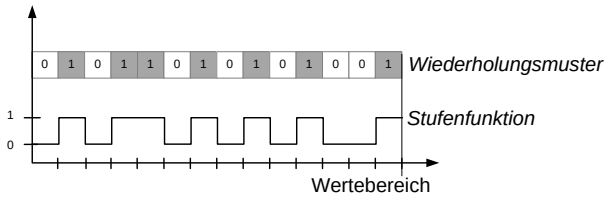


Abbildung 6.5: Wiederholungsmuster als Stufenfunktion

Analog wie bei Funktionen, kann auch ein Wiederholungsmuster mit Periode m als ein Koordinatenvektor eines m-dimensionalen Vektorraums V^m verstanden werden. Das Wiederholungsmuster (oder dessen Stufenfunktion) aus Abbildung 6.5 kann als der Koordinatenvektor $v = (0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1)$ des Raums V^{14} betrachtet werden. Somit ist es ebenfalls möglich, durch lineare Abbildung den Koordinatenvektor eines

Musters in einen anderen beliebigen Vektorraum zu überführen, der etwa zur Realisierung von Operationen geeignetere Eigenschaften aufweist. Im Folgenden nehmen wir stets an, dass es sich bei den betrachteten Vektorräumen um $\mathbb{Z}$-Vektorräume handelt, d.h. Vektorräume über den Zahlenbereich der ganzen Zahlen.

6.2.3.2 Raumeinbettung und Muster unterschiedlicher Periodenlänge

Die Betrachtungsweise eines Musters als Koordinatenvektor führt zu dem Problem, dass Wiederholungsmuster unterschiedlicher Länge zu Koordinatenvektoren von Räumen unterschiedlicher Dimension führen. Ausgehend von zwei Mustern mit Länge m und n ergeben sich die beiden resultierenden Vektorräume V^m und V^n. Um mit den Vektoren solcher Räume irgendwie "rechnen" zu können, ist es unumgänglich, die resultierenden Vektorräume unterschiedlicher Dimensionen zu vereinheitlichen bzw. in einem gemeinsamen Vektorraum einzubetten. Im Worst-Case sind die Dimensionen der beiden Räume teilerfremd und deren einziger gemeinsamer Vektorraum ist der Raum $V^{m \cdot n}$. Die Vereinheitlichung der beiden Vektorräume V^m und V^n in den gemeinsamen Vektorraum wird als Raumeinbettung bezeichnet.

Die Raumeinbettung von Vektorräumen bildet das korrespondierende Problem im Vektorraum, wie es für Wiederholungsmuster unterschiedlicher Länge im Zeitbereich besteht. In beiden Fällen müssen die unterschiedlichen Längen bei Mustern oder die Dimensionen der Räume vereinheitlicht werden, um entsprechende Operation ausführen zu können. Ausgangspunkt bildet ein Wiederholungsmuster r mit Periodenlänge m. Die Verlängerung des Musters r um den Faktor a führt zu dem Muster r' mit Periodenlänge $a \cdot m$. Der Faktor a entspricht etwa der Periodenlänge eines zweiten Musters mit $lcp(a, m) = a \cdot m$. Sei v der Vektor, der das Muster r in einem m-dimensionalen Vektorraum V^m repräsentiert. Bei der Raumeinbettung von V^m in den Raum $V^{a \cdot m}$ wird der Vektor v auf einen Vektor $v' \in V^{a \cdot m}$ überführt, der das Muster r' im größeren Raum repräsentiert. Das heißt, der m-dimensionale Vektor v wird in einen $m \cdot a$-dimensionalen Vektor v' überführt. Der Zusammenhang zwischen Raumeinbettung und der Verlänge-

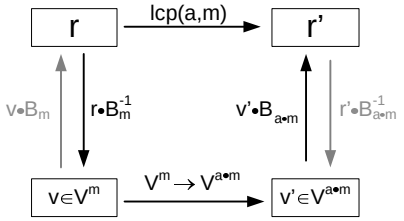


Abbildung 6.6: Zusammenhang von Raumeinbettung und Verlängerung bei Mustern unterschiedlicher Periodenlänge

rung von Wiederholungsmustern mit unterschiedlichen Perioden ist in Abbildung 6.6 zusammenfassend dargestellt. Die lineare Abbildung $r \cdot B_n^{-1}$ überführt ein Wiederholungsmuster in den Vektorraum V mit Basis B_n bzw. die Abbildung $v \cdot B_n$ einen Vektor v zurück auf das entsprechende Wiederholungsmuster.

Die Konsequenz aus dem Diagramm in Abbildung 6.6 ist, dass eine effiziente Lösung für Raumeinbettungen es ermöglichen würde, dem Problem der Verlängerung von Mustern zu begegnen. Anstelle einer aufwändigen Verlängerung können die Muster in einen Vektorraum überführt werden, der anschließend in den gemeinsamen Vektorraum eingebettet wird. In dem gemeinsamen Vektorraum werden dann die entsprechenden Operationen (z.B. Vergleichsoperationen) ausgeführt. Mit der Rücktransformation könnte (wenn erwünscht) das Ergebnis im Vektorraum in das korrespondierende Wiederholungsmuster im Zeitbereich überführt werden.

Voraussetzung damit sich dieser "Umweg" über den Vektorraum als effizienter gegenüber der Realisierung mit Mustern erweist, ist jedoch:

1. Ein effizientes Verfahren zur Einbettung unterschiedlich dimensionaler Räume in einen gemeinsamen Vektorraum.
2. Eine effiziente Hin- und Rücktransformation eines Wiederholungsmusters in (aus) den Vektorraum.
3. Eine effiziente Realisierung von Operationen (z.B.Vergleichsoperationen) im Vektorraum.

Wir werden zunächst das Problem der Raumeinbettung näher betrachten und einen entsprechenden Lösungsansatz in den Abschnitten 6.2.4 und 6.2.5 vorstellen. Die effiziente Transformation in (aus) den Vektorraum wird Gegenstand der Betrachtungen von Abschnitt 6.3 sein. Die Realisierung von Operationen zeigt Abschnitt 6.4.

6.2.3.3 Problematik bei Raumeinbettungen

Die Raumeinbettung kann grundsätzlich nicht vermieden werden, wenn Vektoren unterschiedlicher Dimension vorliegen. Allerdings hängt es von den Eigenschaften eines Vektorraums ab, wie "teuer" bzw. aufwändig eine Raumeinbettung ist. Da die Eigenschaften eines Vektorraums durch seine Basis bestimmt wird, spielt die Auswahl der Basis eine entscheidende Rolle hinsichtlich des Aufwands bei einer Raumeinbettung.

Die Repräsentation der ersten Periode einer periodischen Spezifikation als Wiederholungsmuster entspricht der Verwendung der "Standardbasis" in einem korrespondierenden Vektorraum. In der Standardbasis S^m entspricht jeder Basisvektor einem Einheitsvektor, d.h. einem Vektor, der nur an einer Stelle eine Eins aufweist. Beispielsweise ergeben sich die folgenden Basen bzw. entsprechende Darstellungsmatrizen für die Vektorräume S^2 und S^4:

$$S_2 = \begin{pmatrix} s_0: & 1 & 0 \\ s_1: & 0 & 1 \end{pmatrix}, \quad S_4 = \begin{pmatrix} s_0: & 1 & 0 & 0 & 0 \\ s_1: & 0 & 1 & 0 & 0 \\ s_2: & 0 & 0 & 1 & 0 \\ s_3: & 0 & 0 & 0 & 1 \end{pmatrix}$$

Auf Grundlage der Darstellungsmatrix $S_m \in S^{m \times m}$ kann jedes Wiederholungsmuster mit Länge m mittels linearer Abbildung in den Vektorraum mit Standardbasis überführt werden.[5] Beispielsweise weist das Muster $r = (0,1) \in V^2$ die Koordinate $v = r \cdot S_2 = (0,1) \in S^2$ im Raum S^2 auf. Das Nicht-Null-Element im Vektor v zeigt, dass der Basisvektor s_1 zur Darstellung des Musters r ausreicht. Allerdings führt die Verwendung der Standardbasis zu aufwändigen Raumeinbettungen. So führt etwa die periodische Verlängerung des Musters $r = (0,1) \in S^2$ um den Faktor=2 ($a = 2$) zu dem Muster $r' = (0,1,0,1) \in S^{2 \cdot 2 = 4}$. Für den entsprechenden Koordinatenvektor von r' in dem größeren Vektorraum S^4 ergibt sich $v' = r' \cdot S_4^{-1} = (0,1,0,1) \in S^4$. Die Anzahl der Nicht-Null-Elemente im Koordinatenvektor haben sich somit ebenfalls um den Faktor a erhöht. Das heißt, zur Darstellung des Musters r wird im Raum S^2 nur die einzelne Basis s_1 benötigt. Für das verlängerte Muster r' werden im größeren Vektorraum die beiden Basisvektoren s_1 und s_3 benötigt. Die Erhöhung der Nicht-Null-Elemente im Koordinatenvektor bei Verwendung der Standardbasis ist in ihrer Eigenschaft identisch zu der Erhöhung der Nicht-Null-Belegungen bei Wiederholungsmustern.

Aus mathematischer Sicht ergibt sich als Grund für diese aufwändige Raumeinbettung die *Inkompatibilität der Basen* der Vektoräume V^m und $V^{a \cdot m}$. Durch die Inkompatibilität wird jede Koordinate eines Vektors $v \in V^m$ im Vektorraum $V^{a \cdot m}$ auf andere und in ihrer Anzahl vervielfachte Koordinaten bzw. Basisvektoren abgebildet. Im obigen Beispiel wird etwa die einzige Nicht-Null-Koordinate des Vektors v (Basisvektor s_1) im Raum S^2 bei der Raumeinbettung in die beiden Basisvektoren s_1 und s_3 von S^4 überführt ($s_1 \in S_2$ ist ein anderer Basisvektor als $s_1 \in S_4$). Um dem Problem der aufwändigen Einbettung von Räumen zu begegnen, werden zueinander *kompatible Basen* benötigt. Entsprechende Basen bzw. deren Erzeugung werden im nachfolgenden Abschnitt vorgestellt.

6.2.4 Induktives Basensystem

Die Inkompatibilität von Basen führt bei der Raumeinbettung unterschiedlich dimensionaler Räume zu einer (starken) Erhöhung der Nicht-Null-Elemente in ihren Koordinatenvektoren. Um dies zu vermeiden gilt es, zueinander *kompatible Basen* zu verwenden.

[5] An dieser Stelle, aber auch im weiteren Verlauf der Arbeit, gehen wir aus Gründen der Lesbarkeit stets von einer transponierten Form der Darstellungsmatrix aus. Das heißt, in dieser Matrix bilden die Zeilenvektoren eine Basis für den Vektorraum. Dies ermöglicht es, dass wir Wiederholungsmuster bzw. deren Koordinatenvektor als Zeilenvektor darstellen können, das sich, gegenüber der sonst notwendigen Spaltenvektordarstellung, als besser lesbar erweist.

Die Idee hinter kompatiblen Basen ist, dass die Koordinate eines Basisvektor im Raums V^m nach der Raumeinbettung in den Raum $V^{a \cdot m}$ ebenfalls nur einer einzelnen Koordinate entspricht bzw. durch einen einzelnen Basisvektor dargestellt werden kann. Wenn dies für alle Basisvektoren des Raums V^m gilt, führt das dazu, dass die Anzahl der Nicht-Null-Elemente des Koordinatenvektors bei der Raumeinbettung gleich bleibt. Eine Erhöhung der Nicht-Null-Elemente im Koordinatenvektor würde nicht stattfinden und somit eine "effizientere" Raumeinbettung ermöglichen.

Eine einfache Möglichkeit kompatible Basen zu erzeugen ist es, als Basisvektoren im Raum $V^{a \cdot m}$ periodisch verlängerte Versionen (um den Faktor a) der Basisvektoren aus V^m zu verwenden. Das bedeutet, dass wenn ein Vektor v (Muster r) im Raum V^m durch den Basisvektor b_1 dargestellt wird, so wird sein (um Faktor a) verlängerter Vektor v' (verlängertes Muster r') im Raum $V^{a \cdot m}$ durch den Basisvektor b'_1 dargestellt, der dem periodischen verlängerten Basisvektor b_1 entspricht. Infolgedessen bleibt die Anzahl an Nicht-Null-Koordinaten eines Vektors gleich, unabhängig von der Dimension des Raums $V^{a \cdot m}$. Beispielsweise stellt der Basisvektor $b'_1 = (0, 1, 0, 1) \in B_{2 \cdot 2}$ für den Vektorraum $V^{2 \cdot 2}$ den periodisch verlängerten Basisvektor $b_1 = (0, 1) \in B_2$ des Raums V^2 dar. Das Wiederholungsmuster $r = (0, 1)$ wird im Raum V^2 durch die Koordinaten $v = (0, 1) \in V^2$ bzw. durch den einzelnen Basisvektor $b_1 = (0, 1) \in B_2$ dargestellt. Im Raum $V^{2 \cdot 2}$ existiert wiederum der zum Basisvektor b_1 kompatible Basisvektor $b_2 = (0, 1, 0, 1) \in B_{2 \cdot 2}$ (wir ignorieren die anderen Basisvektoren). Dies führt dazu, dass das periodisch verlängerte Muster $r' = (0, 1, 0, 1)$ im Raum $V^{2 \cdot 2}$ alleine durch den Basisvektor b_2 bzw. durch die Koordinaten $v' = (0, 0, 1, 0) \in V^{2 \cdot 2}$ repräsentiert wird (die Nicht-Null-Koordinate entspricht dem Basisvektor b_2). Das heißt, trotz Raumeinbettung von V^2 in $V^{2 \cdot 2}$ bleibt die Anzahl der Nicht-Null-Elemente in den Koordinatenvektoren gleich und erhöht sich nicht, d.h. $v = (0, 1)$ und $v' = (0, 0, 1, 0)$.

Vereinfacht ausgedrückt, wird bei kompatiblen Basen nicht der Koordinatenvektor (der das Muster darstellt) bei einer Raumeinbettung periodisch verlängert, sondern stattdessen die Basisvektoren der Basen entsprechender Vektorräume. Während Ersteres zu einer Erhöhung der Anzahl der Nicht-Null-Koordinaten führt, erhöht sich bei Letzteren die Anzahl der Nicht-Null-Koordinaten lediglich in den Basisvektoren. Die Anzahl der Nicht-Null Elemente in einem entsprechenden Koordinatenvektor (der das Muster repräsentiert) bleibt bei Letzteren allerdings gleich. Das heißt, trotz Einbettung eines Vektors in einen größeren Vektorraum beliebiger Dimension, bleibt die Anzahl der Nicht-Null Koordinaten des Vektors unverändert. Dies erweist sich gegenüber der Verlängerung bei Mustern und der damit verbundenen Erhöhung der Nicht-Null-Koordinaten bzw. Nicht-Null-Elemente (siehe Abschnitt 6.1) als deutlich vorteilhafter.

Die Idee kompatibler Basen werden wir im Folgenden formal charakterisieren. Wir beginnen mit der Definition der periodischen Verlängerung für Vektoren. Die periodische Verlängerung ist wie folgt definiert:

Definition 6.2 (Periodische Verlängerung): *Sei $v = (c_0, ..., c_{m-1})$ ein Vektor des Raumes V^m. Die periodische Verlängerung des Vektor v um den Faktor $a \in \mathbb{N}$ ist der Vektor $v^a = (c_0, ..., c_{m-1}, c_0, ..., c_{m-1}, ...)$ mit $v^a \in V^{a \cdot m}$.*

Die Kompatibilität von Basen kann als Wiederverwendung von Basisvektoren verstanden werden. Das heißt, die Basisvektoren von Raum V^m werden (neben anderen) im Raum $V^{a \cdot m}$ wiederverwendet, jedoch in einer (um den Faktor a) periodisch verlängerten Variante. Allgemeiner betrachtet, betrifft die Wiederverwendung nicht nur einen einzelnen Faktor a bzw. die Basis $B_{a \cdot m}$, sondern alle (um einen beliebigen Faktor) periodisch verlängerten Basen von B_m. Wir bezeichnen ein solches System von Basen als *induktives Basensystem*, das wie folgt definiert ist.

Definition 6.3 (Induktives Basensystem): *Ein System von Basen B_m mit $m \in \mathbb{N}$ ist induktiv, falls für alle $b \in B_m$ gilt: die periodische Verlängerung b^a ist in $B_{a \cdot m}$ enthalten.*

In diesem Zusammenhang gilt es jedoch zwei Aspekte zu bedenken.

1. Es ist nicht offenkundig, dass für jeden Vektorraum eine induktives Basensystem existiert.
2. Es ist ebenfalls nicht offenkundig, dass ein Vektor aus V^m ausschließlich auf die periodisch verlängerten Basisvektoren von B_m im Raum $V^{a \cdot m}$ abgebildet wird und auf keine anderen Basisvektoren von $V^{a \cdot m}$.

Wir werden zunächst auf den ersten Aspekt fokussieren. Der zweite Aspekt ist Gegenstand der Betrachtung von Abschnitt 6.2.5.

Konstruktionsvorschrift für induktive Basensysteme: Auch wenn es nicht offenkundig ist, so gilt, dass für jeden Vektorraum ein induktives Basensystem angeben werden kann, formal ausgedrückt:

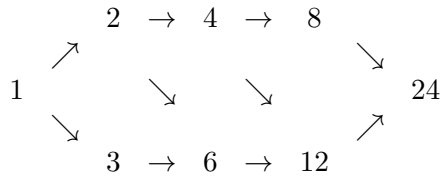
Satz 6.1: *Es existiert ein induktives Basensystem für jeden Vektorraum V^m.*

Beweis: *Siehe Anhang B.2* ∎

Den Existenzbeweis für induktive Basensysteme werden wir im Folgenden in seiner Grundidee vorstellen, da in diesem eine Konstruktionsvorschrift für solche Basensysteme eingeführt wird. Um die Existenz eines induktiven Basensystems zu beweisen, wird von V^m ausgegangen und gezeigt, dass diesem bereits ein induktives Basensystem zugrunde liegt bzw. durch ein solches gebildet werden kann. Per Definition gilt, dass dann auch ein induktives Basensystem für den Vektorraum $V^{a \cdot m}$ besteht, da es sich hier nur um eine, um den Faktor a verlängerte Variante des V^m zugrundeliegenden induktiven Basensystems handelt.

Die Grundidee des Beweises stellt die Betrachtung aller (möglichen) Teiler der Dimension m von V^m dar, d.h. die sogenannte *Teilermenge* der (natürlichen) Zahl m [42]. Die Teilermenge von Raum V^m bzw. seiner Dimension m führt zu einer Menge an Vektorräumen, für die der Raum V^m (potentiell) einen gemeinsamen Vektorraum darstellen kann. Daraus folgt, dass die "Teilervektorräume" die Basisvektoren enthalten, die in periodisch verlängerter Version im Raum V^m wiederverwendet werden, um eine induktive Basis für V^m zu bilden. Beispielsweise besteht für den Vektorraum V^{24} bzw. dessen Dimension die folgende Teilermenge $\mathbf{T}_{12} = \{1, 2, 3, 4, 6, 8, 12\}$. Hieraus folgt, dass bei einer induktiven Basis die Basisvektoren der Teilerräume $\{V^1, V^2, V^3, V^4, V^6, V^8, V^{12}\}$ in periodisch verlängerter Version die Basisvektoren für V^{24} bilden müssen. Nur für diese Räume können die Basisvektoren mit ganzzahligen Faktor periodisch verlängert werden, damit diese einen Basisvektor in V^{24} bilden. Für alle anderen Räume, die kleiner als V^{24} und nicht in dieser Menge enthalten sind, gilt, dass deren Basisvektoren in (ganzzahlig) periodisch verlängerter Variante niemals einen Basisvektor in V^{24} bilden können. Die Teilervektorräume bzw. deren Dimension für einen Vektorraum V^m kann leicht über eine Primzahlzerlegung von m ermittelt werden.

Durch den Beweis zu Satz 6.1 wird sichergestellt, dass für einen beliebigen Vektorraum V^m durch Vereinigung der Basisvektoren seiner Teilervektorräume eine induktive Basis folgt bzw. konstruiert werden kann. Jedoch wird für die Konstruktionsvorschrift die Teilermenge strukturiert bzw. eine Ordnung für diese Menge aufgebaut. Für den Beweis wird hierzu ein spezieller Graph eingeführt, der diese Ordnung repräsentiert. Für das Beispiel V^{24} kann man sich den Graph wie folgt vorstellen:



Vereinfacht ausgedrückt, führt die Ordnung zu (teilweise) disjunkten "Wegen" zur Bildung der Zahl 24. Eine besondere Rolle nehmen die letzten Teiler in einem solchen Weg ein, da diese untereinander keine Teiler darstellen, z.B. wie bei 8 und 12 mit $8 \nmid 12$. Wenn wir nun davon ausgehen, dass wir für die Räume V^8 und V^{12} bereits die Basen B_8 und B_{12} vorliegen haben, so bilden deren Basisvektoren, in periodischer Verlängerung um den Faktor vier bzw. zwei, die Basisvektoren von B_{24}. Das heißt die Basis B_{24} wird durch die Vereinigung der Basen $B_8 \cup B_{12}$ gebildet. Jedoch führt dies noch zu keiner vollständigen Basis. Beide Basen B_8 und B_{12} können gleiche Teiler aufweisen, was wiederum zu Teilerräumen führt, deren Basisvektoren sowohl in B_8 als auch in B_{12} vorkommen können, wie etwa B_4. Aufgrund solcher Überlappungen führt die Vereinigung der (periodisch verlängerten) Basisvektoren im vorliegenden Beispiel lediglich zu 16 Basisvektoren, d.h. es müssen acht Basisvektoren ergänzt werden, um die vollständige Basis B_{24} zu erhalten. Die Basismatrix für den Raum V^{24} weist dann etwa die folgende

Struktur auf, wobei $e_1, ..., e_8$ die Ergänzungsbasisvektoren und B_8, B_{12} die Menge der Basisvektoren der entsprechenden Basen darstellen:

$$B_{24} = \begin{pmatrix} \boxed{B_8} \\ \boxed{B_{12}} \\ e_1 \\ e_2 \\ \vdots \\ e_8 \end{pmatrix}$$

In der Konstruktionsvorschrift werden die Basen B_8 und B_{12} natürlich nicht als bekannt vorausgesetzt, sondern ebenfalls mit der obigen Überlegung konstruiert. Das heißt, für jede der Basen B_8 und B_{12} können die jeweiligen Teilermengen bestimmt werden und für diese erneut rekursiv abgestiegen werden. Die Rekursion bricht bei der Basis B_1 ab, von der wir wissen, dass diese nur einen einzelnen Ergänzungsvektor enthalten kann. Beim Aufsteigen aus der Rekursion "hangelt" sich diese entlang der Teilermengen und kann pro Basis die notwendigen Ergänzungsvektoren ermitteln, d.h. die Basisvektoren der Basen B_1, B_2, B_3, B_4, B_6. Die pro Schritt hinzugefügten Ergänzungsvektoren bilden, in Vereinigung und periodisch verlängerter Version, die Basisvektoren für B_{24}. Für die Basis B_{24} bzw. die enthaltenen Basen B_8 und B_{12} ergeben sich bei der Konstruktionsvorschrift folgende Matrizen, deren Basisvektoren in periodisch verlängerter Version die Basisvektoren in der Basis B_{24} bilden. Einige der Basismatrizen werden bei der Konstruktion (theoretisch) doppelt erzeugt, wie etwa bei B_4, die sowohl in B_8 als auch in B_{12} vorkommt. Daher bildet erst die Vereinigung der periodisch verlängerten Basisvektoren die korrekte Basis für B_{24}.

$$B_8 = \begin{pmatrix} \boxed{B_4} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} \qquad B_4 = \begin{pmatrix} \boxed{B_2} \\ e_1 \\ e_2 \end{pmatrix} \qquad B_2 = \begin{pmatrix} \boxed{B_1} \\ e_1 \end{pmatrix}$$

$$B_{12} = \begin{pmatrix} \boxed{B_4} \\ \boxed{B_6} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} \qquad B_6 = \begin{pmatrix} \boxed{B_2} \\ \boxed{B_3} \\ e_1 \\ e_2 \end{pmatrix} \qquad B_3 = \begin{pmatrix} \boxed{B_1} \\ e_1 \\ e_2 \end{pmatrix} \qquad B_1 = \begin{pmatrix} e_1 \end{pmatrix}$$

Der Beweis von Satz 6.1 stellt sicher, dass die vorgestellte Konstruktionsvorschrift stets zu einem induktiven Basensystem führt. Allerdings wird in dem Beweis lediglich gefordert, dass die pro Rekursionsschritt hinzu zufügenden Ergänzungsbasisvektoren linear

unabhängig sind und somit eine Basis bilden. Aussagen jedoch wie diese Ergänzungsvektoren "aussehen" oder die Eigenschaft der linearen Unabhängigkeit sichergestellt werden kann, werden im Beweis nicht getroffen. Stattdessen wird von einer "geeigneten Wahl" solcher Basisvektoren ausgegangen. Die Auswahl von geeigneten Basisvektoren ist für die Konstruktion induktiver Basen essentiell. Eine solche Auswahl wird im Folgenden mit der Bitletbasis bzw. mit der Standardbitletbasis vorgestellt.

6.2.5 Bitletbasis

Die Einführung induktiver Basensysteme und der Beweis ihrer Existenz für jeden Vektorraum, stellt einen entscheidenden Schritt für eine effiziente Realisierung von Raumeinbettungen dar. Wie sich zeigt, wird bei der Konstruktion induktiver Basen eine adäquate Auswahl an Ergänzungsbasisvektoren benötigt. Aus dieser Auswahl an Vektoren setzt sich jede Basis eines Vektorraums beliebiger Dimension zusammen. Mit der Bitletbasis wird eine solche Auswahl an Basisvektoren eingeführt.

6.2.5.1 Allgemeine Bitletbasis

Auch wenn die Auswahl der Ergänzungsbasisvektoren generell beliebig erfolgen kann, empfiehlt sich für die Anwendungsdomäne der Wiederholungsmuster ein differenzierteres Vorgehen.

Der zu berücksichtigende Zusammenhang ist, dass die Wahl der Basisvektoren in direkter Weise auch den Umfang der Abbildung von Wiederholungsmustern in einem entsprechenden Vektorraum bedingt. Genauer gesagt, durch die Basis wird die Anzahl der Nicht-Null-Koordinaten zur Darstellung eines Wiederholungsmusters beeinflusst. Die Basisvektoren können als "Grundbestandteile" verstanden werden, durch die ein Wiederholungsmuster in dem entsprechenden Vektorraum abgebildet wird. Das heißt, das komplexe periodische Verhalten eines Musters wird im Vektorraum durch diese Bestandteile bzw. Basisvektoren abgebildet. Somit gilt es, "passende" Basisvektoren für ein Muster bzw. Musterklassen auszuwählen, die das periodische Verhalten des Musters gut widerspiegeln. Infolgedessen würde ein einzelner Basisvektor das periodische Verhalten eines Musters bereits in "großen Teilen" alleine darstellen können. Es würden nur einige wenige zusätzliche Basisvektoren benötigt werden, um das Verhalten des Muster vollständig abbilden zu können. Zum Muster passende Basen führen dazu, dass nur wenige Basisvektoren zur Abbildung des Musters benötigt bzw. nur wenige Nicht-Null-Koordinaten erzeugt werden. Diese Verringerung der Nicht-Null-Koordinaten erweist sich nicht nur für den Umfang der Darstellung als vorteilhaft, sondern auch für die (später vorgestellte) Realisierung von Operationen. Abbildung 6.7 zeigt diese Idee an einem Beispiel. Das Wiederholungsmuster r entspricht einem Periodizitätsschema mit Zyklen der Länge 3, wobei der 2. Zyklus und der 4. Zyklus Ausnahmen enthalten. Im mittleren Bereich ist ein Ausschnitt einer Basis mit "gut passenden" Basisvektoren dargestellt.

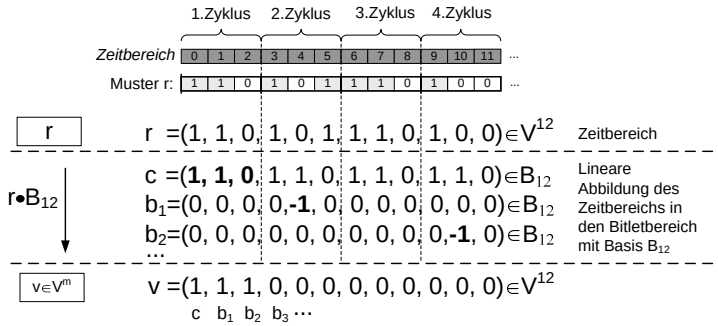


Abbildung 6.7: Beispiel für musterangepasste Basisvektoren

Der Basisvektor c entspricht in seinem periodischen Verhalten dem Grundmuster des Periodizitätsschemas ohne Ausnahmen. Die beiden Ausnahmen im Periodizitätsschema werden durch die beiden Vektoren b_1 und b_2 abgebildet. Wenn wir davon ausgehen, dass keine weiteren Basisvektoren zur Abbildung des Musters benötigt werden, ergibt sich der im unteren Teil dargestellte Koordinatenvektor v. Der Vektor v repräsentiert das Muster r in dem entsprechenden Vektorraum bzw. *Bitletbereich* (siehe später). Es zeigt sich, dass im Bitletbereich lediglich drei Nicht-Null-Koordinaten zur Darstellung benötigt werden, im Gegensatz zu den sieben Nicht-Null-Elementen des Musters r.

Die allgemeine Idee bei Bitletbasen ist es, anstelle die Basisvektoren frei zu wählen, diese auf Grundlage eines anzugebenden Grundmusters entsprechend zu konstruieren. Die formale Charakterisierung der allgemeinen Idee zur musterangepassten Auswahl von Basisvektoren führt zu dem Begriff der *Bitletbasis*. Analog zu anderen Transformationen wie etwa Fourier-, Kosinus- oder Wavelet-Transformation führen wir mit *Bitlet*, *Bitletbasis* und *Bitlet-Transformation* besondere Bezeichnungen für musterangepasste Basisvektoren, Vektorraumbasis und lineare Abbildung in den entsprechenden Vektorraum (Bitletbereich) ein. Die Bitletbasis ist wie folgt definiert:

Definition 6.4 (Bitletbasis): *Ein Bitletsystem ist ein Paar (c, Ψ) bestehend aus einem Grundvektor $c \in V^m$ und einer Konstruktionsvorschrift $\Psi(c, m) = B_m$, die durch skalierte und verschobene Kopien des Grundvektors c eine Basis B_m mit Dimension $m \in \mathbb{N}$ konstruiert, sodass B_m induktiv ist. Die erzeugte Basis B_m wird Bitletbasis genannt und seine Basisvektoren als Bitlets bezeichnet.*

Die allgemeine Definition der Bitletbasis geht von einer Konstruktionsvorschrift aus, die für ein gegebenes Grundmuster eine vollständige Basis bestimmt. Das heißt, das Grundmuster stellt ein Bitlet bzw. Grundbitlet dar, das das periodische Verhalten eines bzw. einer anwendungsabhängigen Klasse von Wiederholungsmustern gut widerspiegelt. Aufbauend auf dem Grundbitlet bzw. mit skalierten und verschobenen Kopien des selbigen konstruiert die Vorschrift Ψ eine vollständige Basis, die induktiv ist und als Bitletbasis bezeichnet wird. Auch wenn formal eine solche Konstruktionsvorschrift Ψ leicht gefor-

dert werden kann, so liegt die Herausforderung in der konkreten Angabe derselben. Hierfür muss, unabhängig vom konkreten Grundmuster, stets dem gleichen Problem begegnet werden: Es muss sichergestellt werden können, dass die Konstruktionsvorschrift stets zu einer Basis führt, d.h. die Menge der konstruierten Basisvektoren muss linear unabhängig sein.

Die lineare Unabhängigkeit der Basisvektoren spielt zusätzlich auch bei dem erwähnten zweiten nicht-offenkundigen Aspekt bei induktiven Basensystemen eine entscheidende Rolle (siehe Abschnitt 6.2.4). Bei diesem muss sichergestellt werden, dass die Koordinate eines Basisvektors in V^m ausschließlich auf die Koordinate seiner periodisch verlängerten Variante in $V^{m \cdot n}$ abgebildet wird (und auf keine anderen Basisvektoren von $V^{m \cdot n}$). Um dies sicherzustellen ist es zwingend notwendig, dass die Basisvektoren von B_m und die Basisvektoren B_n in der Basis $B_{m \cdot n}$ ebenfalls linear unabhängig sind. Würde dies nicht zutreffen und ein Basisvektor von B_m (oder B_n) ist linear abhängig in $B_{m \cdot n}$, dann könnte dieser als Linearkombination einer endlichen Menge an anderen Basisvektoren beschrieben werden. Infolgedessen, würde seine (einzelne) Koordinate in B_m (oder B_n) in eine (endliche) Menge an Koordinaten (die Faktoren der Linearkombination) in der Basis $B_{m \cdot n}$ aufgeteilt werden. In Bezug auf die Raumeinbettung würde dies zu der eigentlich zu vermeidenden Erhöhung der Nicht-Null-Koordinaten entsprechender Vektoren führen.

Die Herausforderung bei der Angabe einer Konstruktionsvorschrift Ψ zur Bildung einer Bitletbasis besteht in der Sicherstellung der linearen Unabhängigkeit der Basisvektoren bzw. Bitlets. Das allgemeine Problem ist, dass obwohl Basisvektoren innerhalb der Basen B_m und B_n jeweils linear unabhängig sind, folgt im Allgemeinen nicht ihre lineare Unabhängigkeit in der Basis $B_{m \cdot n}$. Darüber hinaus ist es kein triviales Problem bzw. sehr aufwändig, die lineare Unabhängigkeit von Vektoren zu überprüfen. Infolgedessen muss mit der Konstruktionsvorschrift Ψ die lineare Unabhängigkeit von Bitlets bzw. induktiven Bitletsbasen sichergestellt werden können – ohne aufwändige Überprüfungen vornehmen zu müssen. Eine solche Konstruktionsvorschrift wird im Folgenden mit der Standardbitletbasis vorgestellt.

6.2.5.2 Die Standardbitletbasis

Als Grundmuster der Standardbitletbasis werden Standardeinheitsvektoren (von rechts) als Bitlets verwendet, die (wie wir später zeigen werden) zu linearer Unabhängigkeit per Konstruktion führt. Die Motivation hinter der Auswahl dieser Bitlets liegt hauptsächlich in ihren positiven Eigenschaft bzgl. der linearen Unabhängigkeit. Welche Wiederholungsmuster bzw. welches periodische Verhalten von Mustern durch diese Bitlets besonders "passend" bzw. mit wenigen Nicht-Null-Koordinaten unterstützt werden, wird im Folgenden zunächst vernachlässigt. Entsprechende Betrachtungen finden in Abschnitt 6.6 statt. Die Standardbitletbasis ist wie folgt definiert:

Definition 6.5 (Standardbitletbasis): *Sei eine Folge von Standardeinheitsvektoren wie folgt definiert:* $e_1^m = (0, \ldots, 1), e_2^m = (0, \ldots, 1, 0), ..., e_n^m = (0^{m-n}, 1, 0^{n-1})$ *mit* $e_i^m \in V^m$ *und* $n < m$. *Sei* $\mathbf{T}_m = \{t_1, ..., t_k\}$ *die Teilermenge der Dimension* m. *Das Bitletsystem* (e_1, Ψ) *mit* $\Psi(e_1^m, m) = B_m$ *wird als Standardbitletsystem und die Basis* B_m *als Standardbitletbasis bezeichnet, falls gilt:*

$$B_m = \{B'_{t_1} \cup ... \cup B'_{t_k}\} \cup \{e_1^m, e_2^m, ..., e_{m-(\#p-1)}^m\}$$

mit B'_{t_i} *als Menge der um den Faktor* $\frac{m}{t_i}$ *periodisch verlängerten Basisvektoren der Standardbitletbasis* $\Psi(e_1^{t_i}, t_i)$ *mit* $1 \leq i \leq k$ *und* $\#p = \left\| \bigcup_{i=1,...,k} B'_{t_i} \right\|$.

Es werden stets Standardeinheitsvektoren als Ergänzungsvektoren bei der Konstruktion der Standardbitletbasis als induktives Basensystem verwendet. Genauer gesagt werden Standardeinheitsvektoren verwendet, in den die "Eins" von hinten (rechts) beginnend für jeden einzufügenden Ergänzungsvektor nach "links" wandert. In der Definition ist dieses Verhalten durch die Folge der Standardeinheitsvektoren definiert. Die Anzahl der zusätzlich notwendigen Ergänzungsvektoren ist abhängig von der Anzahl der Basisvektoren der entsprechenden Teilerräume (in der Definition mit dem Parameter $\#p$ bezeichnet).

Die Struktur der Standardbitletbasis ist im Folgenden anhand von einigen Beispielen dargestellt:

$$B_1 = (1) \quad B_2 = \begin{pmatrix} \mathbf{1} & 1 \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \quad B_3 = \begin{pmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

$$B_4 = \begin{pmatrix} \mathbf{1} & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \quad B_5 = \begin{pmatrix} \mathbf{1} & 1 & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \quad B_6 = \begin{pmatrix} \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

$$
B_8 = \begin{pmatrix}
\mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & B_1 \\
\mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 & 0 & 1 & B_2 \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 1 & 0 & B_4 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 1 & \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & e_4 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & e_3 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & e_2 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & e_1
\end{pmatrix}
$$

Periodisch verlängerte Basisvektoren der Teilerräume

Ergänzungsbasisvektoren

Beispielsweise ergibt sich für den Raum V^6 bzw. dessen Dimension die Teilermenge $\mathbf{T_6} = \{1, 2, 3\}$. Das heißt, die Bitletbasis B_6 ist konstruiert aus periodisch verlängerten Bitlets von B_1, B_2 und B_3. Das Bitlet $(0\,1\,0\,0\,1\,0) \in B_6$ (vierte Bitlet von oben in der Matrix) etwa, ist die periodisch verlängerte Version des Bitlets $(0\,1\,0) \in B_3$. In Summe führen diese Basen zu vier Basisvektoren die ergänzt um die Bitlets $(0\,0\,0\,0\,0\,1)$ und $(0\,0\,0\,0\,1\,0)$ die Basis für V^6 bilden. Die Beispielmatrizen zeigen, dass die periodische Verlängerung der trivialen Basis B_1 in jeder anderen Basis enthalten ist. Der Teiler 1 ist Bestandteil einer jeden Teilermenge und muss somit nach der Konstruktionsvorschrift für induktive Basen konsequenterweise auch in jeder Basis vorkommen.

Aufgrund der Induktivität der Basen können die Koordinatendarstellungen unterschiedlich dimensionaler Räume leicht in ihren gemeinsamen Vektorraum eingebettet werden. Beispielsweise haben das Muster $r_1 \in V^3$ mit der Koordinatendarstellung $v_1 = (0, 1, 1) \in V^3$ bzgl. der Standardbitletbasis B_3 und das Muster $r_2 \in V^2$ mit der Koordinatendarstellung $v_2 = (0, 1) \in V^2$ bzgl. der Standardbitletbasis B_2 den gemeinsamen Vektorraum V^6. Aufgrund der induktiven Eigenschaft der Standardbitletbasis ergeben sich nach der Einbettung die Koordinatendarstellung $v_1^{'} = (0, 0, 1, 1, 0, 0) \in V^6$ bzw. $v_2' = (0, 1, 0, 0, 0, 0) \in V^6$ bzgl. der Standardbitletbasis B_6. Es zeigt sich, dass bei beiden Koordinatendarstellungen die Anzahl der Nicht-Null-Koordinaten gleich bleibt.

Die Standardbitletbasis stellt per Konstruktion zwei Eigenschaften sicher. Erstens sind die konstruierten Basen induktiv, da in der Konstruktionsvorschrift dasselbe Vorgehen wie im Beweis zu Satz 6.1 verwendet wird. Zweitens wird durch die Verwendung von Standardeinheitsvektoren als Bitlets, die lineare Unabhängigkeit per Konstruktion sichergestellt. Letzteres wird im folgenden Satz formal beschrieben und bewiesen.

Satz 6.2: *Das Standardbitletsystem erzeugt Basen.*

Beweis: *Siehe Anhang B.3* ■

Mit dem obigen Beweis zeigt sich, dass die Standardbitletbasis den ersten Aspekt für eine effiziente Realisierung von Raumeinbettungen unterstützt (siehe Abschnitt 6.2.3.3). Zum einen kann bei dieser Basis für jeden Vektorraum beliebiger Dimension eine induktive Basenfolge angegeben werden, zum anderen ist die lineare Unabhängigkeit der

Bitlets – ohne Überprüfung – stets gewährleistet. Ersteres ermöglicht es grundsätzlich, unterschiedlich dimensionale Räume einfach in einen gemeinsamen Vektorraum zu überführen. Mit der zweiten Eigenschaft wird die formale Eigenschaft von Basen sichergestellt. Diese bedingt jedoch auch, dass sich bei der Überführung in den gemeinsamen Vektorraum die Anzahl der Nicht-Null-Koordinaten nicht erhöht und – unabhängig von der Dimension des gemeinsamen Vektorraums – gleich bleibt.

6.2.6 Bitlet-Transformation

Als *Bitlet-Transformation* wird die Überführung eines Wiederholungsmusters aus dem Zeitbereich in die Koordinatendarstellung des Musters bezüglich einer Bitletbasis bezeichnet. Anders ausgedrückt, wird mit der Bitlet-Transformation das Muster in seine periodischen Grundelemente bzw. Bitlets zerlegt. Wie in Abschnitt 6.2.2.2 gezeigt, entspricht eine solche Überführung einer linearen Abbildung zwischen zwei Vektorräumen. Im vorliegen Fall also vom Vektorraum des Zeitbereichs in den Vektorraum mit Bitletbasis. Da es sich bei beiden um endlich dimensionale Vektorräume handelt, kann die Realisierung der linearen Abbildung durch die Darstellungsmatrix der Basis erfolgen. Gegeben sei eine Bitletbasis B_m und ein Wiederholungsmuster $r \in V^m$ der Länge m. Mit Hilfe der Bitlet-Transformation ergibt sich die Koordinatendarstellung des Wiederholungsmusters bezüglich der Bitletbasis B_m durch die Matrixmultiplikation $v = r \cdot B_m^{-1}$. Beispielsweise führt das Muster $r = (1, 0, 1) \in V^3$ zu dem Koordinatenvektor $v = r \cdot B_3^{-1} = (1, -1, 0) \in V^3$ bezüglich der Basis B_3. Analog wird die Überführung der Koordinatendarstellung bezüglich einer Bitletbasis in ein Wiederholungsmuster im Zeitbereich als *inverse Bitlet-Transformation* bezeichnet[6]. Das heißt, wir können jeden Koordinatenvektor bezüglich einer Basis B_m in ein Wiederholungsmuster r im Zeitbereich überführen mit $r = v \cdot B_m$.

Allerdings ist die Bitlet-Transformation basierend auf einer expliziten Matrixmultiplikation lediglich für die formale Semantik der Transformation geeignet. Für eine effiziente Realisierung der Transformation empfiehlt es sich nicht, solche Matrizen konkret zu erzeugen bzw. explizit eine entsprechende Multiplikation durchzuführen. In der Praxis können relativ schnell Musterlängen bzw. entsprechende Matrizendimensionen entstehen, die sich als problematisch erweisen können. Stattdessen gilt es, wenn möglich, Matrizen nicht explizit zu verwenden, sondern diese "indirekt" abzubilden. Eine solche indirekte Abbildung der Basismatrizen führt zu der (inversen) Fast-Bitlet-Transformation, die Gegenstand der Betrachtungen von Abschnitt 6.3 sein wird.

[6] An dieser Stelle muss darauf hingewiesen werden, dass sich der Begriff "Invers" auf die Richtung der Transformation, d.h. vom Bitletbereich in den Zeitbereich, bezieht und nicht auf die Basismatrix. Da bei der inversen Transformation die ursprüngliche Matrix verwendet wird und bei der Bitlet-Transformation die inverse Basismatrix kann es hier leicht zu Verwechslungen kommen.

6.2.7 Bitletrepräsentation

Mit der Einführung der Bitletbasis und der Bitlet-Transformation wird eine Repräsentationsform für Wiederholungsmuster ermöglicht, die dem Problem der operationalen Unterstützung bei Mustern begegnen kann. Diese *Bitletrepräsentation* von Wiederholungsmustern wird im Folgenden vorgestellt.

6.2.7.1 Allgemeine Bitletrepräsentation

Die Überführung eines Wiederholungsmusters mittels Bitlet-Transformation in den Bitletbereich führt zu einem entsprechenden Koordinatenvektor. Grundsätzlich stehen für diesen Vektor die gleichen Repräsentationsmöglichkeiten zur Verfügung wie wir diese bereits für Wiederholungsmuster vorgestellt haben (siehe Abschnitt 6.1). Das heißt, der Koordinatenvektor kann in einer dichtbesetzten oder dünnbesetzten Repräsentationsform abgebildet werden. In der dichtbesetzten Form wird der vollständige Vektor repräsentiert, unabhängig von dem konkreten Wert einer einzelnen Koordinate. In der dünnbesetzten Form hingegen, werden lediglich die Nicht-Null-Koordinaten des Vektors zusammen mit der Länge des Vektors bzw. Dimension des entsprechenden Raums gespeichert.

In Hinblick auf die Eigenschaften von Bitletbasen empfiehlt sich, die dünnbesetzte Repräsentationsform des Koordinatenvektors zu verwenden. Wie gezeigt, stellt die Raumeinbettung unterschiedlich dimensionaler Räume in einem gemeinsamen Vektorraum das korrespondierende Problem zur Verlängerung von Wiederholungsmustern im Zeitbereich dar. Mit der Einführung von Bitletbasen bzw. ihrer Eigenschaft der Induktivität haben wir zwar eine Möglichkeit gefunden, dass bei der Raumeinbettung die Anzahl der Nicht-Null-Koordinaten stabil bleibt, die Länge des Koordinatenvektors "wächst" jedoch unverändert weiter. Das heißt, bei einer Raumeinbettung des Vektors $v \in V^m$ bleibt die Anzahl der Nicht-Null-Koordinaten im korrespondierenden Vektor $v' \in V^{a \cdot m}$ gleich. Die Dimension bzw. Länge des Koordinatenvektors verändert sich aber weiterhin. Bei einer dichtbesetzten Repräsentationsform führt dies dazu, dass mit dem (unvermeidbaren) Anstieg der Dimension des Raums gleichzeitig auch die Länge des Koordinatenvektors entsprechend beeinflusst wird. Es entsteht die gleiche Problematik bzgl. Umfang der Repräsentation wie bei der Verlängerung von Wiederholungsmustern. Infolgedessen kann lediglich eine dünnbesetzte Repräsentationsform, d.h. lediglich die Repräsentation der Nicht-Null-Koordinaten, der Problematik bzgl. Umfang der Repräsentation begegnen. Die zur Repräsentation notwendige Anzahl an Koordinaten bleibt auch nach einer Raumeinbettung gleich, auch wenn die Dimension des Raums ansteigt.

Zur Realisierung einer dünnbesetzten Repräsentationform bedarf es einer geeigneten Adressierung bzw. Identifikation der Nicht-Null-Koordinaten. Bei Wiederholungsmustern wurde dies durch die Angabe der relativen Position im Muster realisiert, d.h. das Muster $r = (0, 1, 0, 1)$ wurde in der dünnbesetzten Form als Tupel $(4, \{1, 4\})$ repräsentiert. Im Bitletbereich bzw. bei einem entsprechenden Koordinatenvektor stehen

die Nicht-Null-Koordinaten für bestimmte Bitlets innerhalb der Bitletbasis. Für diese Bitlets gilt es eindeutige "Identifikatoren" zu verwenden, die dann in der dünnbesetzten Repräsentationform hinterlegt werden. Beispielsweise stehen die beiden Nicht-Null-Koordinaten des Vektors $v = (0, 1, 1) \in V^3$ für die Bitlets b_2 und b_3 in der untenstehenden Standardbitletbasis B_3. Basierend auf der dargestellten "Durchnummerierung" der Bitlets in der Basis kann der Vektor v durch die dünnbesetzte Repräsentation $v_{dünn} = (3, \{b_2, b_3\})$ abgebildet werden.

$$B_3 = \begin{pmatrix} b_1: & \mathbf{1} & 1 & 1 \\ b_2: & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ b_3: & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

Im Zusammenhang mit Bitletbasen ergibt sich jedoch ein Problem. Die angedeutete "einfache" Durchnummerierung von Bitlets in einer Basis reicht nicht aus in Hinblick auf Raumeinbettungen. Genauer gesagt, diese Form der Identifikation der Bitlets ist über das induktive Basensystem nicht stabil bzw. eindeutig. Beispielsweise führt dies dazu, dass die Bitlets b_2 und b_3 des Vektors $v_{dünn} = (3, \{b_2, b_3\}) \in V^3$ bzw. deren periodisch verlängerte Varianten nach Raumeinbettung in V^6 als die Bitlets b_4 und b_3 bzgl. der Basis B_6 indiziert werden. Nach der Raumeinbettung ergibt sich die veränderte Repräsentation $v'_{dünn} = (6, \{b_4, b_3\})$.

$$B_6 = \begin{pmatrix} b_1: & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ b_2: & \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 \\ b_3: & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 \\ b_4: & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\ b_5: & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ b_6: & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

In Hinblick auf die effiziente Raumeinbettung gilt es, solche "schwankende" oder über das induktive Basensystem uneindeutige Indizierungen von Bitlets zu vermeiden. Stattdessen gilt es, eine eindeutige Indizierung von Bitlets über das gesamte induktive Basensystem einer Basis sicherzustellen. Bei eindeutiger Indizierung bleibt die Identifikation der Bitlets auch nach Raumeinbettungen gleich und folglich auch eine entsprechende dünnbesetzte Repräsentationsform unverändert.

Diese Anforderung einer eindeutigen Indizierung von Bitlets über das gesamte induktive Basensystem führt zu dem Begriff der *indizierten Bitletbasis*, der wie folgt definiert ist:

Definition 6.6 (Indizierte Bitletbasis): *Eine indizierte Bitletbasis ist ein Paar (B_m, I) bestehend aus der Bitletbasis B_m und einer Indizierungsfunktion $I(b, B_m)$, die jedes Bitlet $b \in B_m$ und seine um Faktor a periodisch verlängerte Variante b^a in Basis $B_{a \cdot m}$ mit $a \in \mathbb{N}$ eindeutig identifiziert, sodass gilt: $I(b, B_m) = I(b^a, B_{a \cdot m})$.*

In einer indizierten Bitletbasis gehen wir von einer Indizierungsfunktion I aus, die ein Bitlet eindeutig bezüglich seiner Basis identifiziert. Diese Identifikation bzw. Indizierung

bleibt auch bei einer Raumeinbettung bzgl. eines Faktors a, d.h. über die induktiven Basen B_m und $B_{a \cdot m}$, gleich. Die Angabe bzw. Definition einer konkreten Indizierungsfunktion ist jedoch abhängig von der konkreten Bitletbasis bzw. von den verwendeten Bitlets. Wichtig ist hierbei "nur", dass die Indizierungsfunktion die geforderte Eigenschaft $I(b, B_m) = I(b^a, B_{a \cdot m})$ erfüllt, d.h. über die induktive Basen hinweg eindeutig bleibt.

Die Eigenschaft indizierter Bitletbasen bzw. ihrer Indizierungsfunktion ist es, die eine dünnbesetzte Repräsentation des Koordinatenvektors – die Bitletrepräsentation – ermöglicht. Die Bitletrepräsentation ist wie folgt definiert:

Definition 6.7 (Bitletrepräsentation): *Sei $r \in V^m$ ein Wiederholungsmuster und (B_m, I) eine indizierte Bitletbasis. Das Tupel $[m, W = \{(I(b_i, B_m), z_i), ..., (I(b_k, B_m), z_j)\}]$ wird als Bitletrepräsentation des Vektors $v = r \cdot B_m^{-1}$ bezeichnet, falls für alle $(I(b_k, B_m), z_k) \in W$ gilt: $z_k \in \mathbb{Z}/\{0\}$ und $b_k \in B_m$ sind die Nicht-Null-Koordinaten des Vektors v.*

Bei der Bitletrepräsentation wird jedes Nicht-Null-Element eines Koordinatenvektors bzw. der durch die Indizierungsfunktion $I(b, B_m)$ festgelegte Index für das Bitlet b zusammen mit dem Wert der Koordinate gespeichert. Zusätzlich wird die Dimension m des Vektorraums gespeichert.

Aufgrund der indizierten Bitletbasis weist die Bitletrepräsentation bei Raumeinbettungen besonders "gute" Eigenschaften auf. Durch die Indexfunktion weist jedes Bitlet stets eine eindeutige und über eine Raumeinbettung hinweg stabile Indizierung auf. Dies hat zur Folge, dass in der Bitletrepräsentation bei einer Raumeinbettung lediglich die Dimension des Raums angepasst werden muss. Die Menge der repräsentierten Bitlets bleibt unverändert, aufgrund der eindeutigen Indizierung der Bitlets. Diese Eigenschaft wird im folgenden Korollar formal beschrieben und ergibt sich aufgrund der gefordertem Eigenschaft $I(b, B_m) = I(b^a, B_{a \cdot m})$ der Indexfunktion I.

Korollar 6.1: *Sei $[m, \{(I(b_h, B_m), z_h), ..., (I(b_k, B_m), z_k)\}]$ die Bitletrepräsentation des Vektors $v \in V^m$. Die Bitletrepräsentation $[a \cdot m, \{(I(b_h, B_m), z_h), ..., (I(b_k, B_m), z_k)\}]$ stellt den Vektor $v \in V^m$ nach Raumeinbettung in dem Raum $V^{a \cdot m}$ dar.*

Auf Grundlage indizierter Bitletbasen kann mit der Bitletrepräsentation eine Repräsentationsform angegeben werden, bei der eine Raumeinbettung effizient repräsentiert werden kann. Im Gegensatz zu Wiederholungsmustern, führt eine Verlängerung bzw. die korrespondierende Einbettung von Vektorräumen bei der Bitletrepräsentation zu keiner Erhöhung der zu repräsentierenden Daten. Im Gegenteil, die Anzahl der zu repräsentierenden Werte bleibt gleich – unabhängig von der Dimension des Raums bzw. dem Grad der Verlängerung des korrespondierenden Musters. Voraussetzung für diese Eigenschaft der Bitletrepräsentation ist die Angabe einer geeigneten Indizierungsfunktion für die entsprechenden Bitletbasen. Für die Standardbitletbasis wird im Folgenden eine solche Indizierungsfunktion vorgestellt.

6.2.7.2 Bitletrepräsentation der Standardbitletbasis

Die Grundlage der Bitletrepräsentation ist eine eindeutige und über induktive Basensysteme hinweg stabile Indizierung der entsprechenden Bitlets. Im Zusammenhang mit der Standardbitletbasis (siehe Abschnitt 6.2.5.2) ergibt sich eine entsprechende Indizierungsfunktion aufgrund der besonderen Gestalt der Bitlets.

In der Standardbitletbasis stellt jedes Bitlets lediglich einen Standardeinheitsvektor dar, d.h. ein Bitlet enthält in seiner unverlängerten "Urform" lediglich eine einzelne Eins an einer bestimmten Position im Basisvektor. Infolgedessen kann jedes Bitlet stets durch zwei Parameter, (1) Länge des Bitlet in seiner Urform und (2) Position der Eins, identifiziert werden. Die Verwendung dieser beiden Parameter zur Indizierung der Bitlets als $b_{Länge,Position}$ führt zu einer eindeutigen Identifizierung der Bitlets auch für eingebettete Räume $V^{a \cdot m}$. Beispielsweise stellt der Index $b_{3,2}$ das Bitlet $(0,1,0) \in B_3$ dar. Bei der Einbettung in den Raum $V^{2 \cdot 3}$ stellt das Bitlet $(0,1,0,0,1,0) \in B_6$ das um Faktor a verlängerte Bitlet $b_{3,2}$ dar. Dieses Bitlet der Basis B_6 wird jedoch durch das indizierte Bitlet $b_{3,2}$ der Basis B_3 eindeutig charakterisiert. Es muss lediglich berücksichtigt werden, dass das Bitlet $b_{3,2}$ periodisch verlängert werden muss, um dass vollständige Abbild des Bitlets in Basis B_6 zu erhalten. Da in der Bitletrepräsentation jedoch die Dimension des Raums mit gespeichert wird, ist der Faktor, um den das Bitlet verlängert werden muss, bekannt bzw. ermitteltbar. Die Bitletbasis B_6, beispielsweise, besteht aus Bitlets der Basen B_2 und B_3. Jedes Bitlet bzw. die gesamte Basis kann durch diese Indizierung der Bitlets eindeutig charakterisiert werden mit $B_6 = \{b_{1,1}, b_{2,1}, b_{3,2}, b_{3,3}, b_{6,5}, b_{6,6}\}$, d.h. es ergibt sich die Basis:

$$B_6 = \begin{pmatrix} b_{1,1}: & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ b_{2,1}: & \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 \\ b_{3,2}: & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\ b_{3,3}: & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 \\ b_{6,5}: & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ b_{6,6}: & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

Diese Form der Indizierung wird in der *indizierten Standardbitletbasis* verwendet, die wie folgt definiert ist:

Definition 6.8 (Indizierte Standardbitletbasis): *Sei $B_{t_1}, ..., B_{t_k}$ die Mengen an Basisvektoren der Teilerräume von B_m mit $\mathbf{T}_m = \{t_1, ..., t_k\}$ und B_E die Menge der Ergänzungsbasisvektoren mit $B_E = B_m / \{B_{t_1}, ..., B_{t_k}\}$. Das Paar (B_m, I) wird als indizierte Standardbitletbasis bezeichnet, falls B_m eine Standardbitletbasis ist und für die Indizierungsfunktion $I(b, B_m)$ gilt:*

- *$b \in B_E \Rightarrow I(b, B_m) = b_{m,i}$ mit Position i der einzigen Eins im Bitlet*
- *$b \in B_{t_i} \Rightarrow I(b, m) = I(b', t_i)$ mit b als periodisch verlängertes Bitlet $b' = b^{\frac{m}{t_i}}$*

In der Definition werden zur Realisierung der Indexfunktion zwei Fälle unterschieden. Bitlets, die im Bereich der Ergänzungsvektoren von B_m liegen und Bitlets, die periodisch verlängerte Versionen der Teilerräume darstellen. Im ersteren Fall ergibt sich der zweidimensionale Indexwert eines Bitlets unmittelbar aus der Dimension des Raums und der Position der (einzigen) Eins im Bitlet. Im zweiten Fall hingegen, wird der bereits bestehende Indexwert des Bitlets in dem entsprechenden Teilerraum verwendet, der aufgrund der rekursiven Definition ebenfalls nur aus den zwei Parametern Dimension und Position der Eins besteht.

Die vorgestellte Indexfunktion eignet sich in dieser Form lediglich für die Festlegung der formalen Semantik. Die konkrete Realisierung der Indexfunktion oder auch die Überführung eines Wiederholungsmusters in die korrespondierende Bitletrepräsentation ist Teil der Fast-Bitlet-Transformation, die erst in Abschnitt 6.3 vorgestellt wird. Dennoch, unabhängig von der konkreten Realisierung der Indexfunktion, ergibt sich aufgrund ihrer Eindeutigkeit stets, dass die Anzahl der zu repräsentierenden Werte in der Bitletrepräsentation bei Raumeinbettungen gleich bleibt – unabhängig von der Dimension des (höheren) entsprechenden Vektorraums. Beispielsweise hat der Vektor $v = (0, 1, 1) \in V^3$ die Bitletrepräsentation $v = [3, \{(b_{3,2}, 1), (b_{3,3}, 1)\}] \in V^3$, die bzgl. der Bitlets auch bei einer Raumeinbettung in den Raum $V^{6=2\cdot 3}$ gleich bleibt, d.h. $v' = [3 \cdot 2, \{(b_{3,2}, 1), (b_{3,3}, 1)\}] \in V^6$. Lediglich die Dimension des eingebetteten Raums muss in der Bitletrepräsentation angepasst werden, unabhängig von der konkreten Raumeinbettungen, wie etwa $V^6, V^9, V^{12}, \ldots$.

Mit der Stabilität der Bitletrepräsentation bei Raumeinbettungen ist eine der drei Voraussetzung zur effizienten operationalen Unterstützung von Mustern im Bitletbereich erfüllt (siehe Abschnitt 6.2.3.2). Im Gegensatz zu Wiederholungsmustern findet bei der Bitletrepräsentation kein Anstieg der zu repräsentierenden Werte bei Vereinheitlichung der Räume bzw. der korrespondierenden unterschiedlichen Musterlängen statt. Die zweite Voraussetzung, d.h. eine effiziente Hin- und Rücktransformation eines Wiederholungsmusters in (aus) den Vektorraum wird im nachfolgenden Abschnitt vorgestellt.

6.3 Fast-Bitlet-Transformation

Eine effiziente Transformation von Wiederholungsmustern aus dem Zeitbereich in den Bitletbereich und umgekehrt ist für den praktischen Einsatz der Bitletrepräsentation unumgänglich. So sind Wiederholungsmuster mit langen Periodenlängen häufig in der Praxis, sei es aufgrund von spezifischen Ausnahmen oder die Verwendung von bestimmter Granularitäten (z.B. *Monat*). Bei der Betrachtung der Muster als Vektoren eines Vektorraums ergeben sich durch die langen Periodenlängen ebenfalls auch entsprechend große Raumdimensionen. Die Raumdimension wiederum schlägt sich in der Größe der Abbildungsmatrix der Bitletbasis nieder, d.h. die Dimension m führt zu der Matrix $m \times m$. Für eine effiziente Realisierung der Transformation empfiehlt es sich daher nicht, Matrizen mit solchen potentiell großen Dimensionen explizit zu repräsentieren bzw. bei der Bitlet-Transformation explizit die Matrixmultiplikation $v = r \cdot B_m^{-1}$ durchzuführen. Stattdessen sollten, wenn möglich, solche großen Matrizen "indirekt" abgebildet werden bzw. mit diesen umgegangen werden. Allerdings ist eine indirekte Repräsentation und Umgang unmittelbar von den Eigenschaften solcher Matrizen abhängig. Im vorliegenden Abschnitt konzentrieren wir deshalb unsere Betrachtungen auf Standardbitletbasen (siehe Abschnitt 6.2.5.2).

Im Folgenden wird mit der inversen Fast-Bitlet-Transformation eine indirekte Abbildung bzw. ein indirekter Umgang mit Standardbitletbasen vorgestellt. Hierzu wird in Abschnitt 6.3.1 zunächst die Grundidee der indirekten Abbildung vorgestellt. Im Anschluss daran wird für die Standardbitletbasis in Abschnitt 6.3.2 die inverse Fast-Bitlet-Transformation eingeführt und eine Komplexitätsanalyse der Transformation vorgenommen. Abschließend wird in Abschnitt 6.3.3 die Realisierungen der Fast-Bitlet-Transformation betrachtet.

6.3.1 Grundidee

Die Grundidee zur Realisierung der Fast-Bitlet-Transformation basiert auf der Überlegung, dass die benötigten Basismatrizen bei einer Transformation nicht "am Stück" benötigt werden. Stattdessen findet zur Transformation eine Vektormultiplikation eines Wiederholungsmusters oder Bitletvektor mit (allen) Spalten der Matrix statt. Im Ergebnis führt die Multiplikation pro Spalte zu einem einzelnen Wert und über alle Spalte hinweg zu dem korrespondierenden Bitletvektor bzw. Wiederholungsmuster. Beispielsweise ergibt sich bei der Multiplikation des Vektors $x = (x_0, x_1, x_2)$ mit der 1-Spalte einer Matrix B_3, d.h. $v = x \cdot B_3$ die Koordinate v_0 von $v = (v_0, v_1, v_2)$ durch $v_0 = x_0 \cdot B_3(1,1) + x_1 \cdot B_3(2,1) + x_2 \cdot B_3(2,1)$. Offensichtlich sind pro Spalte nur die Nicht-Null-Belegungen der Matrix relevant, da Null-Belegungen keinen Einfluss auf das Ergebnis haben.

Die Grundidee zur effizienten Realisierung Transformation basiert auf der Beobachtung, dass zum einen Standardbitletbasen dünnbesetzt sind (d.h. nur wenige Einsen enthalten) und zum anderen die Matrizen stets nur Einsen oder Nullen enthalten. Für die

Matrixmultiplikation folgt hieraus, dass lediglich die Multiplikation mit Null vermieden werden braucht, um die Effizienz der Transformation zu steigern. Das heißt, wir konzentrieren uns bei den Berechnungen nur auf die Nicht-Null-Elemente der Matrix. Daraus folgt, dass es ausreicht, wenn wir eine Matrix "simulieren", in dem wir eine Funktion bereit stellen, die für eine Spalte dessen Nicht-Null-Belegungen ermittelt. Im Allgemeinen besteht an dieser Stelle die Schwierigkeit, die entsprechende Funktion zu beschreiben. Im Zusammenhang mit Standardbitletbasen erweist sich dieses Problem jedoch als effizient lösbar. Die Konstruktion der Basen bzw. ihre Induktivität führt zu Basismatrizen, die eine stark gleichbleibende Struktur aufweisen. Diese Struktur, die sich aus der Konstruktion der Matrizen aus Matrizen von Teilerräumen ergibt, erlaubt es, die Nicht-Null-Belegungen der Matrix bzw. einer Spalte j effizient zu ermitteln. Wie wir später zeigen werden, reduziert diese "Konzentration auf die Nicht-Null-Elemente" die Worst-Case-Komplexität der Transformation auf $O(m \log(m)^2)$ im Gegensatz zu den ursprünglichen $O(m^2)$.

6.3.2 Realisierung der inversen Fast-Bitlet-Transformation

Wir beginnen unsere Betrachtungen mit der Realisierung der inversen Fast-Bitlet-Transformation $r = v \cdot B_m$, d.h. mit der Überführung einer Bitletrepräsentation in den Zeitbereich. Im Gegensatz zur Fast-Bitlet-Transformation bleiben hier die Basismatrizen in ihrer ursprünglichen Form erhalten.

6.3.2.1 Berechnung der Nicht-Null-Belegung einer Matrixspalte

Die Grundidee zeigt, dass es ausreichend ist, pro Matrixspalte lediglich die Nicht-Null-Belegungen bei der Transformation zu berücksichtigen bzw. eine entsprechende Funktion zur Verfügung stellen, die diese ermittelt. Die Angabe einer konkreten Berechnungsvorschrift für diese Funktion ist jedoch abhängig von der Matrix bzw. der entsprechenden Bitletbasis. Wie zuvor erwähnt, konzentrieren wir uns auf Standardbitletbasen. Aufgrund der benötigten Eigenschaften der Induktivität und der damit vorgestellten Konstruktion weisen Bitletbasen bzw. Standardbitletbasen eine spezifische Strukturierung auf. Diese Strukturierung bildet die Grundlage für die Bestimmung einer entsprechenden Berechnungsvorschrift für die Matrizen.

Struktur von Bitletbasen: Ausgangspunkt der Berechnungsvorschrift für Standardbitletbasen bildet die Betrachtung der Struktur von Standardbitletbasen. Im Folgenden ist diese Struktur für die Standardbitletbasis B_6 dargestellt:

$$B_6 = \left(\begin{array}{cccccc|l} \mathbf{1} & 1 & 1 & 1 & 1 & 1 & B'_1 \\ \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 & B'_2 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 & B'_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 & \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{E_6} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \end{array}\right)$$

$$B_3 = \left(\begin{array}{ccc|l} 1 & 1 & 1 & B'_1 \\ \hline 0 & 1 & 0 & E_3 \\ 0 & 0 & 1 & \end{array}\right) \quad B_2 = \left(\begin{array}{cc|l} 1 & 1 & B'_1 \\ \hline 0 & 1 & E_2 \end{array}\right) \quad B_1 = \left(\begin{array}{c} 1 \end{array}\right)$$

Deutlich zu erkennen ist, dass die Konstruktion von Bitletbasen zu einer Blockstrukturierung der Basismatrix führt. Jede Basis B_m besteht aus einem Ergänzungsbereich E_m, der die bei der Konstruktion der Basis B_m hinzugefügten Bitlets enthält. Bei der Standardbitletbasis sind dies Standardvektoren von rechts. Die Blöcke B_i einer Basis B_m wiederum, enthalten die Basisvektoren der Teilerräume von B_m, d.h. jeder Teilerblock B'_i korrespondiert mit einem Teiler der Basis $B_{i|m}$. Die Teilerblöcke B'_i einer Basis B_m enthalten die periodisch verlängerten Bitlets aus dem Ergänzungsbereich E_i der Basis B_i. Dies ergibt sich durch die Konstruktionsvorschrift von B_m, bei der pro Schritt (rekursiv) stets nur (periodisch verlängerte) Ergänzungsvektoren der Teilerräume auf Basis der Teilermenge $\mathbf{T}_m$ hinzugefügt werden.

Die Bestimmung der Nicht-Null-Belegungen einer Spalte innerhalb des Ergänzungsblock E_m der Basis B_m erweist sich als recht einfach und wird im Folgenden vorgestellt.

Bestimmung der Nicht-Null-Belegungen im Ergänzungsbereich: Zur Bestimmung der Nicht-Null-Belegungen einer Spalte innerhalb eines Ergänzungsblocks E_m nehmen wir zunächst an, dass die Anzahl der Vektoren im Ergänzungsblock E_m (a_m) bekannt ist. Bei der Standardbitletbasis werden zur Vervollständigung der Basis B_m bzw. in dem entsprechenden Ergänzungsbereich E_m nur Standardvektoren verwendet. Für jeden einzufügenden Standardvektor wandert hierbei die Postion der einzigen "1" schrittweise nach links (siehe Abschnitt 6.2.5.2). Daraus folgt jedoch, dass lediglich die Spalten beginnend bei m bis zur Spalte $m-(a_m-1)$ im Ergänzungsbereich eine "1" aufweisen. Alle anderen Spalten, die nicht in diesem Abschnitt liegen, d.h. kleiner als $m-(a_m-1)$ sind, können im Ergänzungsbereich E_m keine Einsen aufweisen. Im Folgenden ist dies für die Basismatrix B_6 dargestellt. Der Ergänzungsbereich E_6 weist zwei Ergänzungsvektoren ($a_6 = 2$) auf, wodurch nur die Spalten j im Abschnitt $5 \leq j \leq 6$ eine Nicht-Null-Belegung aufweisen.

$$B_6 = \left(\begin{array}{cccccc|l} & & & \vdots & & & B_{\mathbf{T}_6} \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 & E_6 \end{array}\right)$$

Zur Überprüfung der Nicht-Null-Belegung für eine Spalte j muss lediglich überprüft werden, ob diese im entsprechenden Abschnitt im Ergänzungsbereich E_m liegt. Nur wenn die Spalte im entsprechenden Abschnitt liegt, gibt es genau ein Bitlet, das für die Spalte eine Nicht-Null-Belegung in E_m aufweist.

Für das obige Vorgehen ist es essentiell, dass die Anzahl der Vektoren im Ergänzungsblock E_m bekannt ist. Diese Anzahl kann wie folgt berechnet werden:

Definition 6.9 (Anzahl der Ergänzungsvektoren): *Sei B_m eine Bitletbasis mit der Teilermenge $\mathbf{T}_m = (t_1, ..., t_k)$. Die Anzahl $a(m)$ der Ergänzungsvektoren der Bitletbasis B_m ergibt sich, wie folgt: $a(m) = m - \sum_{j=1}^{k} a(t_j)$*

Die Bestimmung der Anzahl der Basisvektoren im Bereich E_m der Basis B_m erfolgt rekursiv, in dem für jeden Teilerraum von B_m die Anzahl der Ergänzungsvektoren ermittelt wird. Die Dimension m abzüglich aller Ergänzungsvektoren der anderen Teilerräume ergibt die Anzahl der Ergänzungsvektoren in E_m.

Mit der Anzahl der Vektoren im Ergänzungsbereich E_m kann leicht überprüft werden, ob eine Spalte im entsprechenden Abschnitt im Ergänzungsbereich liegt, der eine Nicht-Null-Belegung aufweist. Ist eine Spalte in dem entsprechenden Abschnitt, stellt die Überprüfung jedoch nur den ersten Schritt dar. Im zweiten Schritt muss das konkrete Bitlet ermittelt werden, das die Nicht-Null-Belegung in der Spalte enthält. Auch hier gestaltet sich das für den Ergänzungsbereich und unter Verwendung der Standardbitletbasis als einfach.

Wie in Abschnitt 6.2.7.2 gezeigt, wird jedes Bitlet $b_{m,j}$ der Standardbitletbasis mit den Parametern Länge des Bitlets in seiner Urform (m) und Position der einzigen Eins (j) eindeutig identifiziert und bei der Bitletrepräsentation indiziert. Im Hinblick auf den Ergänzungsbereich E_m handelt es sich bei den Bitlets um deren Urform, da diese ja zur Vervollständigung der Basis B_m hinzugefügt werden und nicht periodisch verlängert sind. Bei der Konstruktion der Basis werden diese Basisvektoren von "hinten" beginnend zur Vervollständigung der Basis aufgefüllt. Daraus folgt jedoch, dass bei Vektoren im Ergänzungsbereich bei Standardbitletbasen die Position der einzigen Eins mit der Spalte korreliert. Das heißt für E_6 etwa, dass die sechste Spalte mit dem Bitlet korreliert, das an der sechsten Position eine Eins aufweist, d.h. mit den Bitlet $b_{6,6}$. Somit kann jedoch leicht eine Funktion definiert werden, die im Ergänzungsbereich einer Bitletbasis, das Bitlet ermittelt, das bezüglich einer Spalte j, eine Nicht-Null-Belegung aufweist. Diese Funktion ist wie folgt definiert:

Definition 6.10 (Relevante Bitlets im Ergänzungsbereich): *Sei B_m eine Standardbitletbasis und $a(m)$ die Anzahl seiner Ergänzungsvektoren. Die Berechnung der relevanten Bitlets für die Spalte j im Ergänzungsbereich von B_m ergibt sich, wie folgt: $b_j^{(m)} = b_{m,j} \iff j > m - a(m)$.*

Zur Bestimmung der für eine Spalte j relevanten Bitlets wird lediglich überprüft, ob die Spalte in dem Abschnitt im Ergänzungsbereich enthalten ist, der eine Nicht-Null-Belegungen aufweist (siehe oben). Ist dies der Fall, kann unmittelbar das zugehörige Bitlet ermittelt werden, da die Position der einzigen Eins im Bitlet im Ergänzungsbereich stets direkt mit der Spalte j korreliert.

Im anderen Fall, d.h. die Spalte liegt nicht in dem entsprechenden Abschnitt des Ergänzungsbereichs, wird ein anderes Vorgehen benötigt, was im Folgenden vorgestellt wird.

Ermittlung der Nicht-Null-Belegungen der Bitletmatrix: In den vorherigen Betrachtungen haben wir uns auf den Ergänzungsbereich E_m einer Bitletmatrix B_m konzentriert. Wie gezeigt, ist die Bestimmung der Nicht-Null-Belegung bzw. des korrespondierenden Bitlets für eine Spalte j im Ergänzungsbereich relativ einfach. Zur Ermittlung aller Nicht-Null-Belegungen für eine Spalte j muss jedoch auch die restliche Matrix "oberhalb" des Ergänzungsbereichs berücksichtigt werden. Hierzu müssen die Nicht-Null-Belegungen bzw. die entsprechenden Bitlets der Teilerblöcke B'_i der Matrix B_m bestimmt werden.

Wie im vorherigen Abschnitt gezeigt, enthält ein Teilerblock B'_i der Matrix B_m lediglich die Ergänzungsvektoren E_i der Matrix B_i. Somit kann jeder Teilerblock als der Ergänzungsbereich der Matrix B_i verstanden werden kann, wenn auch in periodisch verlängerter Variante. Daraus folgt jedoch, dass das bisherige Vorgehen zur Ermittlung der Nicht-Null-Belegung im Ergänzungsbereich in analoger Form auf die Teilerblöcke übertragen werden kann. Pro Teilerblock muss hierzu lediglich die absolute Spalte j der Matrix B_m in die entsprechende Spalte in dem Teilerblock B_i überführt werden. Dies ist jedoch leicht über eine entsprechende Modulo-Operation $j \bmod i$ realisierbar. So entspricht die Spalte 4 in der Basis B_6 der Spalte 1 in dem Teilerblock B'_3 bzw. in der Matrix B_3 (siehe unten).

$B_6 =$

1	1	1	1	1	1	B'_1
0	**1**	0	1	0	1	B'_2
0	**1**	**0**	0	1	0	B'_3
0	**0**	**1**	0	0	1	
0	**0**	**0**	**0**	**1**	**0**	$\boldsymbol{E_6}$
0	**0**	**0**	**0**	**0**	**1**	

Auf Basis dieser Überlegungen kann eine Funktion angegeben werden, die für eine Spalte j alle Nicht-Null-Belegungen bzw. der relevanten Bitlets der Matrix B_m für diese Spalte ermittelt. Diese Funktion ist wie folgt definiert:

Definition 6.11 (Relevante Bitlets der Standardbitletmatrix): *Sei B_n eine Standardbitletmatrix mit der Teilermenge $\mathbf{T}_n = (t_1, ..., t_k)$. Die für eine Spalte j relevanten Bitlets von B_n werden wie folgt ermittelt:* $B_n(j) = \{b^{(t_1)}_{j \bmod t_1}, ..., b^{(t_k)}_{j \bmod t_k}, b^{(n)}_j\}$.

Die Bestimmung der relevanten Bitlets einer Spalte j für eine Bitletmatrix B_m basiert auf der Teilermenge $\mathbf{T}_m$ für die Dimension der Basis B_m. Über diese Teilermenge werden alle Teilerblöcke von B_m bestimmt und für diese die relevanten Bitlets innerhalb des Ergänzungsbereichs ermittelt. Im Ergebnis liefert diese Funktion eine Menge von Bitlets, die in der Spalte j eine Nicht-Null-Belegung aufweisen. Beispielsweise führt die Funktion für die Spalte $j = 5$ der Bitletmatrix B_6 zu der Ergebnismenge $B_6(5) = \{b_{1,1}, b_{3,2}, b_{6,5}\}$.

Mit der Bestimmung der relevanten Bitlets ist der erste Schritt hin zu einer effizienten Bitlet-Transformation getan. Im nächsten Schritt werden wir zeigen, wie basierend hierauf, effizient einen Bitletvektor in ein Zeitvektor transformiert werden kann.

6.3.2.2 Realisierung der Transformation

Die oben vorgestellte Funktion zur Ermittlung der relevanten Bitlets einer Standardbitletbasis für eine Spalte j bildet die Grundlage für die Transformation einer Bitletrepräsentation v in den korrespondierenden Zeitvektor r, d.h. $r = v \cdot B_m$. Theoretisch müssten für die Transformation die folgenden Gleichungen berechnet werden:

$$\begin{aligned} r_1 &= v_1 \cdot B_m(1,1) + v_2 \cdot B_m(2,1) + \cdots + v_m \cdot B_m(m,1) \\ r_2 &= v_1 \cdot B_m(1,2) + v_2 \cdot B_m(2,2) + \cdots + v_m \cdot B_m(m,2) \\ &\vdots \\ r_m &= v_1 \cdot B_m(1,m) + v_2 \cdot B_m(2,m) + \cdots + v_m \cdot B_m(m,m) \end{aligned}$$

Allerdings haben wir bei der Grundidee zur Fast-Bitlet-Transformation bereits vorgestellt, dass es aufgrund der dünnbesetzten Matrizen deutlich effizienter ist, die Berechnungen, in denen die Matrixbelegung oder der Bitletvektor eine 0 aufweist, zu vermeiden. Das heißt, wir konzentrieren uns bei der Transformation auf die Nicht-Null-Belegungen. Hierzu kann mit der oben vorgestellten Funktion jede Spalte j der Matrix B_m die Menge der relevanten Bitlets ermittelt werden, d.h. die Bitlets, die eine Eins an der Stelle j aufweisen. Diese Menge kann dann mit der Menge B' der Bitlets der entsprechenden Bitletrepräsentation v verglichen werden, d.h. mit der Menge $[m, B' = \{(b_{h,l}, z_1), ..., (b_{k,g}, z_n)\}]$. Für die Ermittlung der Belegung r_j im Zeitvektor r müssen dann lediglich, die Tupel $(b_{h,g}, z_i)$ bzw. der Wert z_i aufsummiert werden, die in der Menge der relevanten Bitlets enthalten sind. Algorithmus 6.1 zeigt dieses Vorgehen im Detail.

Algorithmus 6.1 $iFBT(v) \mapsto r$

input:
v: Standardbitletrepräsentation $v = [n, B' = \{(b_{h,l}, z_1), ..., (b_{k,g}, z_n)\}]$
output:
r: Wiederholungsmuster im Zeitbereich mit $r \in V^n$

```
1: for i = 0 to n do
2:   for all b_{a,b} ∈ B_n(i) do
3:     if ∃(b_{a,b}, z) ∈ B' then
4:       r[i]=r[i]+z
5:     end if
6:   end for
7: end for
8: return r
```

6.3.2.3 Komplexitätsabschätzung der Transformation

Die (inverse) Fast-Bitlet-Transformation ist für eine beliebige Bitletbasis mit Dimension n wesentlich von der Anzahl der Teiler von n abhängig. Das heißt, bei der Ermittlung der relevanten Bitlets der Standardbitletmatrix wird die Teilermenge $\mathbf{T}_n$ ermittelt und für jeden Teiler bzw. den entsprechenden Ergänzungsbereich die relevanten Bitlets ermittelt. Je mehr Elemente die Teilermenge aufweist, umso mehr Ergänzungsbereiche in der Matrix existieren und umso häufiger wird die Funktion $b_j^{(m)}$ pro Matrixspalte aufgerufen, um die relevanten Bitlets der Ergänzungsbereiche zu ermitteln (siehe Definition 6.10).[7]

Im Allgemeinen gilt für die Anzahl der Teiler bzw. den Umfang der Teilermenge T_n einer beliebigen Zahl n folgendes:

Satz 6.3: *Die Anzahl k der Teiler in T_n ist $o(n^\epsilon)$ für jedes $\epsilon > 0$, und im Mittel $O(\log(n))$.*

Beweis: *Die Anzahl $d(n)$ der Teiler von n, erfüllt die Abschätzung $d(n) = o(n^\epsilon)$ für jedes $\epsilon > 0$ sowie die Mittelung*

$$\sum_{n \leq x} d(n) \;=\; x \log(x) + O(x)$$

laut [42]. ∎

[7] Wir vernachlässigen an dieser Stelle die Komplexität der Funktion $a(n)$, d.h. die Funktion, die die Anzahl der Ergänzungsvektoren ermittelt. Anstelle die Funktion $a(n)$ (rekursiv) für jede Basis der Teilerräume aufzurufen, kann auch eine vorausberechnete Liste verwendet werden, wie etwa $A_n = (a(t_1), ..., a(t_k), a(n))$ mit $\mathbf{T}_n = (t_1, ..., t_k)$. Mit einem Zugriff $a(t_i)$ in die Liste A_n erhält man die Anzahl der Ergänzungsvektoren für B_{t_i}. Dies beschränkt den Aufwand zur Ermittlung der Anzahl an an Ergänzungsvektoren in $b_j^{(n)}$ pro Basis B_i auf die Konstante c.

Aus dem obigen Satz folgt, dass jede Bitletmatrix B_n im Mittel nur $\log(n)$-viele Teiler aufweist bzw. Teilerräume enthält. Pro Spalte einer Bitletmatrix B_n sind im Mittel somit nur $O(\log(n))$ Aufrufe der Funktion $b_j^{(m)}$ notwendig, um die relevanten Bitlets im Ergänzungsbereich zu ermitteln. Ein Aufruf der Funktion für eine Spalte j bezüglich des Ergänzungsbereichs der Matrix B_m liefert jedoch maximal nur ein relevantes Bitlet oder keins (siehe Abschnitt 6.3.2.1). Dies wiederum bedeutet, dass eine Spalte in der Bitletmatrix im Mittel nur $\log(n)$ Nicht-Null-Belegungen aufweist, da im Mittel für eine Bitletmatrix B_n nur $\log(n)$-viele Teilerräume mit jeweils einem Ergänzungsbereich existieren. Daraus folgt jedoch, dass die Transformation eines Bitletvektors in den Zeitbereich auf Basis der vorgestellten inversen Fast-Transformation die folgende Worst-Case Komplexität aufweist: $O(n \log(n)^2)$.

Jede Spalte weist $log(n)$-viele Nicht-Null-Belegungen aus, die über alle Spalten der Matrix B_n bei der Transformation zu $log(n) \cdot log(n) = log(n)^2$-vielen Termen führen. Im Worst-Case wird des Weiteren davon ausgegangen, dass in einer entsprechenden Bitletrepräsentation jedes Bitlet der Matrix B_n enthalten ist, was dem n vor dem Term $log(n)^2$ entspricht und zu der dargestellten Komplexität führt. Wenn wir jedoch davon ausgehen, dass unsere Bitletrepräsentation dünnbesetzt ist, d.h. nur wenige Nicht-Null-Koordinaten aufweist, dann verringert sich der Wert für n und die inverse Fast-Bitlet-Transformation wird im Mittel ein noch weitaus besseres Laufzeitverhalten aufweisen. Unabhängig davon, erweist sich das vorgestellte Verfahren mit seiner Worst-Case Komplexität $O(n \log(n)^2)$ gegenüber der naiven Matrizenmultiplikation mit $O(n^2)$ als erheblich effizienter.

Die prototypische Implementierung der Fast-Bitlet-Transformation bestätigt die obige theoretische Komplexitätsabschätzung (siehe Kapitel 7). Neben der vorgestellten Transformation von Bitletvektoren in Zeitvektoren (und anders herum) erlaubt der Prototyp es auch, das geometrische Mittel der relevanten Bitlets, d.h. der Nicht-Null-Belegungen über alle Spalten von Standardbitletmatrizen zu berechnen. In Abbildung 6.8 ist das geometrische Mittel für Standardbitletmatrizen verschiedener Dimension dargestellt. Die obige Abschätzung, dass jede Spalte einer Matrix B_n $log(n)$-viele Nicht-Null-Belegungen aufweist, wird durch die dargestellten Beispielen bestätigt.[8]

6.3.3 Realisierung der Fast-Bitlet-Transformation

Die in den vorherigen Abschnitten vorgestellte (inverse) Fast-Bitlet-Transformation, profitiert von der Tatsache, dass die Eigenschaft der Standardbitlebasis sehr gut bekannt ist und einfach beschrieben werden konnte. Insbesondere die induktive Eigenschaft, dass sich große Matrizen aus verlängerten Basisvektoren bzw. Zeilen kleiner Matrizen zusammensetzen, macht es einfach, die Position der Nicht-Null-Belegungen zu identifizieren.

[8] Der dargestellte Berechnungsaufwand (in ms) steht für die Berechnung des geometrischen Mittels und nicht für die Bestimmung der relevanten Bitlets bzw. Nicht-Null-Belegungen.

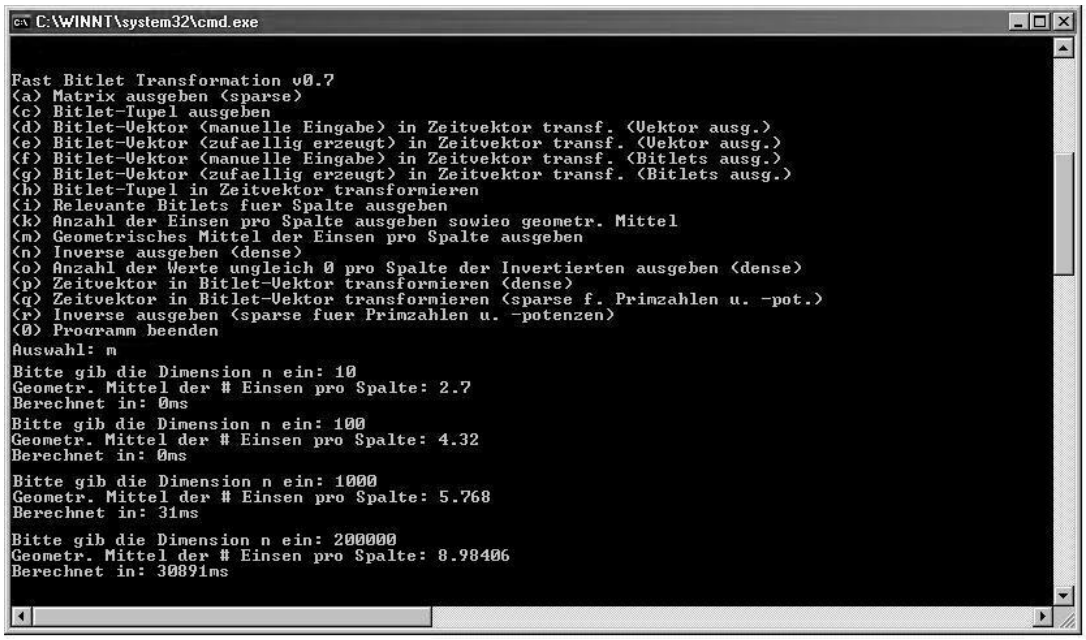


Abbildung 6.8: Ermittlung des geometrischen Mittel an Nicht-Null-Belegungen pro Spalte bei Standardbitletbasen verschiedener Dimension durch die prototypische Realisierung der Fast-Bitlet-Transformation

Für die Realisierung der Fast-Bitlet-Transformation gilt es jedoch, die invertierten Standardbitlebasen zu betrachten bzw. für diese die Positionen der Nicht-Null-Belegung zu identifizieren. Im Folgenden sind einige Beispiele für inverse Standardbitlebasen B_i^{-1} dargestellt.

$$B_1^{-1} = (1) \quad B_2^{-1} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \quad B_3^{-1} = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$B_4^{-1} = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad B_5^{-1} = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Wie gut zu erkennen ist, verhält sich die inverse Standardbitletmatrix nicht sehr "umgänglich". Genauer gesagt, die induktive Eigenschaft, dass große Matrizen sich unmittelbar aus periodisch verlängerten Basisvektoren kleinerer Matrizen zusammensetzen, gilt hier nicht mehr. Am Beispiel der Standardbitlebasis B_6 bzw. B_6^{-1} ist dies im Folgenden dargestellt.

$$B_6 = \begin{pmatrix} \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \qquad B_6^{-1} = \begin{pmatrix} 1 & -1 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In B_6 entspricht etwa die zweite Zeile dem Basisvektor von B_2 in periodisch verlängerter Version. In B_6^{-1} hingegen, ist dies leider nicht mehr so gegeben, d.h. die zweite Zeile in B_6^{-1} weist wenig Ähnlichkeiten mit der zweiten Zeile in B_2^{-1} auf. Zusätzlich verändert sich diese Zeile auch noch in Abhängigkeit von der Raumdimension. Das heißt, die zweite Zeile bzw. die Basis B_2 ist nach Konstruktionsvorschrift für Standardbitletbasen auch Bestandteil der Matrizen B_4 und B_6. Bei den inversen Matrizen zeigt sich jedoch, dass weder in B_4^{-1} noch in B_6^{-1} eine direkte Ähnlichkeit von (inversen) Bitlets untereinander noch zu dem Urspungsbitlet in B_2^{-1} besteht. Dies ist auch gut zu erkennen anhand der ersten Zeile in den inversen Matrizen. In Standardbitletbasen entspricht diese dem Teilleraum V^1 bzw. dessen Basis B_1. In keiner der inversen Matrizen bleibt jedoch diese Zeile gleich oder kann als periodische Verlängerung verstanden werden.

Zusammenfassend betrachtet, ist eine Übertragung des im vorherigen Abschnitt vorgestellten Vorgehens zur Realisierung einer "Fast-Bitlet-Transformation" nicht möglich. Das Verhalten der bei der Invertierung von Standardbitletbasen ergebenen Matrizen ist zu "unregelmäßig" für das einfache und effiziente Verfahren der inversen Fast-Bitlet-Transformation. Es ist an dieser Stelle unklar, ob für inverse Matrizen bzw. inverse Standardbitletbasen entsprechende Regelmäßigkeiten identifiziert werden können. In dieser Arbeit werden hierzu keine weitergehenden Betrachtungen angestellt. Wir überlassen es zukünftigen Arbeiten ein effizienteres Verfahren im Vergleich zu einer expliziten Matrixmultiplikation zu entwickeln, für die Transformation von Wiederholungsmustern in den Bitletbereich. Stattdessen werden im Folgenden die generellen Eigenschaften des Bitletbereichs bei der Realisierung von Vergleichsoperationen im Fokus der Betrachtungen stehen.

6.4 Realisierung von Vergleichsoperationen im Bitletbereich

In Hinblick auf die operationale Unterstützung periodischer Spezifikationen zeigt sich, dass neben den Ableitungs- und Überprüfungsoperationen insbesondere die Vergleichsoperationen eine der größten Herausforderungen darstellt. Bisherige Ansätze zur Realisierung insbesondere von Vergleichsoperationen auf Basis von Wiederholungsmustern sind aufgrund ihres Repräsentationsumfangs und dem damit verbundenen operationalen Aufwand nicht für die Praxis geeignet (siehe Kapitel 2). Wie in Abschnitt 6.1 gezeigt, stellt die notwendige Verlängerung unterschiedlich langer Muster und der damit verbundene Anstieg des Repräsentationsumfangs das entscheidende Problem dar. Mit der in den vorherigen Abschnitten eingeführten Bitletrepräsentation besteht eine Repräsentationform, das dieses Problem nicht mehr aufweist. Das heißt, die Anpassung unterschiedlich langer Muster führt in der Bitletrepräsentation zu keinem Anstieg des Repräsentationsumfangs.

Wir werden im Folgenden die Realisierung von Vergleichsoperationen im Bitletbereich vorstellen, welche von der positiven Eigenschaft dieser Repräsentationsform profitieren. Auch wenn im Folgenden der Fokus lediglich auf der Standardbitletbasis liegt, so sind die vorgestellten Methoden zur Realisierung von Vergleichsoperationen allgemeingültig. Das heißt, bei anderen Bitletbasen als der Standardbitletbasis kann zur Realisierung von Vergleichen (weitestgehend) in analoger Weise verfahren werden.

Im folgenden Abschnitt werden zunächst tiefergehende Betrachtungen zur allgemeinen Realisierung von Vergleichsoperationen vorgenommen.

6.4.1 Vorbetrachtungen zu Vergleichsoperationen

Die Abbildung der Semantik von Vergleichsoperationen kann bei der Realisierung auf Basis von Wiederholungsmustern im Zeitbereich als Anwendung korrespondierender boolescher Funktionen zwischen den einzelnen Positionen bzw. Stützstellen zweier Muster verstanden werden. Abbildung 6.9 zeigt dies für die Mengenoperation $\cap$ zwischen den beiden Beispielmustern s_1 und s_2. Die Realisierung der Operation $\cap$ kann als Anwendung der booleschen Funktion $\wedge$ zwischen den jeweiligen Musterpositionen verstanden werden. Die Repräsentationsform des Musters bzw. des Vektors, d.h. dichtbesetzte oder dünnbesetzte Repräsentationsform, spielt bei der Realisierung keine Rolle. Während bei der dichtbesetzten Repräsentation alle Positionen unabhängig von der konkreten Belegung paarweise bzgl. der Basisfunktion miteinander verglichen werden, erfolgt dies bei der dünnbesetzten nur für die Positionen mit mindestens einer Nicht-Null-Belegung.

Die Verwendung etwa der booleschen Funktion $\wedge$ zur Realisierung der Vergleichsoperation $\cap$ ist jedoch nicht zwingend. Wie aus der allgemeinen Theorie zu Booleschen Funktionen bekannt, kann die Semantik etwa der Funktion $\wedge$ durch andere boolesche Funktion "nachgebildet" werden. So ist es beispielsweise mit der Operationsmenge $\{\vee, \neg\}$ möglich, die Operation $\wedge$ nachzubilden mit $x \wedge y \equiv \neg(\neg x \vee \neg y)$.

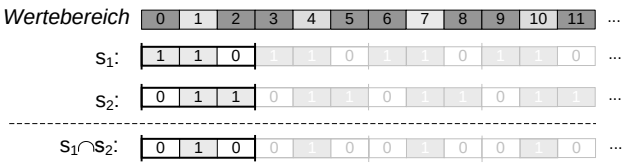


Abbildung 6.9: Schnittoperation von Wiederholungsmustern mit gleicher Periodenlänge

Die Eigenschaft der Operationsmenge $\{\vee, \neg\}$ zur Nachbildung wird als vollständig bezeichnet. Allgemeiner ausgedrückt, jede Operationsmenge von booleschen Funktionen, die es erlaubt, die beiden Operationen $\vee$ und $\wedge$ nachzubilden, wird als *vollständig* bezeichnet und erlaubt es, beliebige aussagenlogische Zusammenhänge zu definieren. Insgesamt stehen 16 verschiedene (zweistellige) boolesche Funktionen zur Verfügung. Tabelle 6.2 zeigt diese in Übersicht. So entspricht etwa die Funktion $F_2 = \vee$, $F_8 = \wedge$ oder $F_{10} = XOR(\oplus)$. Die Funktionen F_2, F_3, F_5, F_8, F_{12}, F_{14} sind vollständig unter der Negation $\neg$, d.h. jede dieser Funktionen F ist mit $\{F, \neg\}$ vollständig [76]. Lediglich die Funktionen F_9 und F_{15} sind *minimal vollständig* und benötigen keine Negation. In Abbildung 6.2 sind die (minimal) vollständigen Funktionen mit einem Punkt gekennzeichnet. Alle anderen Funktionen sind nicht vollständig, d.h. sie können die Funktionen $\wedge$ und $\vee$ nicht nachbilden.

Übertragen auf Vergleichsoperationen bedeutet dies, dass bei der musterbasierten Realisierung von Vergleichsoperationen lediglich eine vollständige boolesche Funktion bzw. Funktionsmenge benötigt wird. Welche Funktion dies genau ist, spielt keine Rolle, da eine vollständige Funktion die Funktionen $\wedge$ und $\vee$ nachbilden kann und somit auch die Semantik der Vergleichsoperationen $\{\cap, \cup, \neg\}$. Diese Menge an Vergleichsoperationen kann ebenfalls als vollständig erachtet werden, d.h. sie erlaubt es, beliebige mengentheoretische Zusammenhänge zu formulieren.

Im Folgenden werden wir die Realisierung von Vergleichsoperationen bzw. ihrer zugrundeliegenden booleschen Funktionen im Zusammenhang mit der Bitletrepräsentation vorstellen.

x	y	F_1	$\dot{F}_2$	$\dot{F}_3$	F_4	$\dot{F}_5$	F_6	F_7	$\dot{F}_8$	$\dot{F}_9$	F_{10}	F_{11}	$\dot{F}_{12}$	F_{13}	$\dot{F}_{14}$	$\dot{F}_{15}$	F_{16}
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Tabelle 6.2: Gesamtmenge zweistelliger Boolescher Funktionen

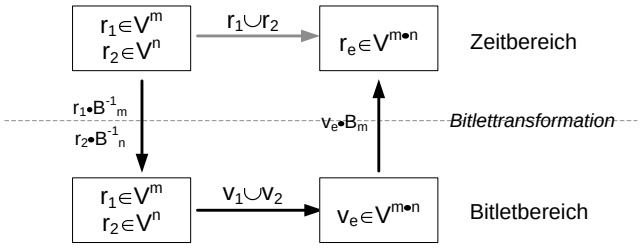


Abbildung 6.10: Auswertungsdiagramm der bitletbasierten Realisierung von Vergleichsoperationen

6.4.2 Grundidee der bitletbasierten Realisierung

Die Grundidee der bitletbasierten Realisierung von Vergleichsoperationen besteht darin, dass die Operationen direkt im Bitletbereich bzw. auf der Bitletrepräsentation von Wiederholungsmustern durchgeführt werden können. Das heißt, das Ergebnis der Vergleichsoperation ist eine Bitletrepräsentation, die nach Transformation zu dem gleichen Ergebnismuster führt, als ob die Vergleichsoperation im Zeitbereich durchgeführt wurden wäre. Abbildung 6.10 zeigt dies anhand eines entsprechenden Auswertungsdiagramms. Aufgrund der effizienten Raumeinbettung durch die Bitletrepräsentation würde sich der "untere" Weg über den Bitletbereich als deutlich effizienter erweisen, als der "obere" Weg über den Zeitbereich und die aufwändige Vereinheitlichung unterschiedlich langer Muster.

Die bitletbasierte Realisierung führt in ihrer Grundidee zu einer orthogonalen Realisierung von Vergleichsoperationen, d.h. jedes Ergebnis einer Vergleichoperation stellt erneut eine Bitletrepräsentation dar. Das Ergebnis kann dadurch in weiteren Vergleichsoperationen im Bitletbereich erneut verwendet werden. Eine Rücktransformation in Zeitbereich bzw. in das korrespondierende Wiederholungsmuster braucht somit nur wenn gewünscht stattfinden.

In der Grundidee zur bitletbasierten Realisierung wird indirekt davon ausgegangen, dass die Vergleichsoperation bzw. deren boolesche Basisfunktionen im Bitletbereich analog wie im Zeitbereich verwendet werden können. Allerdings handelt es sich bei dem Bitletbereich bzw. bei der entsprechende Bitlettransformation um eine lineare Abbildung zwischen zwei Vektorräumen. Wie in Abschnitt 6.2.2 gezeigt, zeichnen sich lineare Abbildungen dadurch aus, dass diese verträglich mit den Operationen Addition und skalare Multiplikation sind. Hierbei bedeutet verträglich, dass es keine Rolle spielt, ob wir die Operation im Zeitbereich verwenden oder im Bitletbereich. Das Ergebnis ist nach entsprechender Transformation stets gleich. Für alle anderen Operationen wie etwa Multiplikation ist diese Verträglichkeit per se nicht sichergestellt. Es hängt von der verwendeten Basis des Vektorraums ab.

In Hinblick auf Vergleichsoperationen bzw. die entsprechenden booleschen Basisfunk-

tionen bedeutet dies jedoch auch, dass deren Verträglichkeit mit der Bitlettransformation ebenfalls per se nicht sichergestellt ist. An dieser Stelle gilt es somit zunächst zu überprüfen, ob die booleschen Basisfunktionen verträglich mit der Bitlettransformation sind. Sollte dies der Fall sein, kann die Realisierung der Vergleichsoperation analog zu Wiederholungsmustern im Zeitbereich erfolgen. Im anderen Fall müssen Vergleichsoperationen durch mit der Bitlettransformation verträgliche Operationen realisiert werden. Wir beginnen mit der Überprüfung der Verträglichkeit von booleschen Funktionen mit der Bitlettransformation.

Verträglichkeit von booleschen Basisfunktionen mit der Bitlettransformation: Die Verträglichkeit boolescher Basisfunktionen mit der Bitlettransformation hängt von der verwendeten Bitletbasis ab. In der vorliegenden Arbeit liegt der Fokus auf der Standardbitletbasis, deren Verträglichkeit wir im Folgenden überprüfen werden. Ausgehend von zwei Wiederholungsmustern $r_1 \in V^m$ und $r_2 \in V^n$ müssen wir daher überprüfen, ob die Gleichung $r_1 \otimes r_2 = B_{m \cdot n}(B_m^{-1}(r_1) \otimes B_n^{-1}(r_2))$ gilt. Das heißt, wir müssen überprüfen, ob das Ergebnis der Operation $\otimes$ im Zeitbereich das Gleiche ist, wie nach Transformation der Muster und Ausführung der Operation im Bitletbereich sowie Rücktransformation im Zeitbereich. Diese Überprüfung gestaltet sich insofern als einfach, da ein Gegenbeispiel ausreicht, um die Verträglichkeit solcher Basisfunktionen mit der Bitlettransformation zu widerlegen.

Allerdings gilt es zu beachten, dass die Bitlettransformation bisher stets im Zahlenbereich der ganzen Zahlen $\mathbb{Z}$ betrachtet wurde, d.h. es wurde stets von einem $\mathbb{Z}$-Vektorraum ausgegangen. Boolesche Basisfunktionen hingegen sind nicht für den Zahlenbereich der ganzen Zahlen definiert, sondern nur für die Wahrheitswerte $\{0, 1\}$. Um die formale Korrektheit der Überprüfung sicherzustellen, müssen wir an dieser Stelle die Bitlettransformation nicht auf dem Zahlenbereich $\mathbb{Z}$ betrachten, sondern auf den Zahlenbereich $\mathbb{Z}^2$, d.h. einen endlichen Körper, der nur die beiden diskreten Elemente $\{0, 1\}$ enthält.[9] Für die Bitlettransformation spielt es formal keine Rolle, ob wir den Zahlenbereich der ganzen Zahlen $\mathbb{Z}$ oder den Zahlenbereich $\mathbb{Z}^2$ verwenden. Beide Zahlenbereiche führen zu einem K-Vektorraum, für den die Bitlettransformation im Allgemeinen definiert ist.

Bei der Überprüfung der Verträglichkeit konzentrieren wir uns auf die (minimal) vollständigen booleschen Funktionen. Diese Funktionen bilden die "interessanten" Funktionen, da diese, wie zuvor erwähnt, die Semantik der Vergleichsoperationen $\{\cap, \cup, \neg\}$ stets abbilden können. Das Ergebnis der Überprüfung zeigt Tabelle 6.3. Für jede der Funktionen existiert ein Gegenbeispiel, was bedeutet, dass diese Funktionen mit der Standardbitletbasis unverträglich sind. Aus diesem Ergebnis folgt somit auch, dass Vergleichsoperationen im Bitletbereich nicht analog wie im Zeitbereich über boolesche Basisfunktionen realisiert werden können. Stattdessen müssen Vergleichsoperationen durch

[9] Praktisch kann dieser Körper bzw. Zahlenbereich über eine Modulo-Rechnung mit Modul = 2 realisiert werden.

Basisfunktion $\otimes$	r_1	r_2	$r_1 \otimes r_2$	$B(B^{-1}(r_1) \otimes B^{-1}(r_2))$
F_2 ($\vee$)	$(0,0,1)$	$(1,0,0)$	$(1,0,1)$	$(1,0,0)$
F_3	$(0,0,1)$	$(0,1,0)$	$(1,0,1)$	$(1,1,0)$
F_5	$(0,1,1)$	$(0,1,1)$	$(1,1,1)$	$(1,0,0)$
F_8 ($\wedge$)	$(0,1,0)$	$(1,0,0)$	$(0,0,0)$	$(0,1,0)$
F_9	$(0,0,1)$	$(1,1,0)$	$(1,1,1)$	$(1,0,1)$
F_{12}	$(0,1,0)$	$(1,0,0)$	$(0,1,0)$	$(0,0,0)$
F_{14}	$(0,1,1)$	$(1,0,0)$	$(1,0,0)$	$(1,1,1)$
F_{15}	$(1,0,1)$	$(0,1,0)$	$(0,0,0)$	$(0,0,1)$

Tabelle 6.3: Gegenbeispiele für die Verträglichkeit boolescher Funktion mit der Bitlettransformation bezüglich der Standardbitletbasis

mit der Standardbitletbasis verträglichen Operationen realisiert werden, was in den folgenden Abschnitten vorgestellt wird.

Zur Realisierung von Vergleichsoperationen auf Basis verträglicher Operationen stehen mit der Bitlettransformation als lineare Abbildung lediglich die Addition und skalare Multiplikation zur Verfügung. Das heißt, anstatt die Koordinatenpaare von Vektoren mit booleschen Funktionen auszuwerten, erfolgt im Bitletbereich etwa eine Addition der einzelnen Paare. In den folgenden Abschnitten wollen wir die Betrachtungen zur Realisierung von Vergleichsoperationen auf Basis von verträglichen Operationen sowohl für den Zahlenbereich $\mathbb{Z}^2$ als auch für den Bereich $\mathbb{Z}$ vornehmen.

In den folgenden Abschnitten wird zunächst die unbeschränkte Auswertung von Vergleichsoperationen im Vordergrund stehen (Die eingeschränkte Auswertung wird erst in Abschnitt 6.5.2 betrachtet.). Hierzu werden wir zu Beginn lediglich die grundsätzliche Realisierbarkeit von Vergleichsoperationen bzw. zugrundeliegenden (booleschen) Basisfunktionen in den verschiedenene Zahlenbereichen betrachten. Wir gehen dabei von einer "einfachen" (dichtbesetzten) Repräsentation eines Koordinatenvektors im Bitletbereich aus. Erst im Anschluß wird die Realisierung der Vergleichsoperationen im Zusammenhang mit der Bitletrepräsentation vorgestellt, da diese einer dünnbesetzten Repräsentationsform des Koordinatenvektors entspricht.

6.4.3 Vergleichsoperationen im $\mathbb{Z}^2$-Vektorraum

Im $\mathbb{Z}^2$-Vektorraum stehen zur Realisierung von Vergleichsoperationen auf Basis mit der linearen Abbildung verträglicher Operationen effektiv nur die Addition zur Verfügung. Die skalare Multiplikation weist hier keine "brauchbare" Semantik auf, da der Vektor lediglich mit Eins bzw. Null multipliziert werden kann. Dies wiederum führt dazu, dass der Vektor bei Multiplikation mit Eins so bleibt wie er ist bzw. bei Multiplikation mit Null zum Nullvektor wird.

6.4.3.1 XOR-Operation ($\oplus$)

Die Addition zweier Vektoren bzw. die paarweise Addition zweier Koordinaten x, y zweier Vektoren u, v weisen in diesem Vektorraum die folgenden Einzelergebnisse[10] auf:

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	0

In der unteren Zeile sieht man die Besonderheit aufgrund der Verwendung des Zahlenbereichs $\mathbb{Z}^2$, d.h. die Addition von $1+1=0$. Infolgedessen entspricht die Addition im Zahlenbereich $\mathbb{Z}^2$ in seiner Semantik der booleschen Funktion XOR ($\oplus$). Seien u, v Vektoren des $\mathbb{Z}^2$-Vektorraum V^n, dann ist die XOR-Operation wie folgt definiert:

$$u \oplus v = u + v$$

Die XOR-Operation wird im $\mathbb{Z}^2$-Vektorraum durch die Addition bzw. Vektoraddition simuliert. Beispielsweise ist das Ergebnis einer XOR-Operation zwischen den Mustern $r_1 = (0,1,0,1)$ und $r_2 = (1,1,0,0)$ das Muster $r_e = (1,0,0,1)$. Die beiden Muster entsprechen im Bitletbereich den Vektoren $u = (0,1,0,0)$ und $v = (1,0,-1,1)$. Die Vektoraddition von u und v führt zu den Vektor $e = (1,1,-1,1)$, der bei Rücktransformation im Zeitbereich zu dem Vektor $e \cdot B_3 = (1,0,0,1) = r_e$ führt.

6.4.3.2 NOT-Operation ($\neg$)

Zur Realisierung der NOT-Operation ($\neg$), d.h. der Negierung eines Wiederholungsmusters im Zeitbereich, muss im Bitletbereich lediglich jede Koordinate des korrespondierenden Vektors mit Eins addiert werden. Das Aufaddieren der einzelnen Eins muss jedoch für jede Koordinate im Vektor erfolgen, unabhängig, ob es sich um eine Null-Belegung oder Nicht-Null-Belegung handelt. Aufgrund des Zahlenbereichs $\mathbb{Z}^2$ ergeben sich bei der Addition $1+1=0$ bzw. $1+0=1$, was zu einer Negierung der ursprünglichen Koordinate führt, was die folgende Wahrheitstafel zeigt:

x	$x+1$
0	1
1	0

[10] Die Ergebnistafel zeigt auch, dass die Multiplikation in ihrer Semantik der booleschen Funktion AND ($\wedge$) entsprechen würde, da nur $1 \cdot 1 = 1$ eine Belegung ungleich Null aufweisen würde. Jedoch ist die Multiplikation nicht verträglich mit der Standardbitletbasis, da wie zuvor gezeigt auch die AND-Funktion (F_8) nicht verträglich ist (siehe Abbildung 6.3).

Sei u ein Vektor des $\mathbb{Z}^2$-Vektorraum V^n dann ist die NOT-Operation, wie folgt definiert:

$$\boxed{\neg u = u + 1}$$

Im Zusammenhang mit der (dünnbesetzten) Bitlettrepräsentation ergibt sich jedoch das Problem, dass das Aufaddieren der Eins (theoretisch) für jede Koordinate im Vektor bzw. für jedes Bitlet erfolgen muss, auch für solche die eine Null-Belegung aufweisen. Aufgrund der vorgestellten Konstruktionsvorschrift für Bitletbasen, gestaltet sich das Aufaddieren der Eins für aller Koordinaten jedoch als einfach. Aufgrund der Induktivität enthält (per Konstruktion) jede Bitletbasis den Basisvektor des Raums V^1, d.h. den Basisvektor, der nur Einsen enthält. Das Setzen bzw. Löschen der einzelnen Koordinate für diesen Basisvektor führt zu einer Addition einer Eins auf alle Koordinaten im entsprechenden Vektor. Dies ergibt sich zwangsläufig bei der Rücktransformation in den Zeitbereich. Die Berücksichtigung (bzw. Nicht-Berücksichtigung) dieses Basisvektors führt dazu, dass automatisch stets eine zusätzliche Eins für jede Koordinate hinzukommt (bzw. wegfällt). Im Fall, dass die Koordinate vor der Negierung ursprünglich die Belegung $= 1$ aufwies, führt die zusätzliche Addition der Eins bei der Rücktransformation mit $1 + 1 = 0$ dazu, dass sich die Belegung im Zeitbereich negiert. Das Umgekehrte gilt, wenn die Koordinate ursprünglich die Belegungen $= 0$ aufwies. Bei einer Bitletrepräsentation mit der Standardbitletbasis muss folglich das Bitlets $b_{1,1}$ zu einer Bitletrepräsentation hinzugefügt oder gelöscht werden, um die NOT-Operation zu realisieren bzw. das mit der Bitletrepräsentation korrespondierende Wiederholungsmuster im Zeitbereich zu negieren.

6.4.3.3 XNOR-Operation ($\equiv$)

Aufgrund der NOT-Operation und der XOR-Operation existiert in $\mathbb{Z}^2$ auch die Äquivalenz-Operation bzw. XNOR-Operation. Hierzu muss lediglich das Ergebnis der XOR-Operation negiert werden, was die folgende Wahrheitstafel zeigt:

x	y	$\neg(x \oplus y)$
0	0	1
0	1	0
1	0	0
1	1	1

Seien u, v Vektoren des $\mathbb{Z}^2$-Vektorraums V^n, dann ist die XNOR-Operation wie folgt definiert:

$$\boxed{(u \equiv v) = \neg(u \oplus v)}$$

Zusammenfassend betrachtet, stehen im Vektorraum $\mathbb{Z}^2$ lediglich die Funktionsmenge $\{\neg, \oplus, \equiv\}$ im Bitletbereich zur Verfügung. Allerdings stellt diese Funktionsmenge keine vollständige boolesche Funktionsbasis dar. Die Semantik der Vergleichsoperationen $\{\cap, \cup, \neg\}$ kann somit jedoch nicht durch diese Funktionen abgebildet werden.

6.4.4 Vergleichsoperationen im $\mathbb{Z}$-Vektorraum

Im $\mathbb{Z}$-Vektorraum stehen zur Realisierung von Vergleichsoperationen auf Basis mit der linearen Abbildung verträgliche Operationen die Addition und skalare Multiplikation zur Verfügung.

6.4.4.1 OR-Operation ($\cup$)

Die Addition zweier Vektoren bzw. die paarweise Addition zweier Koordinaten x, y des Vektors weisen im $\mathbb{Z}$-Vektorraum die folgenden Einzelergebnisse auf:

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	2

Der Unterschied zum vorherigen Vektorraum zeigt sich erneut in der letzten Zeile, d.h. die Addition von $1 + 1 = 2$ führt zu einem Nicht-Null-Ergebnis. In der Semantik kann diese Additionstabelle als eine OR-Operation verstanden werden. Hierzu muss lediglich die resultierende "2" als eine Nicht-Null-Belegung verstanden werden. In unseren bisherigen Betrachtungen sind wir stets davon ausgegangen, dass mit den Werten "1" bzw. "0" das Enthaltensein bzw. Nicht-Enthaltensein eines Zeitwerts im Muster bzw. Vektor kodiert wird (siehe Abschnitt 6.1). Allerdings ist eine solche Kodierung willkürlich. Grundsätzlich gilt es bei Mustern im Zeitbereich nur zu unterscheiden, ob eine Musterposition eine Null-Belegung (0) oder eine von Null verschiedene Belegung aufweist. Letztere kann durch eine "1" oder durch einen anderen belieben Wert ungleich Null kodiert werden.

Übertragen auf die Situation im $\mathbb{Z}$-Vektorraum bedeutet dies, dass die Addition im Bitletbereich in ihrer Semantik eine OR-Operation darstellt. Alle Kombinationen in denen x oder y oder x und y eine von Null-verschiedene Belegung aufweisen, führt auch das Ergebnis zu einer von Null verschiedenen Belegung. Seien u, v Vektoren des $\mathbb{Z}$-Vektorraum V^n, dann ist die OR-Operation wie folgt definiert:

$$u \vee v = u + v$$

Die OR-Operation wird im $\mathbb{Z}$-Vektorraum durch die Addition bzw. Vektoradddition simuliert. Beispielsweise ist das Ergebnis einer OR-Operation zwischen den Muster $r_1 = (0, 1, 0, 1)$ und $r_2 = (1, 1, 0, 0)$ das Muster $r_e = (1, 1, 0, 1)$. Die beiden Muster entsprechen im Bitletbereich den Vektoren $u = (0, 1, 0, 0)$ und $v = (1, 0, -1, 1)$. Die Addition von u und v führt zu dem Vektor $e = (1, 1, -1, 1)$, der bei Rücktransformation in den Zeitbereich zu dem Vektor $e \cdot B_3 = e' = (1, 2, 0, 3)$ führt. Das konkrete Ergebnis der Addition ist für eine entsprechende Vergleichsoperation irrelevant. Es reicht lediglich zu unterscheiden, ob die Addition zu einer Null-Belegung oder zu einer von Null verschiedenen Belegung führt. Bei der Rücktransformation, etwa in entsprechende Muster im Zeitbereich, kann diese Überlegung übertragen werden. Hierzu muss bei der Rücktransformation lediglich entschieden werden, ob eine Position eine Null-Belegung oder von Null verschiedene Belegung aufweist. Je nachdem, kann dann erneut mit den Werten 0 und 1 im Zeitbereich das entsprechende Muster kodiert werden. Für das vorherige Beispiel bedeutet dies, dass der Vektor $e' = (1, 2, 0, 3)$ bei Kodierung der Nicht-Null-Belegung als "1" zu dem Vektor $(1, 0, 1, 1)$ führt, was wiederum dem Ergebnismuster der OR-Operation r_e im Zeitbereich entspricht.

Realisierung der Vergleichsoperation $\cup$ bei Standardbitletrepräsentation: Die paarweise Addition bildet die Basisfunktion zur Realisierung der Vergleichsoperation $\cup$ im Bitletbereich (siehe Abschnitt 6.4.4.1). Diese Addition, ausgeführt auf allen Koordinaten zwischen zwei Bitletvektoren, führt dann zu einem Koordinatenvektor, der das Ergebnis der OR-Operation darstellt. Im Zusammenhang mit der Bitletrepräsentation müssen jedoch nicht alle Koordinaten der beiden Bitletvektoren berücksichtigt werden. Da es sich hier um eine dünnbesetzte Repräsentationsform handelt, müssen lediglich die Nicht-Null-Koordinaten zur Realisierung der Vergleichsoperation $\cup$ zwischen zwei Bitletrepräsentationen $v_1 = [n, B_1]$ und $v_2 = [m, B_2]$ berücksichtigt werden.

Da die beiden Mengen B_1 und B_2 lediglich die Nicht-Null-Belegungen des Vektors im Bitletbereich enthalten, wird eine "dünnbesetzte" Addition zwischen zwei Vektoren benötigt. Das heißt, für die Tupel $(b_{x,y}, z)$, deren Bitlets $b_{x,y}$ sowohl in B_1 als auch B_2 vorkommen, müssen die jeweiligen Werte addiert werden und anschließend in die Bitletmenge B_e der Ergebnis-Bitletrepräsentation $v_e = v_1 \cup v_2$ übertragen werden. Die Bitlets, die entweder in B_1 oder B_2 vorkommen, müssen lediglich in B_e hinzugefügt werden und stellen quasi jeweils eine Addition mit 0 dar (da die andere Menge das Bitlet ja nicht enthält). In der resultierenden Bitletrepräsentation muss zusätzlich die Dimension des gemeinsamen Vektorraums von V^n und V^m hinterlegt werden. Falls n und m teilerfremd sind, ergibt sich die Bitletrepräsentation $v_e = [m \cdot n, B_3]$, in der B_3 die oben beschriebene Vereinigung der Bitletmengen von v_1 und v_2 darstellt. Im Detail ist die Realisierung der Vergleichsoperation $\cup$ im Algorithmus 6.2 dargestellt.

Zur Realisierung der Vergleichsoperationen $\cup$ wird lediglich die Addition verwendet. Wie in Abschnitt 6.2.2.2 ist diese stets mit der linearen Abbildung verträglich. Somit führt ihre Anwendung im Bitletbereich und nach Transformation in den Zeitbereich zu dem

Algorithmus 6.2 $\cup(\mathbf{v_1}, \mathbf{v_2}) \mapsto \mathbf{v_e}$

Input:
v_1 Standardbitletrepräsentation $v_1 = [n, B_1]$
v_2 Standardbitletrepräsentation $v_2 = [m, B_2]$
Output:
v_e Standardbitletrepräsentation v_e des Ergebnis der Vergleichsoperation

1: **for all** $((b_{x,y}, z) \in B_1 \wedge (b_{x,y}, w) \in B_2$ **do**
2: $\quad Insert(res, (b_{x,y}, w + z))$
3: **end for**
4: **for all** $((b_{x,y}, z) \in \{B_1\ XOR\ B_2\}$ **do**
5: $\quad Insert(res, (b_{x,y}, z))$
6: **end for**
7: **return** $[kgV(n, m), res]$ {kgV = kleinste gemeinsame Vielfache}

gleichen Ergebnismuster wie die Anwendung einer entsprechenden Vergleichsoperation nur im Zeitbereich.

6.4.4.2 NOT-Operation ($\neg$)

Wie bei der Realisierung der OR-Operation vorgestellt, können im $\mathbb{Z}$-Vektorraum alle Werte ungleich Null als Nicht-Null-Belegungen im Zeitbereich interpretiert werden. Übertragen auf eine NOT-Operation im $\mathbb{Z}$-Vektorraum bedeutet dies, dass bei einer solchen Interpretation alle Werte ungleich Null zu Null werden müssen und alle Werte gleich Null zu Nicht-Null-Werten. Allerdings, je häufiger eine Bitletrepräsentation, d.h. eine Koordinate bzw. Bitlet, einer OR-Operationen unterzogen wurde, um so größer kann der Wert der Nicht-Null-Belegung eines einzelnen Bitlets sein. Das heißt, bei jeder OR-Operation findet eine Addition statt, die im ungünstigsten Fall für ein Bitlet bzw. dessen Wert stets zu einer Erhöhung um $+1$ führt. Dies ist in der folgenden Tabelle dargestellt.

x	$\neg x$
0	1
1	0
2	0
3	0
$\vdots$	$\vdots$

Für die Realisierung der NOT-Operation im Bitletbereich stehen jedoch nur die beiden Operationen Addition und skalare Multiplikation zur Verfügung. Wie wir im Abschnitt 6.4.3 gezeigt haben, führt eine Addition mit 1 im $\mathbb{Z}^2$-Vektorraum zu einer entsprechenden NOT-Operation. Im $\mathbb{Z}$-Vektorraum ist dies jedoch leider nicht mehr möglich

bzw. führt nicht zu den gewünschten Effekt. Tatsächlich erschließt sich mit Blick auf die obige Tabelle auch keine Berechnungsvorschrift, die ausschließlich mit Addition und skalarer Multiplikation für ein übergebenes x zu dem gewünschten Ergebnis $\neg x$ führt.[11] Dies ändert sich leider auch nicht, wenn wir zusätzlich die Annahme machen, dass eine Negation stets nur nach maximal einer angewendeten OR-Operation stattfindet, d.h. die zu negierenden Klauseln nur die Form $\neg(\cup(v_1, v_2))$ aufweisen können. Hintergrund der Überlegung ist, dass wir an dieser Stelle wissen, dass lediglich die Werte $\{0, 1, 2\}$ im Ergebnis der Operation $\cup(v_1, v_2)$ auftreten können. Aber selbst für diesen Minimalfall erschließt sich keine Realisierungsmöglichkeit der NOT-Operation im $\mathbb{Z}$-Vektorraum.

Alternative NOT-Operation ($\overset{z}{\neg}$): Wie oben gezeigt, ergibt sich keine direkte Realisierung einer NOT-Operation im Bitletbereich. Das heißt, wir können keine Operation bzw. Rechenvorschrift im Bitletbereich angeben, die auf mit der Bitlettransformation verträglichen Operationen basiert und zu einer Negierung des Wiederholungsmusters im Zeitbereich führt.

Allerdings können wir eine solche NOT-Operation ohne Probleme im Zeitbereich realisieren. Die alternative NOT-Operation basiert auf der Überlegung, die Wiederholungsmuster nicht im Bitletbereich zu negieren, sondern dies im Zeitbereich zu tun und anschließend das negierter Muster in den Bitletbereich zu transformieren bzw. dort zu verwenden. Formal ausgedrückt, sei r ein Wiederholungsmuster und $\neg r$ das negierte Muster von r im Zeitbereich. Sei weiterhin B_n eine Bitletbasis und u der Vektor des $\mathbb{Z}$-Vektorraums V^n, der das Wiederholungsmuster r repräsentiert, d.h. $u = r \cdot B_n^{-1}$. Die NOT-Operation $\overset{z}{\neg}$ ist wie folgt definiert:

$$\boxed{\overset{z}{\neg} u = \neg r \cdot B_n^{-1}}$$

Mit dieser alternativen Realisierung der NOT-Operation $\overset{z}{\neg}$ sind jedoch Einschränkungen verbunden. In der vorliegenden Form, kann die NOT-Operation nur für "originale" Wiederholungsmuster bzw. deren Bitletrepräsentationen angewandt werden. Eine Anwendung der NOT-Operation etwa auf dem Ergebnis einer Vergleichsoperation ist nicht ohne weiteres möglich. Die vorgestellte NOT-Operation $\overset{z}{\neg}$ deckt somit lediglich den Fall ab, dass bei einem Vergleich a priori bekannt ist, dass z.T. negierte Varianten von Wiederholungsmustern benötigt werden. Anstelle die benötigten Muster im Bitletbereich zu negieren, werden diese bereits im Zeitbereich negiert. Nach erfolgter Transformation liegt dann die Bitletrepräsentation des negierten Wiederholungsmusters im Bitletbereich

[11] Leicht vorstellbar wäre auch eine Berechnungsvorschrift, die mit Fallunterscheidungen bei der Addition arbeitet. Leider muss eine solche Fallunterscheidung im Bitletbereich ebenfalls als nicht verträgliche Operation verstanden werden. Die im Bitletbereich getroffene "Fallunterscheidung" führt nicht zu der gleichen "Fallunterscheidung" im Zeitbereich. Der Interpretation bzw. Rückschlüsse von Ergebnissen im Bitletbereich auf Ergebnisse im Zeitbereich sind sehr enge Grenzen gesteckt. Wir werden auf diesen Aspekt der Interpretation der Ergebnisse im Bitletbereich noch in Abschnitt 6.5.2 näher eingehen.

vor. Eine Negation im Bitletbereich wird in diesem Fall dann nicht mehr benötigt. Offensichtlich empfiehlt es sich aus Performanzgründen jedoch nicht, die Transformation des negierte Wiederholungsmuster erst zum Vergleichszeitpunkt durchzuführen. Stattdessen sollte die Erzeugung der Bitletrepräsentation des negierten Wiederholungsmusters stets vor dem Vergleichszeitpunkt erfolgen. Es hängt hierbei von der Anwendung ab, ob für jedes Wiederholungsmuster stets auch pauschal die Bitletrepräsentation des negierten Wiederholungsmusters abgelegt wird oder nur "on demand" für einige ausgewählte Muster. Beides ist möglich und vorstellbar.

Aufgrund der oben beschriebenen Einschränkungen, dient die NOT-Operation $\overset{z}{\neg}$ lediglich dem Fall der Negation der ursprünglichen Wiederholungsmuster. In dieser Form entspricht sie jedoch keiner "echten" Negationsfunktion. Das heißt, die NOT-Operation $\overset{z}{\neg}$ und die vorherige OR-Operation bilden keine vollständige boolesche Basis, da wir die NOT-Operation nicht beliebig bzw. auf Ergebnissen von Operation bzw. Vergleichen einsetzten dürfen. Für die Simulation der AND-Operation und somit zur Bildung der booleschen Basis wäre diese jedoch entscheidend, da $x \wedge y = \neg(x \vee y)$, d.h. es müsste eine Negation des Ergebnis von $x \vee y$ stattfinden. Dies ist mit der definierten NOT-Operation $\overset{z}{\neg}$ nicht möglich und für eine entsprechende NOT-Operation im Bitletbereich konnte keine Möglichkeit identifiziert werden (siehe oben). In Hinblick auf eine vollständige booleschen Funktionsbasis wird eine AND-Operation im Bitletbereich benötigt (siehe später).

Auch wenn die vorgestellte alternative NOT-Operation ($\overset{z}{\neg}$) lediglich einen bestimmten Fall der Negation abdeckt, so ist sie wesentlich im Zusammenhang mit beliebigen Vergleichsausdrücken. Grundsätzlich gehen wir davon aus, dass Vergleichsoperationen in beliebiger Form zu einem entsprechenden Vergleichsausdruck kombiniert werden können, um anwendungsspezifische (komplexe) Beziehungen zwischen Wiederholungsmustern zu definieren. Ein beliebig gestalteter Vergleichsausdruck ist für die Vergleichsoperationen $\cup$ und $\cap$ (siehe später) problemlos, da diese in beliebiger Reihenfolge und Häufigkeit kombiniert angewendet werden dürfen. Im Allgemeinen müssen wir ebenfalls davon ausgehen, dass auch die NOT-Operation an beliebiger Stelle in einem entsprechenden Vergleichsausdruck vorkommen kann bzw. darf. Allerdings führt dies bei der NOT-Operation ($\overset{z}{\neg}$) zu einem Problem. In der vorliegenden Form ist diese Operation nur unnmittelbar auf den urspünglichen Wiederholungsmustern bzw. deren Bitletrepräsentationen definiert. Diese Operation darf jedoch nicht auf dem Ergebnis einer erfolgten Vergleichsoperation im Bitletbereich angewendet werden.

Um diesen Problem zu begegnen, fordern wir stets, dass vor der konkreten Durchführung der Vergleichsoperationen, der Vergleichsausdruck umgewandelt wird. Hierzu kann der Vergleichsausdruck als "klassische" aussagenlogische Formel verstanden werden. Das heißt, die Vergleichsoperationen $\cap, \cup$ und NOT können als die boolesche Funktion $\wedge, \vee$ und $\neg$ verstanden werden. Wie aus der allgemeinen Theorie bekannt, lässt sich jede Formel der Aussagenlogik so umwandeln, dass Negationen nur unmittelbar vor den Variablen der Formel stehen [76]. Die Grundlage hierfür bilden die De Morgan'schen

Gesetze, die wie folgt lauten:

$$\neg(a \wedge b) = \neg a \vee \neg b$$
$$\neg(a \vee b) = \neg a \wedge \neg b$$

Auf Basis dieser zwei grundlegenden Regeln für logische Aussagen, kann auch jeder beliebige Vergleichsausdruck auf Basis von Mengen- bzw. Vergleichsoperationen entsprechend umgewandelt werden, d.h. $\overline{A \cap B} = \overline{A} \cup \overline{B}$ bzw. $\overline{A \cup B} = \overline{A} \cap \overline{B}$ [76]. Infolgedessen wird eine "freie" NOT-Operation im Bitletbereich nicht benötigt, da alle Negationen bzw. NOT-Operationen nach der Umwandlung nur noch unmittelbar auf den urspünglichen Wiederholungsmustern bzw. deren Bitletrepräsentationen vorkommen. Grundbedingung hierfür ist, dass im Bitletbereich die Vergleichsoperation $\cap$ zur Verfügung steht, die jedoch im folgenden Abschnitt vorgestellt wird.

6.4.4.3 AND-Operation ($\cap$)

Aufgrund unserer vorherigen Betrachtungen wissen wir, dass eine AND-Operation einer Multiplikation im Bitletbereich entspricht. Leider sind beide Operationen weder im Allgemeinen mit linearen Abbildungen, noch im Speziellen mit der Bitlettransformation verträglich. Eine Realisierung der AND-Operation bzw. der Vergleichsoperation $\cap$ ist – im Gegensatz zu den vorherigen Operationen – nicht durch Operationen abbildbar, die mit der Bitlettransformation verträglich sind.

Unsere bisherigen Überlegungen zielten alle auf eine direkte Realisierung von Vergleichsoperationen im Bitletbereich ab, ohne Berücksichtigung der zugrundeliegenden Basismatrizen. Eine direkte Realisierung ist jedoch nach derzeitigem Stand nicht möglich bzw. es wurden von uns keine entsprechenden Möglichkeiten für eine solche identifiziert. Das im Folgenden vorgestellte Verfahren unterscheidet sich daher maßgeblich von den bisherigen Realisierungen, da dieses nicht alleine im Bitletbereich aggiert, sondern zusätzlich auch die Basismatrizen bzw. die Bitlettransformation berücksichtigt. Wir beginnen unsere Betrachtungen mit der Grundidee zur Realisierung der Multiplikation (AND-Operation) im Bitletbereich.

Grundidee zur Realisierung der Multiplikation im Bitletbereich: Zur Erläuterung der Grundidee betrachten wir zunächst das zu lösende Grundproblem bei der Realisierung der Multiplikation. Es seien $x, y \in V^n$ zwei Wiederholungsmuster des $\mathbb{Z}$-Vektorraums V^n im Zeitbereich. Seien weiterhin B_n eine Bitletbasis der Länge n und die Vektoren $v, u \in V^n$ die Bitletrepräsentationen der beiden Wiederholungsmuster, d.h. $B_n \cdot u = x$ und $B_n \cdot v = y$. Der Vektor $w \in V^n$ ist Bitletrepräsentation, die das Ergebnis der Multiplikation von x und y im Bitletbereich darstellt, d.h. $B_n \cdot w = x \cdot y$, wobei $x \cdot y$ die paarweise Multiplikation ist. Ausgehend von diesen Vektoren haben wir zur Realisierung der Multiplikation die folgende Gleichung zu lösen:

$$\underbrace{B_n \cdot w}_{=x \cdot y} = \underbrace{(B_n \cdot u)}_{=x} \cdot \underbrace{(B_n \cdot v)}_{=y} \tag{6.1}$$

Diese Gleichung stellt ein lineares Gleichungssystem dar, in dem der Vektor w bzw. dessen Koordinaten w_k mit $k = 1, ..., n$ die Unbekannten bilden. Die Grundidee der Realisierung der Multiplikation im Bitletbereich ist es, die Multiplikation zwischen Vektoren u und v auf die Lösung des obigen linearen Gleichungssystems abzubilden. Das heißt, die Lösung des Gleichungssystem führt zu dem konkreten Vektor w, der das Ergebnis der Multiplikation der beiden Vektoren x und y im Bitletbereich repräsentiert.

Zur Lösung linearer Gleichungssysteme sind in der Literatur verschiedene Methoden und Verfahren bekannt, wie etwa das gaußsche Eliminationsverfahren [58]. Leider benötigt das Gauss-Verfahren etwa n^3-Schritt für eine entsprechende Lösung. Die Lösung des linearen Gleichungssystems würde somit zum gleichen bzw. größeren Aufwand führen wie die Transformation der Vektoren u, v in den Zeitbereich und Durchführung der (paarweisen) Multiplikation im Zeitbereich. Auch wenn modernere Verfahren hier bessere Ergebnisse erzielen können, so stehen deren Aufwände immer auch mit der Anzahl der Unbekannten bzw. dem Umfang des Gleichungssystems in Beziehung [58, 74]. Ohne weitere Kenntnisse zu den Eigenschaften der Vektoren und Matrizen kann den anfallenden Aufwänden zur Lösung des Gleichungssystems kaum begegnet werden. Allerdings haben wir im Bitletbereich solche zusätzlichen Kenntnisse zu den Vektoren und Matrizen. Genauer gesagt, kennen wir die Struktur der Matrizen sehr genau und ebenso die der Vektoren. Dies wiederum ermöglicht es uns, den Umfang des linearen Gleichungssystems z.T. stark zu reduzieren. Die Idee ist hierbei vergleichbar zur Fast-Bitlet-Transformation, d.h. wir eleminieren die Teile aus dem Gleichungssystem von denen wir wissen, dass sie Null ergeben bzw. zu einer Koordinate $w_k = 0$ im (gesuchten) Vektor w führen. Nach der Reduktion können dann entsprechende Verfahren zur Lösung des reduzierten Gleichungssystems angewandt werden, was aufgrund der reduzierten Größe zu deutlich geringeren Aufwänden führt.

In den folgenden Abschnitten werden wir die Reduzierung des Gleichungssystems vorstellen. Hierzu werden wir im ersten Schritt eine alternative Darstellung der Gleichung 6.1 verwenden. Im Folgenden bezeichnen wir mit b^j mit $j = 0, ..., n$ eine Spalte der Bitletmatrix B_n. Für jede Spalte $b^j \in B_n$ mit $j = 1, ..., n$ muss das folgende Gleichungssystem gelöst werden:

$$b^j \cdot w = (b^j \cdot u) \cdot (b^j \cdot v) \tag{6.2}$$

Die Gleichung 6.2 formen wir um zu der folgenden Gleichung:

$$0 = b^j \cdot (w - u \cdot b^j \cdot v) \tag{6.3}$$

In "ausgeschriebener" Form ergibt sich für die Gleichung 6.3 die folgende Darstellung:

$$0 = \begin{pmatrix} b_1^j \\ \vdots \\ b_n^j \end{pmatrix} \cdot \left((w_1, ..., w_n) - (u_1, ..., u_n) \cdot \begin{pmatrix} b_1^j \\ \vdots \\ b_n^j \end{pmatrix} \cdot (v_1, ..., v_n) \right) \tag{6.4}$$

Zur Reduktion der einzelnen Gleichung 6.4 bzw. des gesamten Gleichungssystem mit $j = 0, ..., n$ gilt es, die Koordinaten zu identifizieren, die zu einer Null führen bzw. zu einer Koordinaten $w_k = 0$ im Vektor w. Diese Reduzierungen werden wir für die einzelnen Vektoren u, w und b_j nun im Einzelnen vorstellen.

Reduzierung der Vektoren u**:** Zur Reduzierung der Vektoren u und v wollen wir uns das Teilprodukt $u \cdot b^j \cdot v$ in Gleichung 6.3 genauer anschauen. Aufgrund des Assoziativgesetzes kann das Teilprodukt geklammert werden zu $u \cdot (b^j \cdot v)$. Das hintere Produkt führt hierbei zu einer einzelnen Zahl, da nach dem Matrizenmultiplikationsgesetz gilt, *Spalte* $\cdot$ *Zeile* = *Zahl*. Daraus folgt jedoch, egal welche Zahl das hinter Produkt ergibt, falls im Vektor u eine Koordinaten $u_k = 0$ mit $k = 1, ..., n$ enthält, ist das gesamte Teilprodukt $u \cdot b^j \cdot v$ ebenfalls Null. Somit sind in u nur die Koordinaten $u_k \neq 0$ relevant. Die Positionen k der Nicht-Null-Koordinaten werden in der Menge K_1 hinterlegt, die wir später noch benötigen werden.

Reduzierung des Vektors w**:** Zur Reduzierung des Vektors w benötigen wir die Koordinaten, für die wir sicher wissen, dass $w_k = 0$. Allerdings stellt der Vektor w eigentlich den unbekannten bzw. gesuchten Vektor dar.

Zur Identifikation dieser Koordinaten des gesuchten Vektors w benötigen wir ein anwendungsspezifisches Wissen, dass uns eine entsprechende Reduzierung erst ermöglicht. Von einem solchen Wissen sind wir jedoch bereits bei allgemeinen Bitletbasen ausgegangen. Das heißt, bei der Definition solcher Basen gehen wir von problemspezifischen Bitlets aus, die besonders gut zu den in der Anwendungsdomäne vorkommenden Wiederholungsmustern passen (siehe Abschnitt 6.2.7.1). In gleicher Weise wollen wir auch an dieser Stelle verfahren, d.h wir gehen davon aus, dass wir ein solches Wissen besitzen. Somit können wir für den Vektor w eine Menge von Koordinaten w_k angeben, von denen wir wissen bzw. davon ausgehen, dass $w_k = 0$. Die restlichen Koordinaten, d.h. die Koordinaten von denen wir ausgehen, dass $w_k \neq 0$, werden in die Menge K_2 übertragen. Das heißt jedoch auch, je weniger Informationen wir besitzen, umso größer wird die Anzahl der Koordinaten, die wir nicht ausschließen können und und desto länger wird später das Verfahren zur Lösung des reduzierten Gleichungssystems benötigen. Wir werden auf diesen Schritt bzw. auf dessen Annahmen in Abschnitt 6.5 noch genauer eingehen.

Reduzierung des Vektors b_j: Die Reduzierung der Vektoren u und w führt zu einer Menge an Koordinaten $K = K_1 \cup K_2$, von denen wir wissen, dass nur hier Nicht-Null-Koordinaten auftauchen werden. Allerdings haben wir aufgrund von Vektor b^j und $j = 1, ..., n$ noch immer n-viele Gleichungen in unserem Gleichungssystem 6.4. Diese Menge gilt es durch die Reduktion der Vektoren b_j zu verringern.

Zur Reduktion des Vektors b^j greifen wir auf die Ideen bzw. Funktionen der (inversen) Fast-Bitlet-Transformation zurück. Genauer gesagt auf die Funktion $B_n(j)$, die uns für eine Spalte j der Bitletmatrix B_n die Bitlets zurückliefert, die eine Nicht-Null-Belegung aufweisen (siehe Abschnitt 6.3.2.1). Im Folgenden gehen wir von einer leicht geänderten Form dieser Funktion aus, sodass anstelle der eindeutigen Bitletindizes die absoluten Koordinaten k (Zeilennummern in der Bitletmatrix) zurückgegeben werden. Auf Basis dieser Funktion kann dann eine Menge J wie folgt berechnet werden:

$$J = K \cap \bigcup_{j=1,..,n} B_n(j)$$

Die Menge J enthält alle Koordinaten, in denen die Kombination von Nicht-Null-Koordinaten von b^j und Nicht-Null-Koordinaten der Vektoren u und v zusammenfallen. Das heißt, in der Menge J sind die Koordinaten k nicht enthalten, die zwar in b^j eine Nicht-Null-Belegung aufweisen, aber in u oder w eine Null-Belegung und anders herum.

Reduziertes Gleichungssystem: Auf Basis der bei der Reduzierung der einzelnen Vektoren entstandenen Mengen K und J erhalten wir das folgende reduzierte Gleichungssystem:

$$\forall j \in J : \sum_{k \in K} b_{jk} \cdot w_k = \sum_{k \in K} b_{jk} \cdot u_k \cdot (b_j \cdot v) \tag{6.5}$$

Der hintere Term $(b_j \cdot v)$ der Gleichung 6.5 ist hierbei sehr schnell berechenbar, da nur die Nicht-Null-Koordinaten von v und der Spalte b^j berücksichtigt werden müssen. Für Letzteres kommt erneut die Funktion $B_n(j)$ der (inversen) Fast-Bitlet-Transformation zum Einsatz, wie wir sie bereits bei der Reduzierung des Vektors b^j (siehe oben) verwendet haben.

Im Gegensatz zu Gleichung 6.2 müssen zur Lösung des Gleichungssystems in Gleichung 6.5 nicht mehr alle $j = 1, ..., n$ und $k = 1, ..., n$ betrachtet werden, sondern nur noch die aus den Mengen J und K. Der Umfang der beiden Mengen Kund J ist abhängig von Umfang der einzelnen Reduzierungen der Vektoren u, w und b^j. Wenn wir jedoch davon ausgehen, dass diese Vektoren "sparse" waren, d.h. dünnbesetzt, dann werden auch die beiden Mengen Kund J einen geringen Umfang aufweisen. Dies wiederum führt dazu, dass die Größe des linearen Gleichungssystems in seinem Umfang reduziert wird. Dieses reduzierte lineare Gleichungssystem kann mit entsprechenden Verfahren gelöst

werden, wie etwa dem Standard-Gauß-Verfahren oder anderen, moderneren Verfahren (siehe [74, 58]). Unabhängig vom konkreten Verfahren, liefern diese eine Menge von Koordinaten für den gesuchten Vektor w mit $w_k \neq 0$. Im Bezug auf die Bitletbasen müssen die bei der Lösung des Gleichungssystems ermittelten Koordinaten lediglich in die entsprechende Bitletindizierung der Bitletbasen "umgerechnet" werden, was jedoch leicht lösbar ist (siehe Abschnitt 6.2.7.1). Im Ergebnis erhält man die Bitletrepräsentation, die das Resultat der Vergleichsoperation $\cap$ zwischen den beiden Bitletrepräsentationen u und v repräsentiert.

Wir illustrieren im Folgenden das Verfahren anhand eines Beispiels. Wir gehen von den beiden Wiederholungsmustern $x = (1, 0) \in V^2$ und $y = (0, 1, 0) \in V^3$ aus bzw. von deren Bitletrepräsentation $u = (1, -1, 0, 0, 0, 0) \in V^6$ und $v = (0, 0, 1, 0, 0, 0) \in V^6$ im gemeinsamen Bitletraum $V^{2,3}$. Die Standardbitletbasis für V^6 ist wie folgt:

$$B_6 = \begin{pmatrix} \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

Zur Realisierung der Multiplikation bzw. der Vergleichsoperation $u \cap v = w$ ergibt sich das folgende zu lösende Gleichungssystem:

$$j = 1: \quad 0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot \left((w_1, ..., w_n) - (1, -1, 0, 0, 0, 0) \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot (0, 0, 1, 0, 0, 0) \right)$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$j = 6: \quad 0 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot \left((w_1, ..., w_n) - (1, -1, 0, 0, 0, 0) \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot (0, 0, 1, 0, 0, 0) \right)$$

Im ersten Schritt gilt es den Vektor u zu reduzieren, d.h. es sind nur die Nicht-Null-Koordinaten des Vektors u relevant, die in die Menge $K_1 = \{1, 2\}$ übertragen werden.

Im zweiten Schritt erfolgt die Reduzierung des gesuchten Vektors w. Im vorliegenden Fall wissen wir, dass gemeinsame Werte bei den Wiederholungsmustern eher am Ende

stattfinden können als am Anfang. Genauer gesagt zeigt sich, dass bei einem paarweisen Vergleich von Muster x und y innerhalb des unverlängerten Musters y kein gemeinsamer Wert existiert. Folglich müssen die gemeinsamen Werte in den hinteren Positionen anzutreffen sein (aufgrund unserer Betrachtungen aus Abschnitt 5.3.2 wissen wir, dass solche gemeinsamen Werte vorkommen müssen). Des Weiteren wissen wir, dass bei der Standardbitletbasis einzelne Werte im hinteren Bereich eines Musters bzw. Vektors im Zeitbereich im Bitletbereich ebenfalls durch Bitlets im hinteren Bereich des Koordinatensvektors repräsentiert werden. Dies ergibt sich aufgrund der Konstruktion solcher Basen, in dem die Ergänzungsvektoren von rechts aufgefüllte Standardvektoren darstellen (siehe Abschnitt 6.2.5.2). Solche Ergänzungsvektoren bzw. Bitlets weisen nämlich lediglich am Ende eine einzige Eins auf. Ein Wiederholungsmuster im Zeitbereich, das lediglich am Ende eine oder mehrere Einsen aufweist, wird dann (meist) durch diese Basisvektoren bzw. Bitlets im Ergänzungsbereich der Basis repräsentiert. Aufgrund dieser Überlegungen können wir im vorliegenden Beispiel annehmen, dass der Vektor w lediglich an den Positionen $K_2 = \{5, 6\}$ eine Nicht-Null-Koordinate aufweisen wird.

Im letzten Schritt müssen die Spaltenvektoren b^j der Basismatrix für $j = 1, ..., n$ reduziert werden. Hierzu wird mit der Funktion $B_6(j)$ für jede Spalte die Positionen der Nicht-Null-Koordinaten für jede Spalte ermittelt. Im Ergebnis führt dies zu den Teilmengen $B_6(1) = \{1\}$, $B_6(2) = \{1, 2, 3\}$,..., $B_6(6) = \{1, 2, 6\}$, d.h. es ergibt sich die Menge $J = \{1, 2, 5, 6\}$.

Auf Grundlage der Mengen K und J gilt es, das nachfolgende Gleichungssystem zu lösen. Die rechte Seite des Gleichungssystem ist hierbei schon berechnet. Das heißt, auf der rechten Seite der Gleichung 6.5 sind keine Unbekannten enthalten, sodass die konkreten Werte des vorliegenden Beispiels berechnet werden können.

$$
\begin{aligned}
b_{11} \cdot w_1 + b_{12} \cdot w_2 + b_{15} \cdot w_5 + b_{16} \cdot w_6 &= 0 \\
b_{21} \cdot w_1 + b_{22} \cdot w_2 + b_{25} \cdot w_5 + b_{26} \cdot w_6 &= 0 \\
b_{51} \cdot w_1 + b_{52} \cdot w_2 + b_{55} \cdot w_5 + b_{56} \cdot w_6 &= 1 \\
b_{61} \cdot w_1 + b_{62} \cdot w_2 + b_{65} \cdot w_5 + b_{66} \cdot w_6 &= 0
\end{aligned}
$$

Das obige Gleichungssystem hat die folgende alternative Darstellungsform mit Koeffizientenmatrix:

$$
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_5 \\ w_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}
$$

Gegenüber dem ursprünglichen Gleichungssystems des Beispiels ist das reduzierte Gleichungssystem in seinem Umfang eingeschränkt. Die Lösung des Gleichungssystems führt zu dem Ergebnisvektor ($w_1 = 0, w_2 = 0, w_5 = 1, w_6 = 0$) bzw. zu dem Vektor

$w = (0, 0, 0, 0, 1, 0)$ im Vektorraum V^6, der das Ergebnis der Multiplikation $u \cdot v = w$ bzw. der AND-Operation $u \cap v = w$ darstellt.

In dem vergangenen Abschnitt haben wir gezeigt, dass die benötigten Vergleichsoperationen $\{\cap, \cup, \overset{z}{\neg}\}$ im $\mathbb{Z}$-Vektorraum im Bitletbereich vollumfänglich realisiert werden können. Die Vergleichsoperation $\cup$ erweist sich hierbei als sehr einfach und effizient realisierbar. Gleiches gilt für die NOT-Operation ($\overset{z}{\neg}$), auch wenn hierfür der Vergleichsausdruck umgewandelt werden muss. Im Gegensatz dazu wird bei der AND-Operation ein deutlich komplexeres und aufwändigeres Realisierungsverfahren benötigt. Die konkreten Aufwände zur Realisierung der Vergleichsoperationen werden im Folgenden jedoch keine weitere Rolle spielen. Entsprechende Betrachtung zu Aufwänden und Komplexitäten werden im Detail in Abschnitt 6.5 vorgenommen und mit existierenden Verfahren verglichen. Stattdessen werden wir im Folgenden zunächst die Möglichkeiten der eingeschränkten Auswertung von Vergleichsoperationen im Bitletbereich betrachten.

6.4.5 Eingeschränkte Auswertung von Vergleichsoperationen

Eine wesentliche Anforderung an Vergleichsoperationen für periodische Spezifikationen stellt deren eingeschränkte Auswertung dar. Hierbei werden die bei einem Vergleich spezifizierten Mengenbeziehungen zwischen Spezifikationen lediglich innerhalb eines bestimmten Auswertungszeitraums bzw. in dem resultierenden Ereignishorizont überprüft. Als Ereignishorizont wird hierbei der (absolute) Zeitabschnitt im Zeitbereich verstanden, der sich ausgehend vom Zeitpunkt der Vergleichsoperation und dem Auswertungszeitraum ergibt. Im Gegensatz zur unbeschränkten Auswertung sind die getroffenen Vergleichsaussagen nicht allgemeingültig, sondern nur innerhalb des Ereignishorizonts. Eine solche eingeschränkte Auswertung von Vergleichsoperationen werden wir im Folgenden für den Bitletbereich bzw. Bitletrepräsentation vorstellen.

6.4.5.1 Abbildung des Ereignishorizonts im Bitletbereich

Der Ereignishorizont entspricht einem absoluten Zeitabschnitt bestimmter Länge im Zeitbereich, in dem die Vergleichsauswertung stattfindet. Aus Gründen der Vereinfachung gehen wir an dieser Stelle davon aus, dass die Länge des Ereignishorizonts kleiner als die Länge der Wiederholungsmuster bzw. Dimension der Vektorräume ist.[12] Für Wiederholungsmuster im Zeitbereich bedeutet die eingeschränkte Vergleichsauswertung, dass nicht die vollständigen Muster verglichen werden, sondern nur die mit dem Ereignishorizont korrelierenden Teilbereiche der Muster. Das heißt, in Abhängigkeit von der Musterlänge können die relative Position bzw. Positionen innerhalb des Musters bestimmt werden, die mit dem gegebenen Ereignishorizont korrelieren. Nur für

[12] Der andere Fall, d.h. der Ereignishorizont ist größer als die Musterlänge, wird in Abschnitt 6.5 betrachtet.

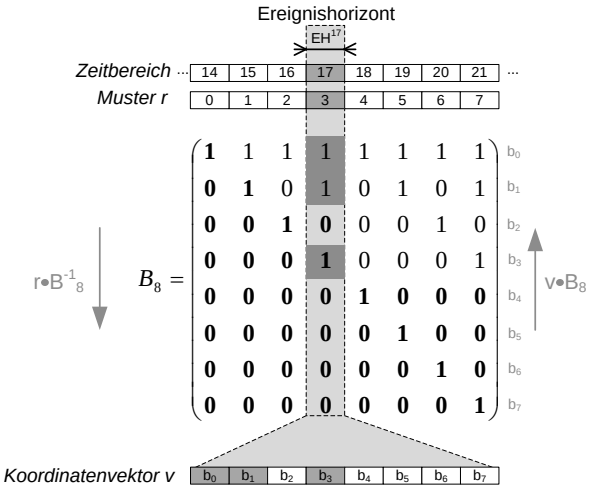


Abbildung 6.11: Relevante Bitlets für den Ereignishorizont = 17

diese korrelierenden Positionen wird ein entsprechender Vergleich vorgenommen. Abbildung 6.11 zeigt ein Beispiel mit Ereignishorizont = 17, der mit der 3. Position des Musters r korreliert (oberer Bereich der Abbildung).

Die Abbildung des im Zeitbereich definierten Ereignishorizonts kann im Bitletbereich als eine Art "Filterfunktion" verstanden werden. Diese Filterfunktion ermittelt, welche der Basisvektoren bzw. Bitlets einer entsprechenden Bitletbasis für den gegebenen Ereignishorizont bzw. Zeitabschnitt relevant sind. Anders ausgedrückt, welche der Bitlets der Basis innerhalb des Zeitabschnitts potentiell überhaupt zu einer Nicht-Null-Belegung im korrespondierenden Wiederholungsmuster im Zeitbereich führen können. Nur die relevanten Bitlets müssten dann bei der eingeschränkten Vergleichsoperation im Bitletbereich berücksichtigt werden (siehe später). Hierbei spielt es keine Rolle, welcher Zahlenbereich bzw. entsprechender Vektorraum zugrundeliegt, das Vorgehen ist bei allen gleich (siehe Abschnitt 6.4.3 und 6.4.4). Abbildung 6.11 zeigt ein entsprechendes Beispiel für den Ereignishorizont = 17 und der Bitletbasis B_8. Hierbei sind für den Ereignishorizont lediglich die Bitlets $\{b_0, b_1, b_3\}$ relevant. Lediglich diese Bitlets weisen innerhalb des Ereignishorizonts eine Nicht-Null-Belegung auf und müssten somit bei der eingeschränkten Vergleichsauswertung als Einzige berücksichtigt werden. Alle anderen Bitlets haben in dem gegebenen Ereignishorizont keinerlei Einfluss auf das entsprechende Wiederholungsmuster im Zeitbereich.

Die Bestimmung der für einen Ereignishorizont relevanten Bitlets gestaltet sich insofern als einfach, da wir eine solche Funktion bereits bei der Fast-Bitlet-Transformation beschrieben haben. Es handelt sich hier um die Funktion $B_n(j)$, die für jede Spalte j einer Matrix B_n die Bitlets ermittelt, deren Belegung ungleich Null ist (siehe Abschnitt 6.3.2). Auf Basis dieser Funktion können wir eine Filterfunktion $EH()$ beschreiben, die für den Ereignishorizont $z_s, z_e \in \mathbb{Z}$ die nicht-relevanten Bitlets aus der Bitletrepräsentation v

Algorithmus 6.3 EH($\mathbf{v}, \mathbf{z_s}, \mathbf{z_e}$) $\mapsto \mathbf{v'}$

Input:

v Standardbitletrepräsentation $v = [m, \{(b_{i,j}, z_0), ..., (b_{k,l}, z_n)\}]$

z_s Beginn Ereignishorizont mit $z_s \in \mathbb{Z}$

z_e Ende Ereignishorizont mit $z_e \in \mathbb{Z}$

Output:

v' Bezüglich des Ereignishorizonts z eingeschränkte Bitletrepräsentation v'

1: **for** $i = z_s$ to z_e **do**
2: $s = i$ mod m {Mit i korrelierende Spalte s der Basismatrix B_m}
3: $Insert(rBitlets, B_m(s))$ {Relevante Bitlets der Basis B_m in Spalte s}
4: **end for**
5: **for all** $(b_{x,y}, z) \in v$ **do**
6: **if** $b_{x,y} \in rBitlets$ **then**
7: $Insert((b_{x,y}, z), v_{tmp})$
8: **end if**
9: **end for**
10: **return** $[m, v_{tmp}]$

"ausfiltert". Im Algorithmus 6.3 ist die Realisierung der Funktion $EH()$ im Detail vorgestellt.

Im weiteren Verlauf der Arbeit wird für die Funktion $EH(x, s, e)$ die Abkürzung $EH_s^e(x)$ verwendet. Beispielsweise ergibt sich für $v = [8, \{(b_1, 1), (b_2, 1), (b_7, -1)\}]$ nach Einschränkung bezüglich des gegeben Ereignishorizonts $= 17$ die Bitletrepräsentation $v' = EH_{17}^{17}(v) = [8, \{(b_1, 1)\}]$. Die ursprünglich in der Bitletrepräsentation v enthaltenen Bitlets b_2 und b_7 müssen für diesen Ereignishorizont nicht berücksichtigt werden.

6.4.5.2 Problem der eingeschränkten Vergleichsauswertung

Die eingeschränkte Vergleichsauswertung im Bitletbereich bedeutet, dass beim Vergleich lediglich die für den gegebenen Ereignishorizont relevanten Bitlets einer Bitletrepräsentation berücksichtigt werden dürfen. Hierbei können grundsätzlich zwei Arten der eingeschränkten Vergleichsauswertung unterschieden werden:

- **A priori Realisierung:** Die Bestimmung der relevanten Bitlets findet vor der Ausführung der Vergleichsoperation statt. Die eingeschränkte Auswertung des Vergleichs $\otimes$ der Bitletrepräsentationen v_1, v_2 weist in dieser Realisierungsvariante die Struktur $EH_s^e(v_1) \otimes EH_s^e(v_2)$ auf.
- **A posteriori Realisierung:** Die Bestimmung der relevanten Bitlets findet nach der Ausführung der Vergleichsoperation statt. Das heißt, ein eingeschränkter Vergleich $\otimes$ weist die Struktur $EH_s^e(v_1 \otimes v_2)$ auf.

Im Bitletbereich weisen die beiden obigen Realisierungsvarianten identische Eigenschaften auf. Es spielt im Bitletbereich daher keine Rolle, ob wir die Einschränkung vor der Vergleichsoperation auf den zu vergleichenden Wiederholungsmustern vornehmen oder nach der Vergleichsoperation auf dem Ergebnismuster des Vergleichs. Es hängt von der Größe der Bitletrepräsentation und Länge des Ereignishorizonts ab, welche der beiden Varianten im konkreten Fall verwendet werden sollte (siehe Abschnitt 6.5).

Für die unbeschränkte Auswertung haben wir gezeigt, dass die Interpretation des Vergleichsergebnisses, d.h. die Aussage zur Erfüllung oder Nicht-Erfüllung des Vergleichs einfach zu realisieren ist. Die hierfür notwendige Überprüfung, ob sich die leere Menge ergibt oder nicht, kann direkt an der resultierenden Bitletmenge einer Bitletrepräsentation "abgelesen" werden. Genauer gesagt, eine leere Bitletmenge in der Bitletrepräsentation bedeutet stets, dass der Vergleich erfüllt ist, d.h. kein Wert im Zeitbereich existiert, der gegen die im Vergleich spezifizierte Mengenbeziehung verstößt. Das Gleiche gilt aber auch andersherum, d.h. enthält die Bitletmenge einer Bitletrepräsentation ein Bitlet bzw. ein Bitlet mit Wert ungleich Null, so folgt daraus, dass Werte im Zeitbereich existieren, die gegen die Mengenbeziehung verstoßen. Diese Menge verstoßender Werte wird durch die Bitletrepräsentation dargestellt (siehe Abschnitt 6.4.4).

Allerdings tritt bei der eingeschränkten Vergleichsauswertung im Bitletbereich im Zusammenhang mit der Interpretation der Vergleichsergebnisse ein entscheidendes Problem auf – unabhängig von der Realisierungsvariante. Bei der eingeschränkten Vergleichsauswertung existiert der oben dargestellte Zusammenhang nicht mehr. Es ergibt sich das Problem, dass wir anhand der resultierenden Bitletrepräsentation nicht direkt die Erfüllung oder Nicht-Erfüllung der Vergleichsoperation folgern können. Genauer gesagt können wir alleine aus der Existenz von Bitlets nicht entscheiden, ob diese innerhalb des gegebenen Ereignishorizonts im korrespondierenden Wiederholungsmuster im Zeitbereich zu einer Nicht-Null-Belegung führen oder nicht. Lediglich bei der leeren Bitletmenge kann dies auch bei der eingeschränkten Auswertung entschieden werden. Beispielsweise weist die Bitletrepräsentation $[8, \{(b_0, -1), (b_1, 1), (b_4, 1)\}]$ in dem Ereignishorizont $EH_{14}^{17}()$ aus Abbildung 6.12 keine Belegung auf, obwohl alle Bitlets für den Ereignishorizont relevant sind und einen Wert ungleich Null aufweisen. Im Gegensatz dazu, führt die Bitletrepräsentation $[8, \{(b_0, -1), (b_1, 1), (b_3, 1)\}]$ zu Belegungen innerhalb des Ereignishorizonts (an erster und zweiter Stelle). Für beide Beispiele ist die Interpretation jedoch nicht anhand der Bitletrepräsentation entscheidbar.

Die Interpretation des Vergleichsergebnisses kann bei der eingeschränkten Auswertung ausschließlich über eine Rücktransformation in den Zeitbereich innerhalb des Ereignishorizonts erfolgen. Hierbei wird für jeden Wert im Ereignishorizont bzw. mit diesem korrelierende Musterposition überprüft, ob die Rücktransformation der Bitletrepräsentation an dieser Position eine Nicht-Null-Belegung aufweist oder nicht. Das Vorkommen einer Nicht-Null-Belegung im Ereignishorizont entspricht der Nicht-Erfüllung des Vergleichs bzw. bei Nicht-Vorkommen dessen Erfüllung.

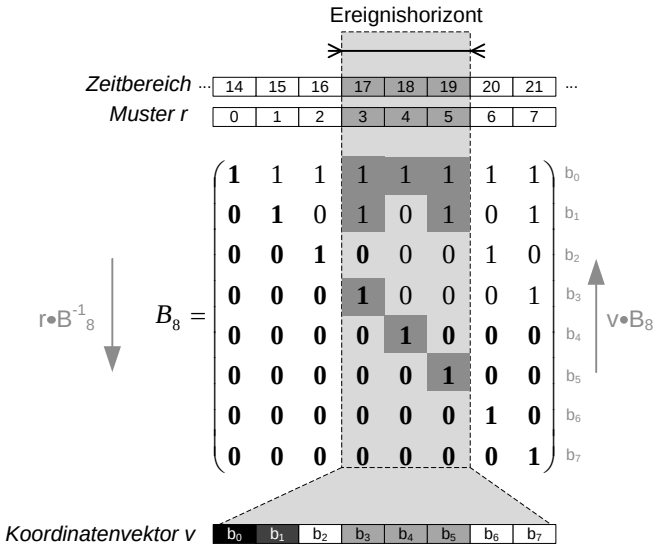


Abbildung 6.12: Relevante Bitlets für den Ereignishorizont $= 17 - 19$

Das Verhalten der eingeschränkten Auswertung bzgl. der Interpretation des Vergleichsergebnisses widerspricht der eigentlichen Grundidee der bitletbasierten Realisierung. Bei dieser sind wir davon ausgegangen, dass wir Vergleiche vollständig im Bitletbereich durchführen und auch das Ergebnis interpretieren können. Eine Rücktransformation in den Zeitbereich bzw. in das korrespondierende Wiederholungsmuster braucht somit nur wenn gewünscht stattfinden. Allerdings stellt dieses Verhalten der Bitletrepräsentation kein spezifisches Problem dar, sondern ist vielmehr kennzeichnend für lineare Abbildungen wie die Bitlet-Transformation (eine ist) [44, 13]. Beispielsweise weist die Interpretation der Ergebnisse bei der Fouriertransformation vergleichbare Probleme auf. So kann aus der Existenz bestimmter Frequenzen im Frequenzbereich keine Aussage zu dem "Ort", wo diese Frequenzen im Zeitbereich konkret wirken, abgeleitet werden. Dieses "Lokalisationsproblem" bzw. diese Unschärfe Relation ist analog zu dem Problem der Bitlet-Transformation, bei der wir aus der Existenz der Bitlets keinen Ort im Zeitbereich schlussfolgern können. Genauer betrachtet besteht das Problem bereits bei der unbeschränkten Auswertung von Vergleichsoperationen. Allerdings entspricht hier die Auswertung der Frage "*besteht irgendwo im Zeitbereich eine Nicht-Null-Belegung*", was, wie gezeigt, leicht entscheidbar ist. Die Frage hingegen nach dem "wo im Zeitbereich", ist auch hier nur nach entsprechender Rücktransformation beantwortbar. Bei einer unbeschränkten Auswertung ist das "irgendwo im Zeitbereich" ausreichend, nicht jedoch bei der eingeschränkten Auswertung, in welcher das "Wo", d.h. ob innerhalb oder außerhalb des Ereignishorizonts, entscheidend ist.

6.5 Diskussion

Das Ziel der Einführung der Bitletrepräsentation war es, die Schwächen bestehender Repräsentationsformen von Wiederholungsmustern bei der Realisierung von Vergleichsoperationen zu beheben. Hierzu haben wir in den vorherigen Abschnitten zunächst die Bitletrepräsentation eingeführt und eine bitletbasierte Realisierung von Vergleichsoperation vorgestellt. Im Folgendem werden wir die Ergebnisse bzw. Eigenschaften der bitletbasierten Realisierung zusammenfassen und diese mit den anderen Repräsentationsformen, d.h. der dichtbesetzten und dünnbesetzten linearen Musterrepräsentation, vergleichen.

6.5.1 Unbeschränkte Auswertung von Vergleichsoperationen

Das Kernproblem bei der Realisierung von Vergleichsoperationen bei unbeschränkter Auswertung bildet die notwendige Vereinheitlichung von Wiederholungsmustern unterschiedlicher Periodenlänge. Hiervon waren in gleicher Weise die dichtbesetzte und dünnbesetzte Variante der linearen Musterrepräsentation betroffen. Während sich bei ersteren etwa die Länge des Bitvektors entsprechend vergrößert, erhöht sich bei zweiteren die Anzahl der Nicht-Null-Belegungen. Bei beiden führt dies zu einem starken Anstieg der notwendigen (paarweisen) Einzelvergleiche zwischen den Musterpositionen (siehe Abschnitt 6.1).

In der Bitletrepräsentation wird, wie gezeigt, ebenfalls eine Vereinheitlichung bei unterschiedlicher Länge der Wiederholungsmuster benötigt. Im Bitletbereich wird dies als Raumeinbettung in einen gemeinsamen Vektorraum realisiert. Ausgehend von zwei Mustern der Längen m und n, müssen deren Bitletrepräsentationen $v_1 \in V^n$ und $v_2 \in V^m$ zunächst in den gemeinsamen Vektorraum $V^{n \cdot m}$ eingebettet werden, bevor etwa Vergleichsoperationen durchgeführt werden können. Die Raumeinbettung ist das analoge Vorgehen im Bitletbereich wie die Vereinheitlichung von Mustern im Zeitbereich. Im Gegensatz zum Zeitbereich erweist sich jedoch die Raumeinbettung als deutlich effizienter. Genauer gesagt erweist sich die Raumeinbettung bei der Bitletrepräsentation, als dünnbesetzte Repräsentation des korrespondierenden Vektors im Bitletbereich, als effizienter. Wenn $[m, \{(b_0, z_0), ..., (b_k, z_k)\}]$ die Bitletrepräsentation des Vektors $v \in V^m$ ist, so ist $[a \cdot m, \{(b_0, z_0), ..., (b_k, z_k)\}]$ die Bitletrepräsentation von v nach Raumeinbettung in den Raum $V^{a \cdot m}$ (siehe Korollar 6.1). Man sieht, bei der Raumeinbettung wird die Bitletrepräsentation bzw. die notwendige Bitletmenge nicht verändert und es muss lediglich die Raumdimension von m auf $a \cdot m$ angepasst werden.

6.5.1.1 Vergleichsoperation NOT

Wir beginnen unsere Betrachtungen mit der NOT-Operation ($\neg$) im $\mathbb{Z}^2$-Vektorraum. Wie gezeigt, muss zur Realisierung der Operation bei der Standardbitletrepräsentation

Repräsentationsvariante	Komplexität
dichtbesetzte lineare Musterrepräsentation	$O(n)$
dünnbesetzte lineare Musterrepräsentation	$O(n)$
Bitletrepräsentation	c

Tabelle 6.4: Realisierungkomplexität der NOT-Operation im $\mathbb{Z}^2$-Vektorraum

lediglich das Bitlets $b_{1,1}$ hinzugefügt oder gelöscht werden (siehe Abschnitt 6.4.4.2). Der Aufwand hierfür ist konstant (c). Zur Negierung eines Wiederholungsmusters im Zeitbereich (unabhängig ob dichtbesetzte oder dünnbesetzte Repräsentationsform) muss das Muster vollständig durchlaufen werden, um die einzelnen Positionen bzw. Belegungen des Musters zu negieren. Dies wiederum führt zu dem Aufwand $O(n)$, wobei n die Dimension des Vektors bzw. die Länge des Wiederholungsmusters ist. Tabelle 6.4 zeigt die Aufwände in Übersicht.

Im $\mathbb{Z}$-Vektorraum konnte lediglich eine eingeschränkte Form einer NOT-Operation angegeben werden (siehe Abschnitt 6.4.4.2). Für diese alternative NOT-Operation ($\overset{z}{\neg}$) ist jedoch eine Betrachtung des Aufwands nicht sinnvoll. Wir gehen bei der NOT-Operation davon aus, dass entweder die Bitletrepräsentation des unnegierten bzw. negierten Wiederholungsmusters im Bitletbereich verwendet wird. Ebenso gehen wir davon aus, dass diese beiden Bitletrepräsentationen bereits vor dem Vergleichszeitpunkt erzeugt werden. Somit fällt zum Vergleichszeitpunkt kein konkreter Aufwand an bzw. kann ebenfalls als konstant betrachtet werden. Letzlich kann das Vorgehen bei der alternativen NOT-Operation auch bei Wiederholungsmustern im Zeitbereich angewandt werden, d.h. zum Vergleichszeitpunkt liegt sowohl das negierte als auch unnegierte Wiederholungsmuster bereits vor. Unter dieser Annahme ist der Aufwand der NOT-Operation im Zeitbereich dann ebenfalls konstant.

Zusammenfassend betrachtet erweist sich das Vorhalten des negierten und des unnegierten Wiederholungsmusters bzw. Bitletrepräsentation als grundsätzlich gangbarer Weg. Allerdings führt das Vorhalten jeweils der negierten und unnegierten Version zu einem entsprechenden Anstieg des Repräsentationsumfangs, was sich als nachteilig erweisen kann. Besser bzw. für die Zukunft wünschenswerter wären hier jedoch Realisierung der NOT-Operation im $\mathbb{Z}$-Vektorraum, wie diese für den $\mathbb{Z}^2$-Vektorraum vorgestellt wurde. Diese weist ebenfalls einen konstanten Aufwand auf, ohne hierfür jedoch das negierte und unnegierte Muster vorhalten zu müssen.

6.5.1.2 Vergleichsoperation $\{\oplus, \equiv, \cup\}$

Die Anzahl der zur Darstellung notwendigen Bitlets bleibt bei der Raumeinbettung gleich und somit auch die Anzahl der notwendigen Einzelvergleiche bei einer Vergleichsoperation. Ein Anstieg der zur Repräsentation notwendigen Bitlets alleine aufgrund der Vereinheitlichung der Musterlängen bzw. Raumeinbettung besteht somit nicht. Hieraus

folgt, dass der Vergleich zweier Bitletrepräsentationen $v_1 \in V^n$ und $v_2 \in V^m$ im Worst-Case zu der folgenden Anzahl an Einzelvergleichen führt:

$$\boxed{((\beta'_n + \beta'_m) \cdot (n + m))}$$

Der hintere Term ergibt sich aus der Überlegung, dass nach der Raumeinbettung nur maximal n-Bitlets aus V^n und m-Bitlets aus V^m eine Nicht-Null-Belegung im gemeinsamen Raum $V^{m \cdot n}$ aufweisen können und beim Vergleich berücksichtigt werden müssen. Dies ergibt sich aufgrund der induktiven Eigenschaft von Bitletbasen, die dafür sorgt, dass die Basisvektoren bzw. Bitlets etwa aus dem Raum V^n ausreichend sind, um das gleiche Muster im Raum $V^{m \cdot n}$ zu repräsentieren.

Da es sich bei der Bitletrepräsentation um eine dünnbesetzte Repräsentationsform handelt, werden in der Regel weniger Bitlets zur Repräsentation eines Musters benötigt, als theoretisch mit den Raumdimensionen zur Verfügung stehen. Dies wird im vorderen Teil des Aufwandsterms mit den Parametern β'_n und β'_m berücksichtigt. Das heißt, die beiden Parameter stellen den *Besetzungsgrad* der Bitletmenge einer Bitletrepräsentation dar, d.h. das normierte Verhältnis der Anzahl der Bitlets zur Dimension des Vektorraums mit $0 < \beta' \leq 1$. Der Besetzungsgrad der Bitletmenge hängt davon ab, wie gut das periodische Verhalten eines Wiederholungsmusters durch eine Bitletbasis repräsentiert wird. Je besser das periodische Verhalten der Bitlets zu dem Verhalten des Musters passt, umso weniger Bitlets werden zur Repräsentation des Musters im Bitletbereich benötigt und umso kleiner wird der Besetzungsgrad.[13]

Tabelle 6.5 zeigt die resultierende Anzahl paarweiser Einzelvergleiche (im Worst-Case Fall, d.h. für die Periodenlänge der Muster gilt $LCP(n, m) = n \cdot m$) für die verschiedenen Repräsentationsvarianten (vgl. Abschnitt 6.1.3). Wenn wir zur Vereinfachung davon ausgehen, dass die Besetzungsgrade β_m, β_n der Muster und die Besetzungsgrade β'_m, β'_n der Bitletrepräsentation jeweils in Summe gleich sind, zeigt sich der signifikante Unterschied am hinteren Term. Während die beiden linearen Musterrepräsentationen hier zu $(n \cdot m)$ Einzelvergleichen führen, weist die Bitletrepräsentation lediglich $(n + m)$ Einzelvergleiche auf, was einer signifikant geringeren Anzahl entspricht. Wir betrachten dies für ein Einzelbeispiel einer Chemotherapie der Länge von 29 Tagen und dem Muster für *"Jeden 1.Tag im Monat"* mit Periodenlänge von 400 Jahren (= 146.097 Tage) die ein LCP von $lcp(29, 146.097) = 4.236.813$ aufweisen. Bei einer linearen Musterrepräsentation der Beispiele würden sich 4.236.813 Einzelvergleiche ergeben. Im Gegensatz dazu ergeben sich bei der Bitletrepräsentation $29 + 146.097 = 146.126$ Einzelvergleiche. Die Bitlet-Transformation bzw. Bitletrepräsentation ermöglicht gegenüber linearen

[13] Aus diesem Zusammenhang folgt, dass der Besetzungsgrad β eines Wiederholungsmusters und der Besetzungsgrad β' der Bitletmenge in gewisser Beziehung zueinander stehen. So existieren Muster die im Bitletbereich die gleichen Anzahl an Bitlets aufweisen wie Stützstellen im Zeitbereich. Genauso gibt es aber auch Muster bei denen weniger Bitlets als Stützstellen im Zeitbereich benötigt werden und andersherum. Es hängt letztlich von dem periodischen Verhalten der Muster und wie gut dieses durch eine Bitletbasis repräsentiert wird, welcher der Besetzungsgrade der Größere ist (siehe Abschnitt 6.6).

Musterrepräsentationen somit eine signifikant effizientere Realisierung unbeschränkter Vergleichsoperationen.

Repräsentationsvariante	#Einzelvergleiche
dichtbesetzte lineare Musterrepräsentation	$n \cdot m$
dünnbesetzte lineare Musterrepräsentation	$(\beta_n + \beta_m) \cdot (n \cdot m)$
Bitletrepräsentation	$(\beta'_n + \beta'_m) \cdot (n + m)$

Tabelle 6.5: Realisierungsaufwände bei unbeschränkten Vergleichsoperationen für unterschiedliche Repräsentationsvarianten

6.5.1.3 Vergleichsoperation ∩

Die Realisierung der Vergleichsoperation ∩ unterscheidet sich maßgeblich von der anderer Vergleichsoperationen. Hierbei wird die Realisierung der Vergleichsoperation ∩ auf das Problem der Lösung linearer Gleichungssysteme übertragen. Die zur Lösung solcher Gleichungssysteme bekannten Verfahren weisen im Allgemeinen polynomiale Komplexitäten bezüglich der Koeffizientenmatrix $n \times n$ auf, wie etwa das Gauß-Verfahren mit $O(n^3)$ [58]. Modernere, iterative Verfahren können gegenüber den direkten Verfahren hier zu besseren Ergebnissen führen. So kann etwa das Krylow-Unterraum-Verfahren bei großen dünnbesetzten Matrizen mit m Nicht-Null-Matrixeinträgen hier auch lineare Komplexität $O(m)$ erzielen [74].

Aufgrund dieser Komplexitäten der Lösungsverfahren im Allgemeinen, haben wir im vorgestellten Realisierungsansatz den Umfang des Gleichungssystems zuvor reduziert. Dies führt dazu, dass nicht mehr alle Zeilen und Spalten der Koeffizientenmatrix betrachtet werden müssen. Stattdessen ermittelt das Verfahren nur die "relevanten" Zeilen und Spalten, die in den Mengen Kund J hinterlegt werden. Die Lösung des sich ergebenen reduzierten Gleichungssystems ist dann polynomial in $max(\#J, \#K)$, d.h. polynomial bzgl. der Anzahl der Elemente der (größeren) Menge K oder J. Welches Verfahren konkret dann zur Lösung des (reduzierten) linearen Gleichungssystems anzuwenden gilt, wird in der vorliegenden Arbeit nicht weiter betrachtet. Allerdings empfehlen sich hierzu zukünftige Untersuchungen, um das ganze Vorgehen zur Realisierung der AND-Operation zu optimieren.

Der Aufwand bei der Realisierung der Vergleichsoperation ∩ ist direkt abhängig von der Güte der Reduzierung des Gleichungssystems.[14] Dieser Zusammenhang ist insbesondere für die Menge K wesentlich. Wie in Abschnitt 6.4.4.3 gezeigt, bedarf es zur Reduzierung des Gleichungssystems auch einer Reduzierung des eigentlich unbekannten Ergebnisvektors im Bitletbereich. Die Reduzierung des Ergebnisvektors stellt den

[14] Der Aufwand des Reduzierungsverfahrens selber wollen wir an dieser Stelle vernachlässigen, da dieser Aufwand gegenüber dem Aufwand zur Lösung des Gleichungssystems nicht wesentlich ins Gewicht fällt.

Knackpunkt des ganzen Realisierungsverfahrens dar. Je weniger Informationen wir besitzen, umso größer wird die Menge K und desto länger wird später das Verfahren zur Lösung des reduzierten Gleichungssystems benötigen. Falls wir kein anwendungsspezifisches Wissen besitzen, enthält die Menge K alle Koordinaten des Ergebnisvektors, da wir in diesem Fall davon ausgehen müssen, dass alle Koordinaten grundsätzlich Nicht-Null-Koordinaten sein können. In diesem Fall erzielen wir allerdings auch keine Reduzierung des Gleichungssystems und somit des Aufwands. An dieser Stelle sind kaum Aussagen möglich, ob in den konkreten Anwendungsgebieten ein entsprechendes Wissen vorliegt bzw. gesammelt werden kann, die eine Reduktion des Ergebnisvektors vor der Vergleichsoperation ermöglicht. Hierzu gilt es in zukünftigen Arbeiten entsprechende empirische Untersuchungen durchzuführen.

Zusammenfassend kann festgehalten werden, dass im Vergleich zur Vergleichsoperation $\cap$ im Zeitbereich (siehe Tabelle 6.5) das vorgestellte Realisierungsverfahren im Bitletbereich – in der heutigen Version – unterlegen ist. Dennoch, mit der vorliegenden Arbeit ist die grundsätzliche Realisierbarkeit der Vergleichsoperation $\cap$ im Bitletbereich gezeigt worden. Darüber hinaus wurden auch die "Stellschrauben" identifiziert, die in weiteren Arbeiten betrachtet werden sollten, um das Verfahren weiter zu optimieren. Dazu zählen auch Untersuchungen zu weiteren alternativen Lösungswegen zur Realisierung der Vergleichsoperation $\cap$.

6.5.2 Eingeschränkte Auswertung von Vergleichsoperationen

Die eingeschränkte Auswertung von Vergleichsoperationen innerhalb eines Ereignishorizonts bildete eine wesentliche Anforderung an die operationale Unterstützung periodischer Spezifikationen. Im Folgenden werden wir zunächst die eingeschränkte Vergleichsauswertung von Wiederholungsmustern im Zeitbereich näher betrachten, bevor wir diese mit der bitletbasierten Realisierung vergleichen.

6.5.2.1 Eingeschränkte Auswertung bei linearer Musterrepräsentation

Bei bitletbasierten Vergleichsoperationen haben wir zwei Realisierungsformen der eingeschränkten Vergleichsauswertung unterschieden, die a priori und die a posteriori Realisierung (siehe Abschnitt 6.4.5.2). Die gleichen Realisierungsformen gilt es auch bei Wiederholungsmustern im Zeitbereich zu unterscheiden.

Bei der **a posteriori Realisierung** findet die Einschränkung des Vergleichsergebnisses nach der Ausführung der Vergleichsoperation $\otimes$ zwischen Wiederholungsmustern r_1, r_2 statt, d.h. $EH_s^e(r_1 \otimes r_2)$. Dies bedeutet, dass zunächst die Vergleichsoperation durchgeführt wird und anschließend der mit dem Ereignishorizont korrespondierende Teilbereich des Musters überprüft wird. Wie leicht nachzuvollziehen ist, bietet sich diese Variante insbesondere dann an, wenn die Länge des Ereignishorizonts größer als die Länge des

resultierenden Vergleichsmuster ist bzw. dem LCP der beiden zu vergleichenden Muster. Abbildung 6.13 zeigt ein entsprechendes Beispiel.

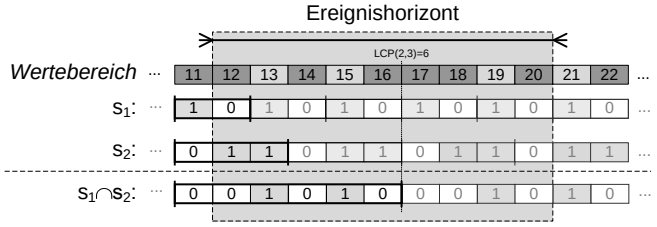


Abbildung 6.13: Eingeschränkte Vergleichsoperation bei Wiederholungsmustern und Ereignishorizont > Musterlänge

Bei der a posteriori Realisierung bleibt die grundlegende Vergleichsausführung unberührt. Das heißt, es gelten im Wesentlichen die gleichen Realisierungsaufwände für die Vergleichsoperation wie im vorherigen unbeschränkten Fall.

Bei der **a priori Realisierung** findet die Einschränkung des Vergleichsergebnisses vor der Ausführung der Vergleichsoperation $\otimes$ statt, d.h. $EH_s^e(r_1) \otimes EH_s^e(r_2)$. Dies bedeutet, dass zunächst die mit dem Ereignishorizont kkorrespondierenden Teilbereiche des Musters bestimmt werden und anschließend die Vergleichsoperation durchgeführt wird. Diese Variante bietet sich insbesondere dann an, wenn die Länge des Ereignishorizonts kleiner als die Länge des resultierenden Vergleichsmuster ist. Abbildung 6.13 zeigt ein entsprechendes Beispiel.

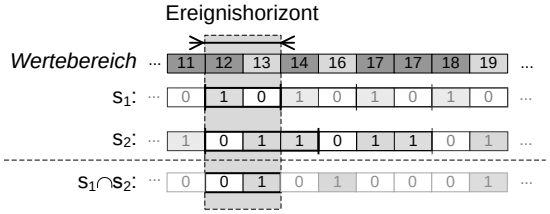


Abbildung 6.14: Eingeschränkte Vergleichsoperation bei Wiederholungsmustern und Ereignishorizont < Musterlänge

Allerdings führt die a priori Realisierung zu einer grundlegenden Veränderung in der Vergleichsausführung. Genauer gesagt, in dieser Realisierungsform besteht das Problem der Vereinheitlichung unterschiedlich langer Wiederholungsmuster nicht. Aufgrund der vor der Vergleichsoperation stattfindenden "Beschneidung" der Wiederholungsmuster auf den Ereignishorizont führt dies zu Mustern bzw. "Teilmustern" gleicher Länge, die nicht mehr vereinheitlicht werden müssen. Folglich ergibt sich bei der a priori Variante

ein anderer Vergleichsaufwand als bei der unbeschränkten Auswertung. Gegeben zwei Muster der Länge n, m und die Länge EH des Ereignishorizonts mit $EH < lcp(m, n)$, so ergibt sich im Worst-Case ($lcp(m, n) = m \cdot n$) die folgenden Aufwände bzw. Anzahl an notwendigen Einzelvergleichen:

- EH = bei der dichtbesetzten linearen Musterrepräsentation
- $(\beta_n + \beta_m) \cdot EH$ = bei der dünnbesetzten linearen Musterrepräsentation

Aufgrund dessen, dass gilt $EH < m \cdot n$, führt dies dazu, dass sich der Aufwand der a priori Realisierung zur eingeschränkten Vergleichsauswertung gegenüber der unbeschränkten Auswertung als erheblich effizienter erweist. Das heißt, die Anzahl der Einzelvergleiche ist hier deutlich geringer.

6.5.2.2 Eingeschränkte Auswertung bei Bitletrepräsentationen

Die eingeschränkte Auswertung im Bitletbereich führt zu dem Problem, dass zur Ergebnisinterpretation eines eingeschränkten Vergleichs eine Rücktransformation in den Zeitbereich vorgenommen werden muss. Dieses Lokalisationsproblem der Bitletrepräsentation bzw. Bitlet-Transformation stellt ein generelles Problem entsprechender linearer Abbildungen bzw. Transformationen dar. Allerdings tritt dieses Problem nur in der a priori Variante mit $EH < n$ auf mit n als Raumdimension. In der a posteriori Variante mit $EH > n$ wird weiterhin das ganze Muster bzw. die vollständige Bitletrepräsentation betrachtet, was zu einem Vergleichsaufwand wie im unbeschränkten Fall führt.

Der Vergleichsaufwand in der a priori Variante ist vom zusätzlichen Aufwand gekennzeichnet, der aufgrund der Rücktransformation anfällt. Aus unseren Betrachtungen zur Fast-Bitlet-Transformation wissen wir bereits, dass die Rücktransformation in den Zeitbereich pro Matrixspalte bzw. korrespondierenden Wert im Zeitbereich den Aufwand $log(n)^2$ aufweist, mit n als Dimension des Raums. Die Anzahl der für den Ereignishorizont relevanten Bitlets wird mit EH' bezeichnet. Der Vergleich der Bitletrepräsentation $v_1 \in V^n$ und $v_2 \in V^m$ führt somit im Worst-Case zu dem Aufwand:

$$\boxed{((\beta'_n + \beta'_m) \cdot EH') \cdot (EH \cdot log(n \cdot m)^2)}$$

Der vordere Term beschreibt die, aufgrund der Einschränkung durch den Ereignishorizont, reduzierte Menge an zu vergleichenden Bitlets. Der hintere Term beschreibt die Rücktransformation für jeden Wert aus dem Ereignishorizont, d.h. die Länge EH gibt die Anzahl der einzelnen notwendigen Rücktransformationen pro Wert an.

Tabelle 6.6 zeigt die Ergebnisse bzw. die Aufwände der eingeschränkten Vergleichsauswertung der verschiedenen Repräsentationsvarianten in Übersicht. Hierbei zeigt sich, dass in der a posteriori Variante die gleiche Situation wie im unbeschränkten Fall vorliegt und sich die Bitletrepräsentation als die effizientere Repräsentationsform erweist.

Repräsentationsvariante	#Einzelvergleiche	
	A Posteriori ($EH > m \cdot n$)	A Priori ($EH < m \cdot n$)
dichtbesetzte lineare Musterrepräsentation	$n \cdot m$	EH
dünnbesetzte lineare Musterrepräsentation	$(\beta_n + \beta_m) \cdot (n \cdot m)$	$(\beta_n + \beta_m) \cdot EH$
Bitletrepräsentation	$(\beta'_n + \beta'_m) \cdot (n + m)$	$((\beta'_n + \beta'_m) \cdot EH') \cdot (EH \cdot log(n \cdot m)^2)$

Tabelle 6.6: Realisierungsaufwände bei eingeschränkten Vergleichsoperationen für unterschiedliche Repräsentationsvarianten

In der a priori Variante ergibt sich hingegen eine andere Situation. In dieser Realisierungsvariante wird bei linearer Musterrepräsentation keine Vereinheitlichung der Musterlängen benötigt, da aufgrund der Beschränkung auf den Ereignishorizont gleich lange Muster entstehen. Der Aufwand entfällt, der sonst durch die Vereinheitlichung entsteht. Im Gegensatz dazu, fällt aufgrund des Lokalisationsproblems der Bitletrepräsentation zusätzlicher Aufwand durch die notwendigen Rücktransformation an. Infolgedessen erweist sich sich die eingeschränkte Vergleichsauswertung bei linearer Musterrepräsentation in der a priori Variante als deutlich effizienter als die vorgestellte bitletbasierte Realisierung.

Dieses Ergebnis der Realisierungsaufwände muss für den praktischen Einsatz der Bitletrepräsentation bei einer eingeschränkten Vergleichsauswertung als kritisch betrachtet werden. Wir sind in unseren bisherigen Überlegungen stets davon ausgegangen, dass wir in der Praxis häufig Wiederholungsmuster mit langen bis sehr langen Periodenlängen vorfinden werden. Je nach Anwendungsgebiet wird somit die a priori Realisierungsvariante bei eingeschränkten Vergleichen eher die Regel als die Ausnahme sein. Damit ist jedoch, wie oben gezeigt, eine lineare Musterrepräsentation der Bitletrepräsentation vorzuziehen.

Zusammenfassend kann festgestellt werden, dass bei einer unbeschränkten und bei der eingeschränkten, a posteriori Auswertung von Vergleichsoperationen (speziell bei der Operation $\cup$) die Stärken der Bitletrepräsentation eindeutig zum Tragen kommen. Bei der eingeschränkten a priori Auswertung hingegen, kommen diese Stärken gegenüber alternativen Repräsentationen nicht zum Tragen. Das hierbei zugrundeliegende "Lokalisationsproblem" ist kein spezifisches Problem der Bitlettransformation, sondern ein grundlegendes Problem, das auch bei vergleichbaren Transformationen besteht. Es handelt sich hierbei um kein triviales Problem. Allerdings sind in der Literatur Verfahren bekannt, wie diesem Problem begegnet werden kann (siehe Abschnitt 6.6). Die Anwendbarkeit und ggf. Übertragung dieser Verfahren auf die Bitlettransformation stellt noch eine offene Fragestellung dar. Wir überlassen es jedoch zukünftigen Arbeiten dieses Lokalisationsproblem der Bitlettransformation zu lösen, was einen weiteren Schritt

darstellt, auf dem Weg zu einem umfassenden Einsatz der Bitletrepräsentation in der Praxis. Die in dieser Arbeit vorgestellten Erkenntnisse und Methoden bilden hierfür eine sehr gute Grundlage.

6.6 Ausblick

In den vorherigen Abschnitten haben wir die Bitletrepräsentation vorgestellt, aber auch ihre Möglichkeiten und Probleme aufgezeigt. Im Folgenden möchten wir erste Überlegungen zu den wesentlichen, noch zu klärenden Problemen der Bitletrepräsentation vorstellen.

Kompressionsverhalten

Die in diesem Kapitel vorgestellte Bitlet-Transformation bzw. Bitletrepräsentation kann auf höherer Ebene als eine (verlustfreie) Komprimierung von Wiederholungsmustern verstanden werden. Bei geeigneter Wahl der Basisvektoren in Bezug auf das periodische Verhalten eines Wiederholungsmusters kann eine im Umfang verringerte Darstellung des Wiederholungsmusters erreicht werden. In Abhängigkeit von der Güte der ausgewählten Basisvektoren in Bezug auf das periodische Verhalten des Musters ergibt sich ein entsprechender Komprimierungsfaktor. Dieser Faktor drückt sich dann in der Anzahl der benötigten Nicht-Null-Koordinaten zur Darstellung des Musters aus. Das Kompressionsverhalten der Bitletrepräsentation wird somit unmittelbar von den verwendeten Bitlets bzw. deren periodischem Verhalten beeinflusst. Um dieses Verhalten zu optimieren, d.h. eine kompaktere Repräsentation des Wiederholungsmusters im Bitletbereich zu erhalten, gilt es, neue Bitletbasen bzw. musterangepasste Bitletbasen einzuführen. In der Praxis entstehen sehr lange Muster meist aufgrund von Ausnahmen in periodischen Spezifikationen. Das wiederum führt dazu, dass das "Ursprungsmuster", d.h. das Muster ohne Ausnahmen, im langen Muster sich wiederholt und sich nur an wenigen Stellen aufgrund der Ausnahmen unterscheidet. Dieses Verhalten bzw. Eigenschaft von Mustern kann etwa durch problemangepasste Bitlets nachgebildet werden, um eine kompaktere Repräsentationen zu erzielen.

Das Problem an dieser Stelle ist jedoch, dass die Anforderungen an eine Bitletbasis sehr hoch sind bzw. sehr spezifische Eigenschaften (z.B. Induktivität) aufweisen muss. In zukünftigen Arbeiten gilt es zu untersuchen, ob solche Bitletbasen quasi "auf Knopfdruck" und in Abhängigkeit bestimmter Mustereigenschaften konstruiert werden können. In der Theorie zur Wavelet-Transformation, beispielsweise, besteht ein solches Verfahren zur Entwicklung problemangepasster Wavelets, die so genannten Wavelet-Pakete [44, 33].

Lokalisationsproblem

Das Hauptproblem der Bitletrepräsentation bei der eingeschränkten, a priori Auswertung von Vergleichsoperationen bildet das Lokalisationsproblem. So können für einen bestimmten Zeitabschnitt relevante Bitlets im Bitletbereich identifiziert werden. Das Lokalisationsproblem besagt jedoch, dass aus der Existenz der identifzierten Bitlets (mit

Nicht-Null-Belegung) keine direkten Schlussfolgerungen auf die Belegungen des Wiederholungsmusters im entsprechenden Zeitabschnitt geschlussfolgert werden können. Wie erwähnt, weisen andere Transformationen, wie etwa die Fourier-Transformation das gleiche Problem auf. Das Lokalisationsproblem hat bei der Fourier-Transformation unter anderem zu der *gefensterten Fourier-Transformation [13, 47]* bzw. bei der Wavelet-Transformation zur *Multiskalenanalyse* geführt [56, 33, 13, 44]. Beides stellen Methoden bzw. Verfahren dar, dem Lokalisationsproblem zu begegnen. Genauer gesagt ermöglichen solche Verfahren deutlich exaktere Aussagen, wie im Frequenzbereich oder Waveletbereich vorliegende Werte im Zeitbereich wirken bzw. entsprechende Schlussfolgerung abgeleitet werden können.

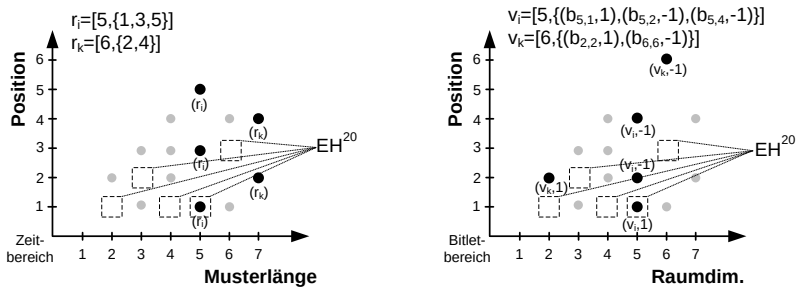
Die direkte Übertragung der Verfahren ist jedoch nicht ohne weiteres möglich. So wird in den Verfahren etwa von konstanten bzw. über alle betrachteten Muster bzw. Funktionen gleichbleibende Periodenlängen ausgegangen. Ebenso steht die Analyse des einzelnen Musters im Vordergrund. Die Einsetzbarkeit des Verfahrens bei vielen verschiedenen Wiederholungsmustern mit unterschiedlichen Periodenlängen bzw. Raumdimensionen sowie der Eigenschaft der Induktivität von Bitletbasen muss somit zunächst erst untersucht werden.

Indizierte Unterstützung von Vergleichsoperationen

Die bisherigen Betrachtungen von Vergleichsoperationen zielen auf den einzelnen Vergleich (zweier) Wiederholungsmuster bzw. periodischer Spezifikationen ab. Wie erwähnt, bedarf es in der Praxis jedoch auch einer effizienten Realisierung von Vergleichen bei großer Anzahl an periodischen Spezifikationen etwa durch Unterstützung von entsprechenden Index- und Zugriffsstrukturen.

Allerdings sind solche Index- und Zugriffsstrukturen abhängig davon, welche Struktur die Bitlets aufweisen, genauer gesagt, welche Form die Indexfunktion für Bitlets aufweist. Wie in der allgemeinen Theorie vorgestellt, weisen indizierte Bitletbasen eine Indexfunktion $I(b, B_m)$ auf, die jedes Bitlet b sowie deren periodische Verlängerungen eindeutig und über alle Raumdimensionen hinweg identifiziert (siehe Abschnitt 6.2.7.1). Für die (indizierte) Standardbitletbasis haben wir eine entsprechende Indexfunktion definiert (siehe Abschnitt 6.2.7.2). Hierbei kann jedes Bitlet durch zwei Parameter eindeutig identifiziert werden. Somit kann eine Standardbitletrepräsentation als Menge zweidimensionaler Punkte im Bitletbereich verstanden werden (siehe Abbildung 6.15b) und etwa auf bekannte Index- oder Zugriffsstrukturen für solche zweidimensionale Werte zurückgegriffen werden.

Die Grundidee zur indizierten Auswertung von eingeschränkten Vergleichsanfragen besteht darin, dass für einen gegebenen Ereignishorizont nur einige wenige Bitlets einer Bitletrepräsentation relevant sind (siehe Abschnitt 6.5.2). Auf Grundlage einer entsprechenden Index- oder Zugriffsstruktur könnten die Spezifikationen ermittelt werden, die für den Ereignishorizont relevante Bitlets aufweisen. Alle anderen Spezifikationen sind nicht für den Ereignishorizont relevant und müssten nicht berücksichtigt werden. Abbildung 6.15b zeigt dies anhand eines Beispiels für den Ereignishorizont EH^{20}, dessen



(a) Zeitbereich (b) Bitletbereich

Abbildung 6.15: Beispiel einer Zugriffstruktur im Zeitbereich und Bitletbereich

(potentiell) relevante Bitlets durch das gestrichelte Quadrat gekennzeichnet sind. Hierbei weist lediglich die Spezifikation v_i ein für den angegebenen Ereignishorizont relevantes Bitlet auf. Allerdings kommt an dieser Stelle erneut das Lokalisationsproblem zum Tragen. So können für einen Ereignishorizont relevante Bitlets zwar identifiziert, aber aufgrund der Bitlets alleine keine Aussagen im Zeitbereich getroffen werden. In der Treffermenge können somit Spezifikationen enthalten sein, die zwar relevante Bitlets aufweisen, aber in der vorliegenden Kombination für diesen Ereignishorizonts zu keinem Wert im Zeitbereich führen. Solche "False-Hits" gilt es, nachträglich zu bereinigen. Hierbei bildet weniger die False-Hit-Bereinigung das eigentliche Problem, sondern die Anzahl der relevanten Bitlets. So kann eine vergleichbare Indexstruktur auch für eine "normale" (dünnbesetzte) lineare Musterrepräsentation definiert werden. Jedes (zweidimensionale) Tupel repräsentiert die Musterlänge und die Position der Nicht-Null-Belegung (siehe Abbildung 6.15a). Auch hier kann für einen Ereignishorizont eine Menge relevanter Tupel identifiziert werden. Das Problem ist hierbei, dass die Anzahl der relevanten Bitlets und Anzahl der relevanten Tupel für Wiederholungsmuster jedoch gleich ist (siehe Abbildung 6.15). Das heißt, die Selektivität der Zugriffsstruktur ist in beiden Repräsentationsformen identisch. Lediglich aufgrund der potentiellen Kompressionseigenschaft der Bitletrepräsentation können sich Unterschiede im Umfang solcher Strukturen ergeben.

Mit der Begegnung des Lokalisationsproblems bzw. dessen "Entschärfung" ergibt sich gleichzeitig die Möglichkeit die Selektivität einer solchen Zugriffsstruktur zu erhöhen. Je besser Bitlets bzw. deren Auswirkungen im Zeitbereich lokalisiert werden können, umso geringer ist die Anzahl der für einen Ereignishorizont relevanter Bitlets und umso höher damit die Selektivität entsprechender Strukturen. Darüber hinaus gilt es zu überlegen, ob auf Basis von Bitletrepräsentationen etwa zweidimensionale Wertebereiche definiert werden können, in denen "klassische" Nachbarschaftsbeziehungen im Zusammenhang mit relevanten Bitlets stehen. Wie in Abbildung 6.15b zu sehen ist, befinden sich die für den Ereignishorizont relevanten Bitlets sehr "verstreut" in der Struktur. Um die Anzahl an Zugriffen zu reduzieren, könnten auf Basis solcher Nachbarschaftsbeziehungen konsolidierte Zugriffe erfolgen.

Allerdings gilt es für alle weiteren Schritte und Überlegungen zunächst dem nicht trivialen Lokalisationsproblem zu begegnen. Auf Basis der entsprechenden Lösung könnten dann Index- und Zugriffsstrukturen untersucht werden, die eine höhere Selektivität gegenüber alternativen Repräsentationsformen aufweisen oder semantisch relevante Nachbarschaftsbeziehungen definieren können.

6.7 Zusammenfassung

In diesem Kapitel haben wir mit der Bitletrepräsentation einen neuen Lösungsansatz vorgestellt, hin zu einer kompakten Repräsentation und effizienten operationalen Realisierung von Vergleichsoperationen zwischen Wiederholungsmustern. Ausgangspunkt der Betrachtungen bildete die Schwächen heutiger Verfahren bei Vergleichsoperationen zwischen Wiederholungsmustern unterschiedlicher Länge. Es zeigt sich, dass das Kernproblem die notwendige Vereinheitlichung unterschiedlich langer Wiederholungsmuster ist und der damit verbundene starke Anstieg an durchzuführenden Einzeloperationen bei einem Vergleich. Mit der Bitletrepräsentation wurde in der vorliegenden Arbeit eine Repräsentationsform vorgestellt, die von einem solchen Anstieg des Aufwands nicht betroffen ist.

In der allgemeinen Theorie zur Bitletrepräsentation werden Wiederholungsmuster als Vektoren entsprechend dimensionaler Vektorräume verstanden. Hierbei zeigte sich, dass Vereinheitlichung von Wiederholungsmuster unterschiedlicher Länge auch bei dieser Betrachtungsweise besteht. Genauer gesagt stellt die Raumeinbettung von Vektorräumen unterschiedlicher Dimension das korrespondierende Problem im Vektorraum dar, wie es für Wiederholungsmuster im Zeitbereich besteht. Auf Grundlage der eingeführten Bitletbasen mit ihren spezifischen Eigenschaften kann mit der Bitletrepräsentation eine Repräsentationsform angegeben werden, bei der eine Raumeinbettung effizient repräsentiert werden kann. Im Gegensatz zu Wiederholungsmustern, führt eine Verlängerung bzw. die korrespondierende Raumeinbettung zu keiner Erhöhung der zu repräsentierenden Werte einer Bitletrepräsentation. Im Gegenteil, die Anzahl an zu repräsentierenden Werten bleibt gleich – unabhängig von der Dimension des Raums bzw. dem Grad der Verlängerung des korrespondierenden Musters.

Eine effiziente Transformation von Wiederholungsmustern im Zeitbereich in die Bitletrepräsentation im Bitletbereich und umgekehrt ist für den praktischen Einsatz unumgänglich. Theoretisch müssen für die Hin- und Rücktransformationen Matrizenmultiplikationen durchgeführt werden. Für eine effiziente Realisierung der Transformation empfiehlt es sich jedoch nicht, eine solche Multiplikation explizit durchzuführen, da in der Praxis schnell problematische Matrizendimensionen vorliegen können. Mit der vorgestellten (inversen) Fast-Bitlet-Transformation haben wir ein Verfahren vorgestellt, dass eine effiziente Transformation aus dem Bitletbereich in den Zeitbereich ermöglicht. Die Grundidee besteht darin, von den potentiell sehr großen Basismatrizen nur genau die Nicht-Null-Belegungen zu berücksichtigen. Wie wir zeigen konnten, führt ein solches

Vorgehen dazu, dass die vorgestellte inverse Fast-Bitlet-Transformation sich gegenüber der naiven Matrizenmultiplikation als erheblich effizienter erweist. Leider konnte für die Transformation vom Zeitbereich in den Bitletbereich kein effizienteres Verfahren als eine Matrizenmultiplikation gefunden werden. Hierfür werden weiterführende Untersuchungen in zukünftigen Arbeiten benötigt.

Das Ziel der Einführung der Bitletrepräsentation war es, die Schwächen bestehender Repräsentationsformen von Wiederholungsmustern bei der Realisierung von Vergleichsoperationen zu beheben. Mit der Stabilität bei Raumeinbettungen und der Fast-Bitlet-Transformation weist die Bitletrepräsentation bereits zwei wesentliche Grundlagen auf, den Schwächen bestehender Repräsentationsformen zu begegnen. Es zeigte sich jedoch, dass zur konkreten Realisierung von Vergleichsoperationen im Bitletbereich nicht unmittelbar das gleiche Vorgehen wie im Zeitbereich verwendet werden kann. Stattdessen gilt es, die Semantik der Vergleichsoperation auf die im Bitletbereich zur Verfügung stehenden Operationen nachzubilden.

Für die Vergleichsoperationen $\{\oplus, \equiv, \neg\}$ in dem Zahlenbereich $\mathbb{Z}^2$ konnten wir zeigen, dass deren Semantik direkt durch Operationen im Bitletbereich nachgebildet werden kann. Die Bitletrepräsentation ermöglicht hier gegenüber bisherigen Musterrepräsentationen eine signifikant effizientere Realisierung bei unbeschränkten Vergleichen. Leider stellen die Vergleichsoperationen im Zahlenbereich $\mathbb{Z}^2$ keine vollständige boolesche Funktionsbasis dar. Eine entsprechende Funktionsbasis ist jedoch essentiel, um beliebige (mengentheoretische) Beziehungen bzw. Vergleiche definieren zu können. Das Gleiche gilt auch für die Vergleichsoperation $\cup$ und die (eingeschränkte) NOT-Operation ($\overset{z}{\neg}$) im ebenfalls betrachteten Zahlenbereich $\mathbb{Z}$. Auch diese Operationen sind sehr effizient im Bitletbereich realisierbar. Leider stellt hierbei die NOT-Operation keine "echte" Negationsfunktion dar, sodass für eine vollständige boolesche Funktionsbasis zusätzlich die Vergleichsoperation $\cap$ benötigt wird.

Wie sich zeigt, ist die Vergleichsoperation $\cap$ nicht direkt im Bitletbereich realisierbar. Stattdessen gilt es zur Realisierung dieser Operation die den Räumen zugrundeliegenden Bitletbasen mit zu berücksichtigen. Wir haben gezeigt, dass es möglich ist, die Realisierung der Vergleichsoperation $\cap$ auf das Problem der Lösung linearer Gleichungssysteme zu übertragen. Jedoch weisen entsprechende Verfahren zur Lösung solcher Gleichungssysteme meist polynomiale Komplexitäten auf. Um diesen zu begegnen, konnten wir zeigen, dass das zu lösende Gleichungssystem aufgrund unseres Wissens zu den Bitletbasen z.T. erheblich reduziert werden kann. Dies führt dann zu einem erheblich geringeren Aufwand bei der Lösung des reduzierten Gleichungssystems. Allerdings ist die Güte der Reduzierung des Gleichungssystems unmittelbar auch von einem anwendungsspezifischen Wissen abhängig. Erst ein solches führt zu signifikanten Reduzierungen des Gleichungssystems. Ob in den konkreten Anwendungsgebieten ein entsprechendes Wissen vorliegt bzw. gesammelt werden kann, bleibt eine offene Fragestellung, für die empirische Untersuchungen in zukünftigen Arbeiten benötigt werden.

Im Gegensatz zu den obigen Ergebnissen bei unbeschränkter Vergleichsauswertung weist

die Bitletrepräsentation jedoch Probleme bei eingeschränkten Vergleichsauswertungen auf. Bei eingeschränkten Auswertungen findet ein Vergleich lediglich innerhalb eines bestimmten Auswertungszeitraums bzw. Ereignishorizonts statt. Es zeigt sich, dass hier die Bitletrepräsentation gegenüber den Repräsentationen bzw. Realisierungen eingeschränkter Vergleiche im Zeitbereich (meist) unterlegen ist. Allerdings konnten wir zeigen, dass das zugrundeliegende "Lokalisationsproblem" kein spezifisches Problem der Bitlettransformation, sondern ein grundlegendes Problem ist, das auch bei vergleichbaren Transformationen besteht. Es handelt sich hierbei um kein triviales Problem. Allerdings sind in der Literatur Verfahren bekannt, wie diesem Problem begegnet werden kann. Die Anwendbarkeit und ggf. Übertragung dieser Verfahren auf die Bitlettransformation stellt jedoch noch eine offene Fragestellungen dar.

Zusammenfassend kann festgestellt werden, dass die Bitletrepräsentation einen ersten Schritt in die richtige Richtung für den Umgang mit Wiederholungsmustern darstellt. Unter anderem konnten durch eine prototypische Realisierung der vorliegenden Konzepte die erwarteten Vorteile und Eigenschaften der Bitletrepräsentation bestätigt sowie erste Erfahrungen mit praktischen Beispielen gesammelt werden. Hier zeigte sich aber auch, dass es viele weitere Schritte braucht, um das Potential dieses Lösungsansatzes vollständig zu entfalten. Mit der in dieser Arbeit geschaffenen Grundlagen haben wir hierfür jedoch ein solides Fundament gelegt.

Teil IV

Proof-of-Concept Prototyp, Zusammenfassung und Anhang

7 Proof-of-Concept Prototyp

Für die im Rahmen dieser Arbeit vorgestellten Konzepte zur Beschreibung und Auswertung periodischer Spezifikationen wurde eine prototypische Implementierung vorgenommen. Ziel des Prototyps war es, sowohl die effiziente Realisierbarkeit und praktische Anwendbarkeit der (theoretischen) Einzelkonzepte und deren Zusammenspiel untereinander aufzuzeigen. Entstanden ist ein Stand-Alone Prototyp mit dem auf der logischen Grundlage des zyklenorientierten Zeitmodells und seiner algebraischen Notationsform komplexe periodische Spezifikationen beschrieben werden können (siehe Kapitel 3). Repräsentiert werden die Spezifikationen auf Grundlage von HSP-Bäumen, da diese die berechenbare Repräsentationsform des zyklenorientieren Zeitmodells darstellen (siehe Kapitel 4). Des Weiteren wurde die in Kapitel 5 vorgestellte Auswertung von HSP-Bäumen realisiert, sodass zum einen entsprechende Wiederholungsmuster abgeleitet und zum anderen aber auch in die entsprechende Bitletrepräsentation (siehe Kapitel 6) überführt werden können.

Im Folgenden werden wir in Abschnitt 7.1 zunächst eine grobe Einführung bzw. Gesamtübersicht zu dem Prototypen und seinen Funktionen geben. In Abschnitt 7.2 gehen wir dann detailiert auf den Funktionsumfang der einzelnen realisierten Konzepte ein. Im Anschluss wird in Abschnitt 7.3 ein Vorgehensmodell bei der Beschreibung periodischer Spezifikatione mit dem Prototypen vorgestellt. Den Abschluss bildet in Abschnitt 7.4 ein konkretes Anwendungsbeispiel an dem das Vorgehensmodell und die Handhabung des Prototypen verdeutlicht werden.

7.1 Gesamtübersicht

Abbildung 7.1 zeigt das "Look-and-Feel" des Prototypen in einer Gesamtansicht. Jede periodische Spezifikation wird in dem Prototyp als ein Fenster dargestellt, bestehend aus den drei Bereichen *Befehlszeile*, *HSP-Baumdarstellung* und *Auswertungsergebnis*.

In dem Bereich *Befehlszeile* auf der linken Seiten des Fenster können die Operationen der algebraischen Notationsform oder auch Labelausdrücke direkt eingegeben werden. Alle eingegebenen Operationen beziehen sich stets auf den im mittleren Bereich dargestellten HSP-Baum. Zusätzlich wird im Bereich unmittelbar oberhalb der Befehlszeile das Ergebnis der verwendeten Labelausdrücke, d.h. die Menge der korrespondierenden Knoten im HSP-Baum, dargestellt. Im Beispiel in Abbildung 7.1 führt der dargestellten Labelausdruck etwa zu der Knotenmenge $\{H_8\}$.

Im Bereich *HSP-Baumdarstellung* wird der HSP-Baum der periodischen Spezifikation dargestellt. Der Anwender kann jedoch wählen, ob er die periodische Spezifikation als

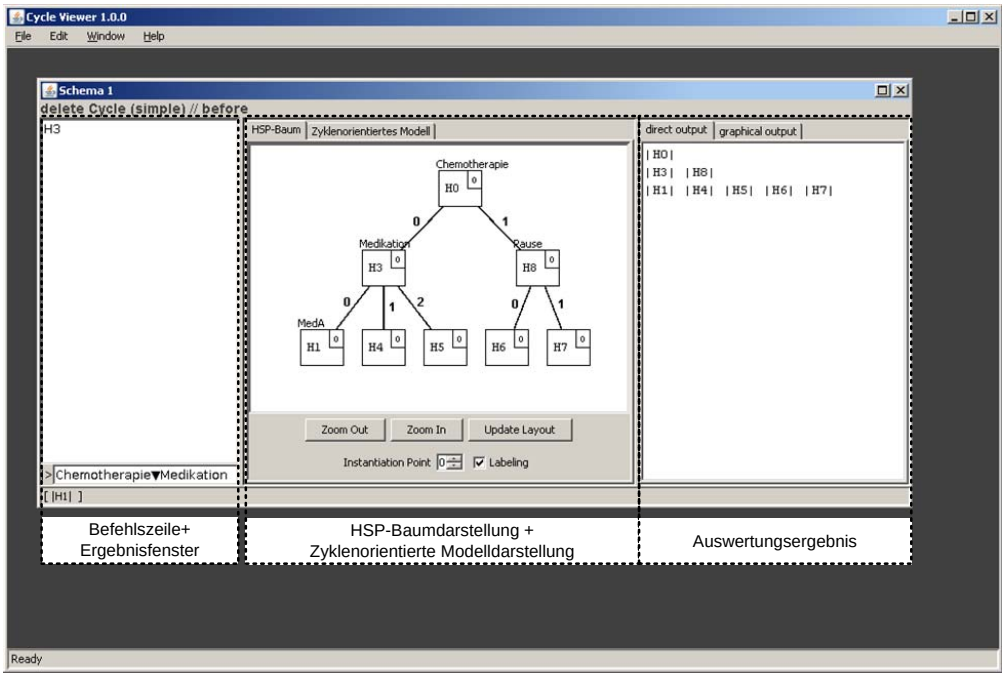


Abbildung 7.1: Gesamtansicht des Prototyps

HSP-Baum oder in Form des (logischen) zyklenorientierten Zeitmodells dargestellt haben möchte (siehe Reiter "Zyklenorientiertes Modell"). Beide Darstellungsformen sind möglich. In der HSP-Baumdarstellung können jedoch zusätzlich die Operationen der algebraischen Notationsform auf Basis einer grafischen Benutzerschnittstelle angewandt werden (siehe Abschnitt 7.2.1).

Der Bereich *Auswertungsergebnis* zeigt das resultierende Wiederholungsmuster und die Zwischenergebnisse pro Level bei der Auswertung des HSP-Baums.

7.2 Funktionsumfang und Bedienung des Prototyps

Nicht alle der im Rahmen der Arbeit vorgestellten Konzepte wurden vollumfänglich implementiert, sondern lediglich einige wesentliche. Die umgesetzten Konzepte zur Beschreibung komplexer periodischer Spezifikationen und zu deren Auswertung bzw. operationalen Unterstützung werden wir im Folgenden näher erläutern.

7.2.1 Beschreibung periodischer Spezifikationen

Die Operationen der algebraischen Notationsform können in die drei Gruppen *Labelausdrücke, Änderungsoperationen* und *Auswahloperation* unterteilt werden (siehe Kapi-

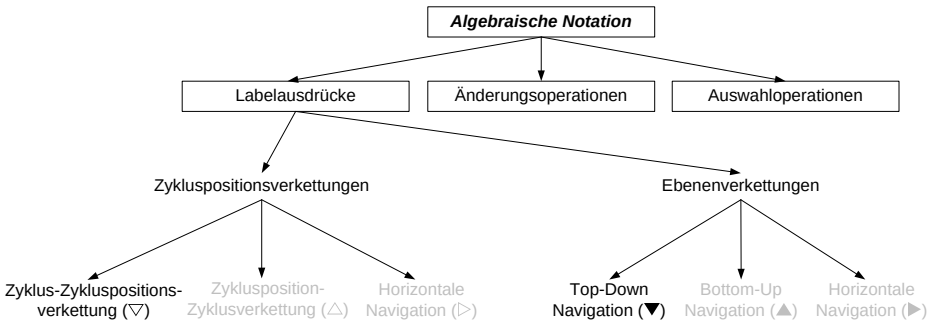


Abbildung 7.2: Implementierte Verkettungsfunktionen für Labelausdrücke

tel 3). Die realisierten Einzeloperationen der jeweiligen Gruppe und ihre Handhabung werden wir nachfolgend vorstellen.

7.2.1.1 Labelausdrücke und Labels

Zur Adressierung von (Teil-) Zyklen und Zykluspositionen wurden verschiedene Verkettungsoperationen eingeführt, die in ihrer kombinierten Verwendung zu entsprechenden Labelausdrücken führen. Abbildung 7.2 zeigt die realisierten Verkettungsfunktionen.

Labelausdrücke werden im Prototyp innerhalb der Befehlszeile eingegeben. Hierbei können diese entweder separat oder als Parameter einer Änderungs- oder Auswahloperation (siehe später) abgesetzt werden. Das Ergebnis des Labelausdrucks, d.h. die adressierte Menge an Knoten wird direkt oberhalb der Befehlszeile im Ergebnisfenster angezeigt. Labelausdrücke können, so wie in Abschnitt 3.4.1 gezeigt, direkt eingegeben werden, d.h. konkrete Labelbezeichnungen gefolgt von den entsprechenden Verkettungsfunktionen. Über die Tastenkombination `STRG+s` bzw. `STRG+d` können die Top-Down-Navigation ($\overset{x}{\blacktriangledown}$) bzw. die Zyklus-Zykluspositionsverkettung ($\overset{x}{\triangledown}$) eingegeben werden. Die optionale Positionsangaben x können in geschweiften Klammern unmittelbar nach der Verkettungsfunktion angegeben werden. Somit wird etwa der Labelausdruck `Chemotherapie▼Medikation` $\overset{1-14}{\triangledown} *$ in der Befehlszeile zu `Chemotherapie▼Medikation` $\triangledown\{1-14\}*$.

Die in Labelausdrücken bzw. in Änderungsoperationen verwendbaren Labels müssen (zuvor) in einem speziellen Repository hinterlegt bzw. angelegt werden. Das Labelrepository entspricht einem "kontrollierten Vokabular", was bei der Beschreibung periodischer Spezifikationen verwendet wird (siehe Abschnitt 3.4.1). In dem Repository müssen Labels entweder der Labelmenge *StructureSet* oder der Labelmenge *ActivitySet* zugeordnet werden, um deren konkrete Semantik festzulegen. (siehe Abschnitt 4.2.1.2). Das Repository wird im Prototypen über das Menü *Windows→Labels* aufgerufen. Abbildung 7.3 zeigt das Repository anhand eines Beispiels.

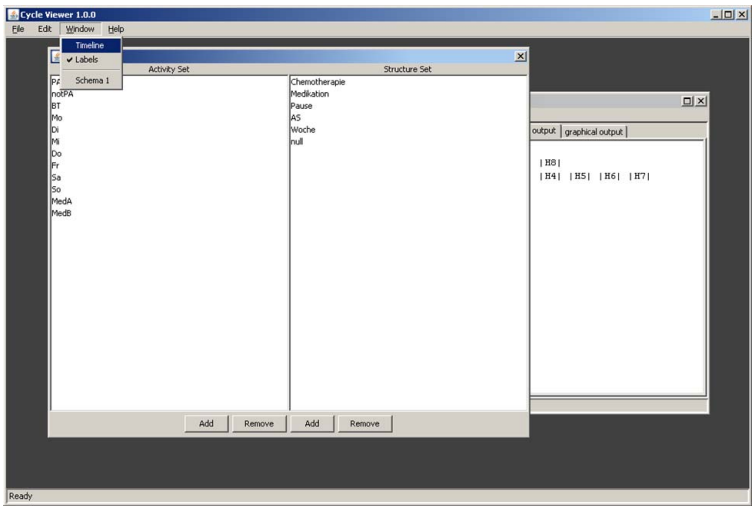


Abbildung 7.3: Repository zur Verwaltung der Labels

7.2.1.2 Änderungs- und Auswahloperationen

Zur Manipulation von (Teil-) Zyklen und Labels wurden verschiedene Änderungsoperationen vorgestellt (siehe Abschnitt 3.4.2.1). Abbildung 7.4 zeigt die im Prototyp realisierten Änderungsoperationen.

Zur Anwendung von Änderungs- und Auswahloperationen sind grundsätzlich zwei verschiedene Möglichkeiten im Prototyp realisiert. In der ersten Möglichkeit können Operationen als entsprechende Befehle auf der Befehlszeile abgesetzt werden. So kann etwa für das Beispiel aus Abbildung 7.1 die Zykluslänge des Teilzyklus `Medikation` auf die Länge=10 verlängert werden, mit der Befehlseingabe *increaseLength*(`Chemotherapie▼ Medikation`, 7). Eine Übersicht zu den erlaubten Befehlen bzw. ihren Parametern kann über das Menü *Help*→*Help* bzw. `STRG+h` aufgerufen werden. Die zweite Möglichkeit zur Anwendung von Änderungs- und Auswahloperationen bildet eine einfache grafische Be-

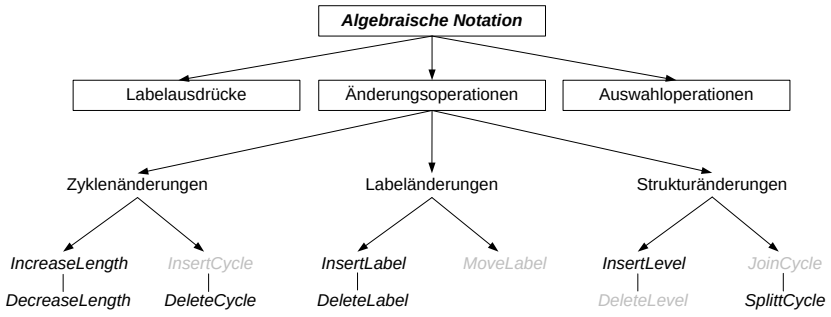


Abbildung 7.4: Implementierte Änderungsoperationen

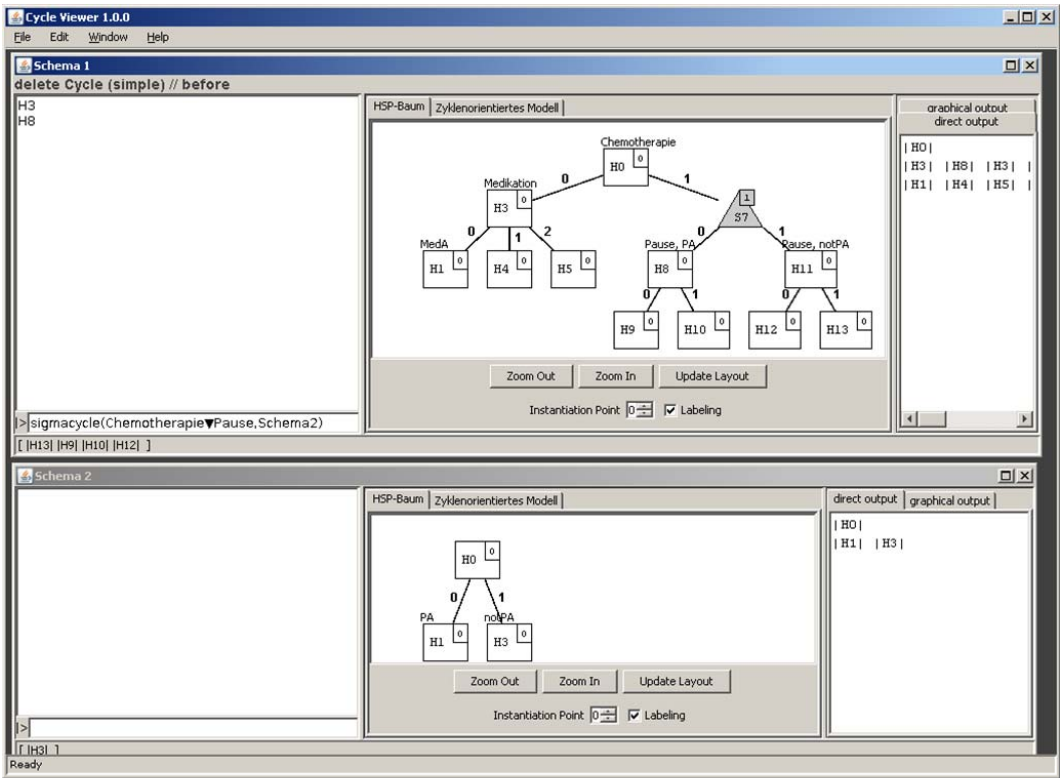


Abbildung 7.5: Beispiel für eine Auswahloperation

nutzerschnittstelle. Diese wird durch die rechte Maustaste über der HSP-Baumdarstellung aufgerufen. Kenzeichnend ist, dass der User anstelle von Labelausdrücken, Knoten im HSP-Baum direkt mit der Maus auswählen kann. Über das Kontextmenü kann dann für die adressierten Knoten bzw. die korrespondierenden (Teil-) Zyklen entsprechende Änderungs- und Auswahloperationen vorgenommen werden. Unabhängig davon, welche der beiden Möglichkeiten ein Anwender bevorzugt, in beiden werden die Vor- und Nachbedingungen von Operationen überprüft, um die Korrektheit der Spezifikation bzw. das HSP-Baums zu gewährleisten.

Jede der in Abschnitt 3.4.3 vorgestellen Auswahloperation wurde in dem Prototypen implementiert (siehe Abschnitt 3.4.3). Bei einer Auswahloperation wird zur Auswahl von Teilzyklen oder Zykluspositionen ein entsprechendes periodisches Auswahlschema benötigt. Da Auswahlschemas ebenfalls periodische Spezifikationen darstellen, werden diese in unserem Prototypen zunächst als eigenständiges (neues) Periodizitätsschema angelegt. Über den Namen des Periodizitätsschemas kann dieses dann in einer Auswahloperation referenziert werden. Ein Beispiel zeigt Abbildung 7.5. Im oberen Bereich ist das Schema nach erfolgter Zyklusauswahl ("*Jede 2.Pause*") zu sehen. Auf der linken Seite ist der entsprechende Befehl der Auswahloperation zu sehen. Wie zu erkennen ist,

wird im zweiten Parameter das Periodizitätsschema "Schema2" als Auswahlschema referenziert. Das so referenzierte "Schema2" ist zuvor als eigenständiges Periodizitätsschema separat beschrieben wurden, was im unteren Bereich von Abbildung 7.5 dargestellt ist. Bei der Projektionsauswahl spielt zusätzlich der Instanziierungspunkt der beiden Schemas eine Rolle. Dieser kann unterhalb des HSP-Baums unter dem Punkt "Instantiation Point" für beide Schemas entsprechend ausgewählt bzw. geändert werden.

7.2.2 Auswertungen und Bitletrepräsentation

Wie zuvor in Abschnitt 7.1 gezeigt, wird das Ergebnis der Auswertung von HSP-Bäumen bzw. das entsprechende Wiederholungsmuster an der rechten Seite angezeigt (siehe Abbildung 7.1). Die Auswertung findet hierbei automatisch statt, d.h. bei jeder Änderung an einem HSP-Baum wird dieser ausgewertet und die Auswertungsergebnisse pro Level ausgegeben. Zusätzlich kann der Anwender über den Reiter "Graphical Output" eine alternative Darstellung des Auswertungsergebnisses wählen, in dem weitere Zwischenergebnisse der Auswertung dargestellt werden (z.B. der Übertrag von Level zu Level).

Über den Menüpunkt *Windows→Timeline* kann die Bitletrepräsentation des Wiederholungsmusters des HSP-Baums dargestellt werden. Abbildung 7.6 zeigt dies, für das Beispiel aus Abbildung 7.1. Im oberen Bereich ist die resultierende Knotenliste bzw. Wiederholungsmuster nach Auswertung des HSP-Baums angezeigt. Im mittleren Bereich wird das Wiederholungsmuster im Zeitbereich dargestellt. Hierbei müssen im unteren Bereich die Labels ausgewählt werden (z.B. `MedA`), für deren Vorkommen das Wiederholungsmuster (hier z.B. die Gabe von Medikament A) abgeleitet werden soll. Im unteren Bereich ist das resultierende Wiederholungsmuster nach Transformation in den Bitletbereich dargestellt.

Eine zusätzliche Möglichkeit für den Umgang mit der Bitletrepräsentation bildet die (separate) Bitletkonsole. In dieser Konsole können manuell verschiedene Transformationen und Repräsentationen im Zusammenhang mit der Bitletrepräsentation durchgeführt werden. So kann sowohl die Fast-Bitlet-Transformation (Menüpunkte d-h) als auch die (aufwendigere) "klassische" Bitlet-Transformation (Menüpunkt p-q) durchgeführt werden. Über die Menüpunkte i-m können die bei der Fast-Bitlet-Transformation (theoretisch) vorgestellten Eigenschaften von Standardbitlebasen an Beispielen mit beliebiger Dimension nachvollzogen werden. Ebenso können die Standardbitletbasen bzw. deren Inverse für beliebige Dimensionen angezeigt werden (Menüpunkt a bzw. n). Abbildung 7.7 zeigt die Bitletkonsole.

7.3 Vorgehensmodell

In der prototypischen Umsetzung der algebraischen Notationsform sind nicht alle Operationen inklusive ihrer Gegenoperationen realisiert. Somit können periodische Spezifikationen nicht gänzlich "frei" beschrieben werden, da entstandene Fehler in der Beschrei-

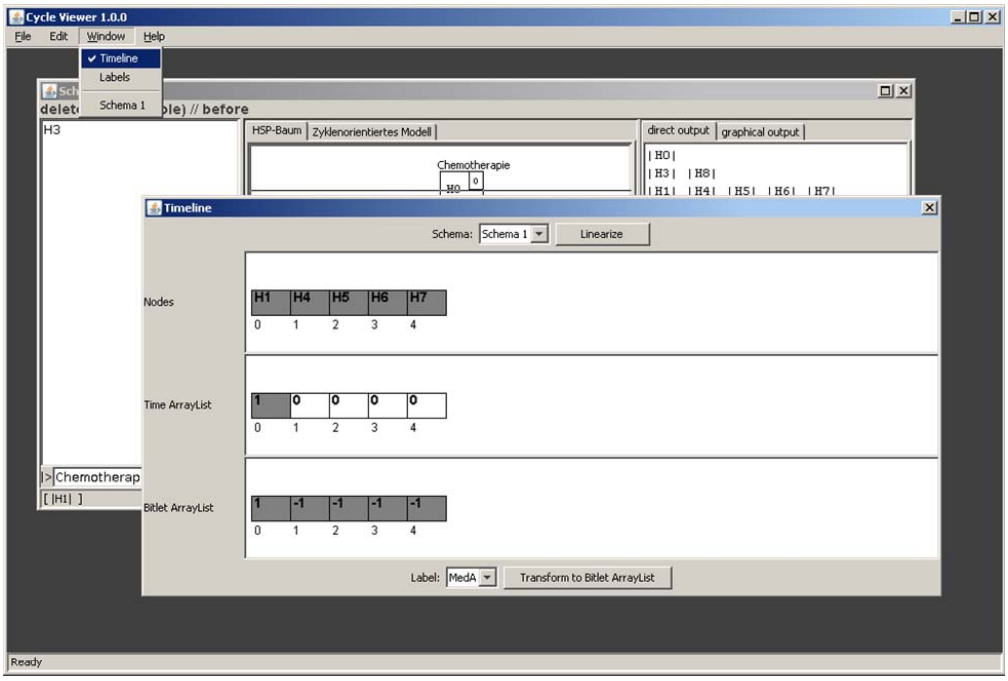


Abbildung 7.6: Bitletdarstellung im Prototyp

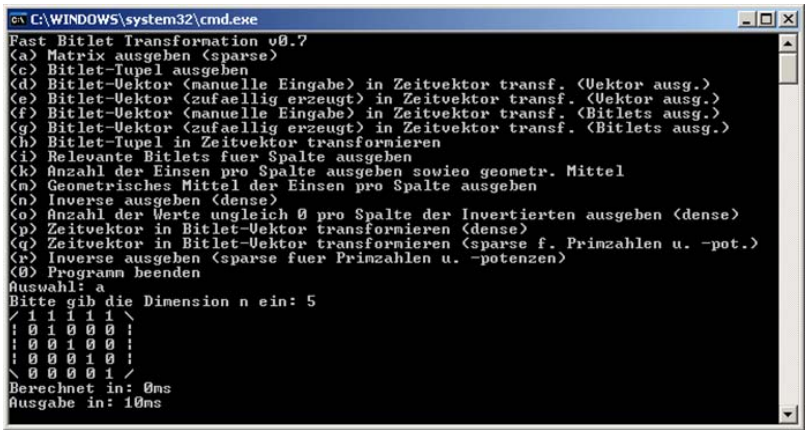


Abbildung 7.7: Darstellung der Bitletkonsole

bung nicht immer rückgängig gemacht werden können, wie es etwa bei einer vollumfänglichen Operationsmenge möglich wäre. Um diesen Aspekt zu begegnen, werden wir im Folgenden ein einfaches Vorgehensmodell zur Beschreibung periodischer Spezifikationen vorstellen. Ziel ist es, mit der strukturierten Vorgehensweise eine grundsätzliche Logik zu beschreiben, wie periodische Spezifikationen mit Hilfe der algebraischen Notationsform beschrieben werden können. Auch wenn das Vorgehensmodell die Einschränkungen des Prototypen berücksichtigt, so ist es ebenfalls auf Implementierungen mit vollständiger Operationsmenge übertragbar. Auch wenn bei vollständiger Operationsmenge stets eine Gegenoperation existiert, erleichtert das strukturierte Vorgehen des folgenden Vorgehensmodells die Beschreibung periodischer Spezifikationen:

1. Im ersten Schritt empfiehlt es sich, die bereits bekannten bzw. benötigten Labels in das Labelrepository aufzunehmen und der entsprechenden Labelmenge zuzuweisen. Die Labelmenge *StructureSet* fasst die Labels zusammen, die zur Strukturierung von Periodizitätsschemas verwendet werden können. Die Labelmenge *ActivitySet* enthält alle Labels, die mit konkreten Aktivitäten (z.B. Medikamentengaben) in der realen Welt korrespondieren.
2. In diesem Schritt finden die Strukturierungen der (Teil-) Zyklen und die Einführung der verschiedenen Levels statt. Pro Level gilt es zu überlegen, ob es sich um einen einfachen Zyklus handelt oder um einen (horizontal und/oder vertikal) strukturierten Zyklus, bestehend aus mehreren Teilzyklen oder Levels. Im ersten Fall kann der erste Schritt entfallen.
   a) *Anlegen eines neuen Levels*: Unabhängig ob ein Zyklus horizontal oder vertikal strukturiert werden soll, muss zunächst stets ein neuer Level unterhalb des aktuellen Levels eingefügt werden (*InsertLevel*). Dieser neue Level wird dann in den nächsten Schritten wie gewünscht strukturiert.
   b) *Anpassung der Zykluslängen*: Mit der Funktion *increaseLength* kann die gewünschte Länge des Zyklus angegeben werden. Bei horizontal strukturierten Zyklen empfiehlt es sich zunächst die Gesamtlänge, d.h. die Summe aller Teilzykluslängen anzugeben und dann den folgenden Zwischenschritt vorzunehmen.
      i. *Strukturierung des Level*s: Mit der Funktion *splittCycle* kann der zuvor angelegte "große" Zyklus in verschiedene kleinere Teilzyklen aufgeteilt werden.
3. Mit den Funktionen *insertLabel* und *deleteLabel* können die im 1.Schritt angelegten Labels je nach Semantik den gewünschten Zyklen oder Zykluspositionen zugeordnet werden. Im initialen Fall, d.h. keine der (neuen) Strukturen weist Labelzuordnungen auf, werden in den Labelausdrücken ausschließlich Positionsangaben verwendet. Beispielsweise kann der erste Teilzyklus eines strukturierten Zyklus `Chemotherapie` über die Positionsangabe 1 im Labelausdruck $\texttt{Chemotherapie}^{1}\blacktriangledown*$ adressiert werden.

4. Je nach Anwendungsfall wird eine Zyklus-, Zykluspositions- oder Projektionsauswahl benötigt. Hierbei gilt es für jede dieser drei Operationen, die folgenden zwei Schritte vorzunehmen.

   a) *Anlegen des Auswahlschemas*: Das Auswahlschema wird zunächst als "normale" periodische Spezifikation angelegt. Hierbei empfiehlt sich das gleiche Vorgehensmodell wie in dem vorliegenden Abschnitt vorgestellt.

   b) *Durchführen der Auswahl*: Auf Basis des Auswahlschemas und der entsprechenden Auswahloperation können bestimmte Zyklen oder Zykluspositionen periodisch ausgewählt werden. Allerdings hängt es vom konkreten Anwendungsfall ab, zu welchem Zeitpunkt eine Auswahloperation stattfinden muss. Die Auswahl kann entweder bereits unmittelbar für einen gerade angelegten (Teil-)Zyklus oder Level erfolgen oder erst nachdem die Gesamtstruktur der periodischen Spezifikation definiert wurde. Insbesondere bei Projektionsauswahl gilt es dies zu beachten. Wie in Abschnitt 3.4.3.3 gezeigt, werden die Eigenschaften einer periodischen Spezifikation zum Zeitpunkt der Projektion "eingefroren". Das heißt, sollte sich die periodische Spezifikation (für die eine Projektionsauswahl vorgenommen wurde) ändern, so hat diese Änderung keinen Einfluss auf die zuvor vorgenommene Projektionsauswahl (Nichtauswahlerhaltende Operationsreihenfolgen, vgl. Abschnitt 3.4.3.3). Lediglich wenn eine Projektionsauswahl als letzte Änderungsoperation bei der Beschreibung einer periodischen Spezifikation angewendet wird, können alle Besonderheiten und Eigenschaften des Periodizitätsschemas bei der Projektion berücksichtigt werden.

Auf Basis der Auswahl aus dem 5.Schritt können nun nachfolgend erneut Änderungsoperationen vorgenommen werden. Neben Veränderungen an der Struktur der ausgewählten Zyklen (vgl. 2.Schritt), dem Einfügen und Löschen von Labels (vgl. 3.Schritt) können auch (Teil-) Zyklen (periodisch) gelöscht werden (*deleteCycle*). Das bedeutet jedoch, dass die Schritte 2-4 sich wiederholen bzw. das Vorgehensmodell erneut durchlaufen wird.

Zusammengefasst betrachtet, stellt das Vorgehensmodell lediglich einen Leitfaden dar, in welcher Reihenfolge die Operationen der algebraischen Notationsform durchgeführt werden können. Es hilft hierbei konzeptionelle/logische Fehler bei der Beschreibung komplexer periodischer Spezifikationen zu vermeiden. Es schütz jedoch nicht davor, dass bei einer falschen Einzeloperation Fehler entstehen können, die nicht wieder rückgängig gemacht werden können bzw. nur durch eine "Neustart" der Beschreibung behoben werden können.

7.4 Anwendungsbeispiel

Im Folgenden werden wir das Vorgehensmodell und den Prototypen anhand eines konkreten Anwendungsbeispiels erläutern bzw. vorführen. Als Beispiel dient das folgende Beispiel einer Chemotherapie:

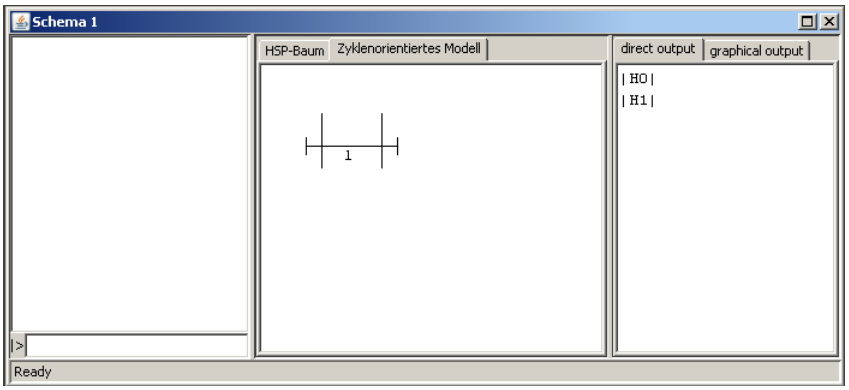
"*Eine Chemotherapie besteht aus der Gabe von Medikament A und Medikament B. Medikament A wird in einem Zyklus der Länge 28 vom 1.-14.Tag gegeben und Medikament B am 1. und 8.Tag. Danach findet eine Pause von 5 Tagen statt, in der am letzten Tag ein Bluttest stattfindet. Nach jeder zweiten Pause (beginnend bei der Letzten) werden zusätzliche Laborwerte ermittelt.*"

Hierbei soll die folgende organisatorische Rahmenbedingung berücksichtigt werden, die sich aufgrund der hochgradig an den Patienten zugeschnittenen Medikamente ergeben:

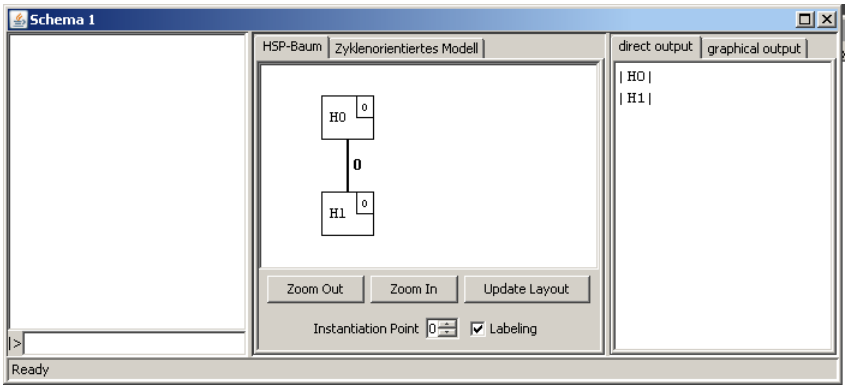
"Eine Packung von Medikament A reicht lediglich für 20 Einzelgaben und von Medikament B für 5 Einzelgaben. Neue Packungen müssen bestellt werden sobald pro Medikament nur noch zwei Einzeldosen vorhanden sind."

Wir beginnen mit der Beschreibung der periodischen Spezifikation der Chemotherapie.

- Im 1.Schritt werden alle Labels im Labelrepository angelegt.
  - In der Labelmenge *ActivitySet* wird das Medikament A wird durch das Label `MedA` und von Medikament B durch das Label `MedB` repräsentiert. Der Bluttest bzw. die Laborwerte entsprechen den Labels `BT` und `LW`
  - In der Labelmenge *StructureSet* wird die gesamte Chemotherapie durch das Label `Chemotherapie`, die Medikationszeit (Tag 1-14) durch das Label `Medikation`, die medikationsfreie Zeit (Tag 15-28) durch das Label `Medikationsfrei` und Pausen durch das Label `Pause` repräsentiert.
- Im 2.Schritt wird die Struktur eines Chemotherapiezyklus festgelegt. Wir entscheiden uns für einen strukturierten Zyklus bestehend aus den drei Teilzyklen `Medikation`, `Medikationsfrei` und `Pause`. Ausgangspunkt bildet ein initiales Periodizitätsschema mit einem Zyklus der Länge 1 bzw. dem minimalen HSP-Baum bestehend aus einem Wurzelknoten und einem Blattknoten. In Abbildung 7.8a ist der initiale Fall im zyklenorientierten Modell und in Abbildung 7.8b der korrespondierende (minimale) HSP-Baum dargestellt.
  - Zunächst wird mit der Operation $insertLevel(*)$ ein neuer Level unterhalb des Ausgangslevel, der durch Wildcardzeichen $*$ adressiert wird, angelegt.
  - Die Zykluslänge des Gesamtzyklus beträgt 33 Tage (28 Tage Medikation + 5 Tage Pause). Mit der Operation $increaseLength(*\blacktriangledown *, 32)$ wird die Länge im neuen Level angepasst.
  - Die horizontale Strukturierung des Gesamtzyklus in drei Teilzyklen erfolgt mit den Operationen $splittCycle(*\overset{1}{\blacktriangledown} *, 14)$ und $splittCycle(*\overset{2}{\blacktriangledown} *, 14)$.

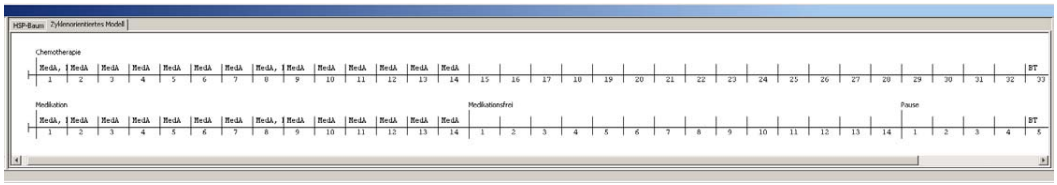


(a) Darstellung im zyklenorientierten Modell

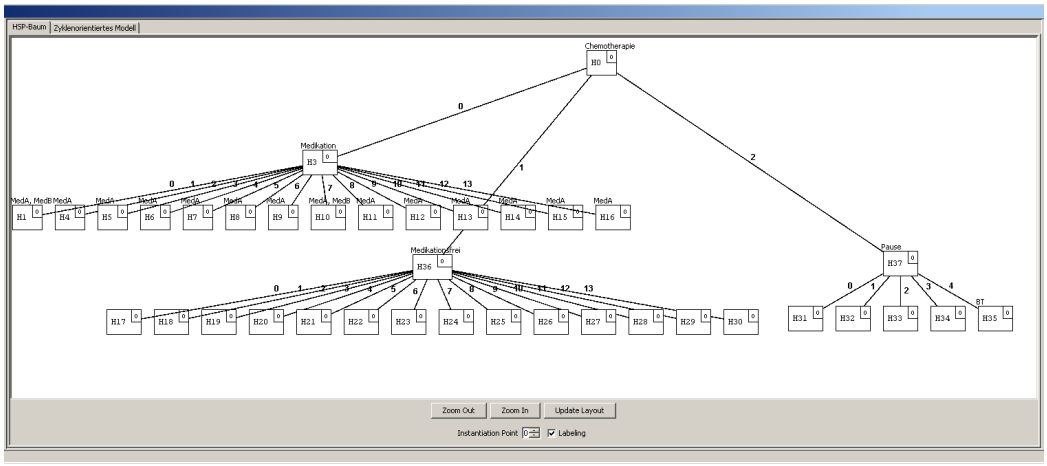


(b) HSP-Baumdarstellung

Abbildung 7.8: Initiale Periodizitätsschema



(a) Darstellung im zyklenorientierten Modell



(b) HSP-Baumdarstellung

Abbildung 7.9: Periodizitätsschema mit eingefügten Labels

- Im 3.Schritt werden die Labels den Teilzyklen und Zykluspositionen zugewiesen. Das Ergebnis zeigt Abbildung 7.9

- Im 4.Schritt wird die Ausnahme bzgl. jeder 2.Pausen und zusätzlichen Laborwerten mittels einer Zyklusauswahl begegnet.

  – Die Aussage "*Jede 2.Pause*" entspricht einem Auswahlschema bestehend aus einem Zyklus der Länge = 2, dass in Abbildung 7.10 dem Schema 2 entspricht.

  – Die Zyklusauswahl erfolgt auf dem Teilzyklus `Pause` mit dem Operationsaufruf $\sigma_{Cycle}(\texttt{Chemotherapie▼Pause}, Schema\,2)$.

  – Basierend auf der obigen Auswahl wird das Label `LW` eingefügt, dass die zusätzlichen Laborwerte (`LW`) zu jeder 2.Pause repräsentiert.

Nachdem wir das grundsätzliche Periodizitätsschema beschrieben haben, werden wir im Folgenden die zusätzliche Rahmenbedingung berücksichtigen. Der Lösungsansatz besteht darin, dass die einzelnen Mengenangaben einer Packung als eine periodische Angabe verstanden werden kann, d.h. dass jede 20.Gabe von Medikament A und jede 5.Gabe von Medikament B die jeweiligen Packungen aufgebraucht sind. Entsprechend

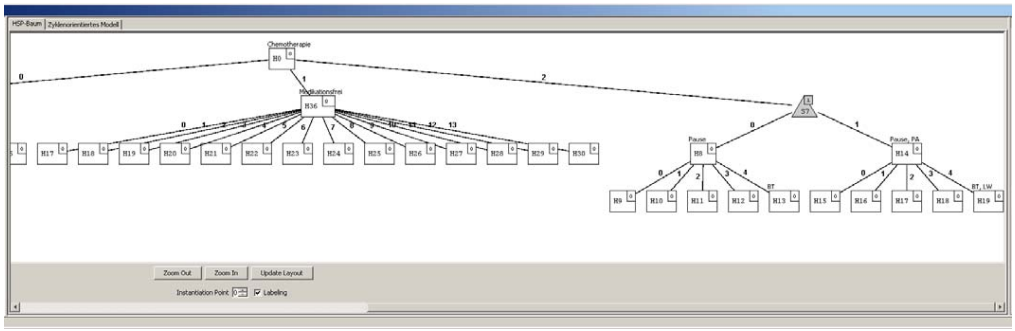


Abbildung 7.10: Periodizitätsschema (Ausschnitt) mit Zyklusauswahl

können wir eine Zyklusauswahl auf den Positionen im obigen Periodizitätsschema durchführen, an denen etwa Medikament A gegeben wird. Das heißt, mit der Zyklusauswahl können wir jede 20.Gabe von Medikament A auswählen (unabhängig vom konkreten Zyklus, siehe Abschnitt 3.4.3.2) und somit den Bestellzeitpunkt für neue Packungen. Allerdings soll bereits bestellt werden, wenn nur noch jeweils 2 Einzeldosen vorhanden sind. Daraus folgt, dass unser Auswahlschema zwar die Zykluslänge = 20 auf (20 Einzelgaben) benötigt, der Bestellzeitpunkt entspricht jedoch der 18.Position (18.Gaben von Medikament A) im Auswahlschema. Diese Position wird mit dem Label `BZA` (Bestellzeitpunkt Medikament A) repräsentiert. Auf die gleiche Weise können wir mit Medikament B verfahren. Hier weist das Auswahlschema eine Zykluslänge = 5 auf und das Label `BZB` findet sich an der 3.Position im Auswahlschema. Für die Zyklusauswahl werden die beiden Auswahlschemas $Schema2$ und $Schema3$ definiert und die Auswahl mit den folgenden Operationsaufrufen durchgeführt:

- $\sigma_{Position}(\texttt{Chemotherapie}\blacktriangledown\texttt{Medikation}\triangledown\texttt{MedA}, Schema\,2)$
- $\sigma_{Position}(\texttt{Chemotherapie}\blacktriangledown\texttt{Medikation}\triangledown\texttt{MedB}, Schema\,3)$

Nach erfolgter Auswahl kann mit dem Labelausdruck `Chemotherapie▼Medikation▽BZA` bzw. `Chemotherapie▼Medikation▽BZB` auf die Positionen zugegriffen werden, an denen neue Bestellung für Medikamente erfolgen müssen. Ebenso kann ein entsprechendes Wiederholungsmuster für die Bestellzeitpunkte für beide Medikamente abgeleitet werden. Aus darstellungstechnischen Gründen ist in Abbildung 7.11 lediglich das (schematische) Ergebnis der Zyklusauswahl für Medikament B und seine Bestellzeitpunkten dargestellt. Abbildung 7.12 zeigt ein Ausschnitt der Wiederholungsmuster für den Bestellzeitpunkt von Medikament B.

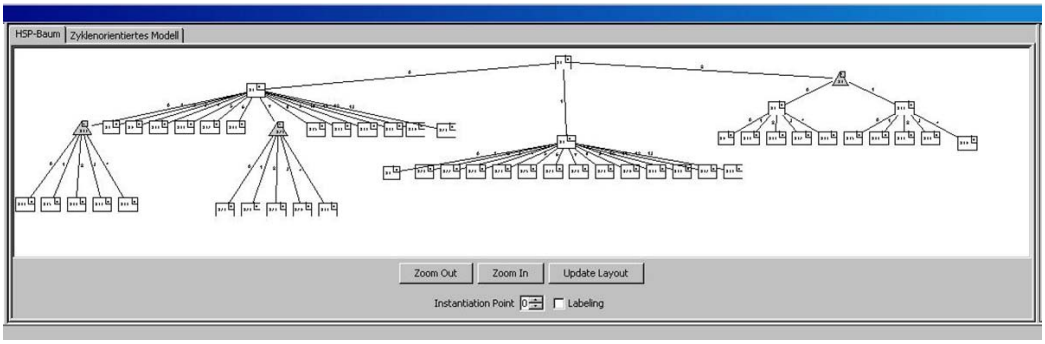


Abbildung 7.11: Berücksichtigung der Bestellzeitpunkte

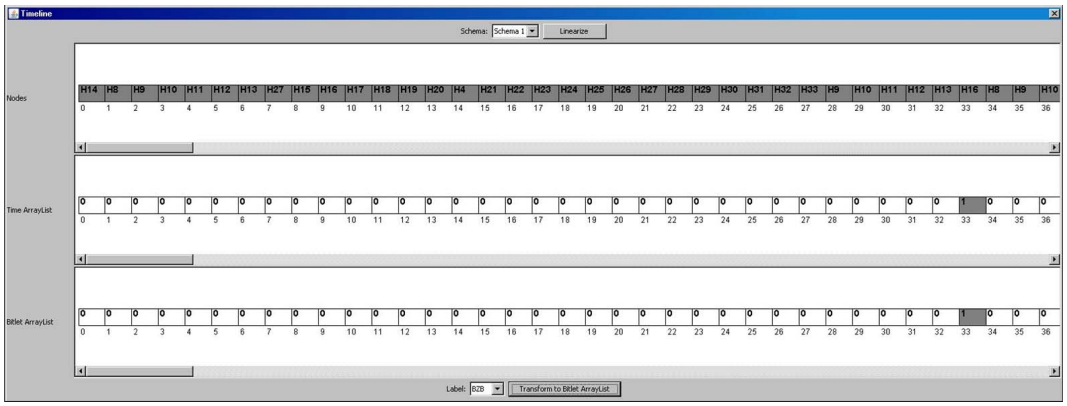


Abbildung 7.12: Wiederholungsmuster (Ausschnitt) für die Bestellzeitpunkte von Medikament B

Zusammenfassend betrachtet, zeigt der Prototyp eindrucksvoll, dass die im Rahmen dieser Arbeit vorgestellten Einzelkonzepte zum einen effizient und robust implementierbar sind und zum anderen diese untereinander auch ein effektives Zusammenspiel erlauben. Die im Prototyp umgesetzten ausgewählten Konzepte erlauben es bereits komplexe periodische Spezifikationen bzw. Zusammenhänge zu beschreiben und auszuwerten. Auch wenn nicht alle Einzelkonzepte in den Prototypen eingeflossen sind, so ermöglicht das vorgestellte Vorgehensmodell einige der dadurch auftretenden Schwächen zu kompensieren. Das abschließende Anwendungsbeispiel hat nicht nur die Handhabung des Prototypen demonstriert, sondern auch gezeigt, dass periodische Spezifikationen bzw. Zusammenhänge auch an unerwarteter Stelle auftreten können bzw. als solche verstanden werden können.

8 Zusammenfassung und Ausblick

In vielen Anwendungsgebieten kommt zunehmend der Wunsch auf, temporale Aspekte systemseitig zu unterstützen. Aus diesen Gründen werden prozessorientierte Informationssysteme zunehmend mit Anforderungen konfrontiert, die so bislang nur in Projektmanagement-Werkzeugen bzw. speziellen Planungssystemen realisiert worden sind (z.B. Scheduling von Aktivitäten, Spezifikation und Überwachung von zeitlichen Minimal- bzw. Maximalabständen zwischen Vorgängen). Die Berücksichtigung temporaler Aspekte erfordert, neben der Erweiterung von Prozess-Metamodelle um zeitliche Konstrukte, auch geeignete Methoden für die Spezifikation, Verifikation, Koordination und Überwachung zeitlicher Beschränkungen in Prozessen. Insbesondere zählen hierzu auch die in der Praxis häufig anzutreffenden Prozesse, in denen elementare Aktivitäten oder Abfolgen von Aktivitäten in bestimmten zeitlichen Abständen periodisch wiederholt werden müssen.

Das Fehlen einer adäquaten Unterstützung periodischer Zusammenhänge (Periodizität) ist für viele Anwendungen bisher nicht bzw. nicht in ausreichendem Maße unterstützt. Diese für viele Anwendungen nicht zufriedenstellende Situation bildete den Ausgangspunkt unserer Betrachtungen. Das Ziel dieser Arbeit war die Entwicklung eines ganzheitlichen Ansatzes zur integrierten Unterstützung von Periodizität. Das zentrale Anliegen war hierbei, Konzepte und Modelle zu entwickeln, die als Basistechnologien zur Unterstützung periodischer Zusammenhänge in unternehmensweiten (prozessorientierten) Informationssystemen (auch bei tausenden von Benutzern und aktiven Prozess-Instanzen), wesentlich sein werden. Vor diesem Hintergrund ergaben sich verschiedene Forschungsfragestellungen (siehe Abschnitt 1.3), die wir im Folgenden beantworten werden.

Welche logischen Zeitmodelle bzw. Formalismen ermöglichen eine ausdrucksstarke Spezifikation äquidistanter und nichtäquidistanter Periodizität, so dass auch Ausnahmen und Granularitäten berücksichtigt werden können?

Es zeigte sich, dass die generische Unterstützung von Ausnahmen in Periodizitätsschemas bzw. periodischen Spezifikationen eine der wichtigsten Anforderungen darstellt. Eine entsprechende Unterstützung ist die Grundvoraussetzung für die Verwendung unterschiedlicher Granularitäten und der Handhabung damit verbundener Kalenderphänomene (z.B. Sommer-/Winterzeit bei Fahrplänen). Allerdings stellt die adäquate Unterstützung von Ausnahmen ein großes Manko existierender Beschreibungsformalismen dar. In keinem der in der Literatur vorliegenden Ansätze finden sich Möglichkeiten, einheitlich periodische oder nicht-periodische Ausnahmen zu beschreiben (siehe Abschnitt 2.4). Mit dem in dieser Arbeit eingeführten zyklenorientierten Modell, haben wir ein neues logisches Zeitmodell und Beschreibungsformalismus eingeführt, das den Anforderungen an

eine umfassende und generische Unterstützung von Ausnahmen in Periodizitätsschemas begegnen kann. Auf Basis der Trennung von Struktur (vorgegeben durch die Menge an Zyklen, Teilzyklen und Zyklenpositionen) und Semantik (definiert über die Labels) im Zeitmodell, können neben einfachen Ausnahmen auch komplexe nichtäquidistante Ausnahmen beliebiger Komplexität auf generische Weise beschrieben werden. Insbesondere die Unterstützung von "Projektionsausnahmen" ist an dieser Stelle besonders erwähnenswert. Solche Ausnahmen ermöglichen es dem Anwender, zwei zueinander eigentlich unabhängige, periodische Spezifikationen in Beziehung zu setzen und Abhängigkeiten zwischen ihnen auszudrücken. Somit können bei der Konstruktion eines Periodizitätsschemas a priori bekannte Abhängigkeiten mit anderen Periodizitätsschemas berücksichtigt werden. Bereits zum Spezifikationszeitpunkt kann ersten Aspekten begegnet werden, die sich bezüglich Vergleichbarkeit periodischer Spezifikationen und Bewertung ihres Zusammenwirkens untereinander ergeben. Eine solche A-priori-Berücksichtigung von Abhängigkeiten ist in vielen Anwendungen nicht nur wünschenswert, sondern auch wesentlich für den praktischen Einsatz.

Wie kann der Umgang mit periodischen Spezifikationen im Zusammenhang mit unterschiedlichen zeitlichen Einheiten (Granularitäten) adäquat unterstützt werden?

Das zyklenorientierte Modell bzw. der HSP-Baum (als dessen interne endliche Repräsentation) bilden ein solides Fundament, zur Berücksichtigung unterschiedlicher zeitlicher Granularitäten in periodischen Spezifikationen. Wir haben die Grundideen vorgestellt, wie die in der vorliegenden Arbeit eingeführten Modelle und Theorien zur Realisierung solch semantisch "höherwertiger" Modell bzw. Strukturen verwendet werden können (siehe Abschnitt 4.5). Zusätzliche weiterführende Forschungsarbeiten sind jedoch notwendig, um hier den mit der Forderung nach Granularitätsfähigkeit bzw. mit der Handhabung unterschiedlicher Granularität verbundenen zusätzlichen Anforderungen vollumfänglich begegnen zu können.

Welche Ansätze erlauben es, periodische Spezifikationen auf semantisch hoher Ebene beschreiben zu können, damit diese sowohl für "Nicht-Experten" intuitiv verständlich als auch für "Fachanwender" leicht anpassbar sind?

Ein Benutzer sollte periodische Spezifikationen bzw. Periodizitätsschemas auf semantisch hoher Ebene beschreiben können. Es gilt, den Anwender so gut wie möglich bei der Spezifikation periodischer Zusammenhänge zu unterstützen. Hierzu zählt unter anderem auch, dass semantisch falsche, inkonsistente oder widersprüchliche Angaben erkannt bzw. vermieden werden. Zur Beschreibungen periodischer Spezifikationen auf semantisch hoher Ebene ist ein zugrundliegendes logisches (Meta-) Modell unumgänglich, dass die Syntax und Semantik entsprechender Spezifikationen präzise festlegt. Ein solches Modell ist unerlässlich, wenn die Handhabung und Implementierung periodischer Spezifikationen effektiv und effizient sein soll. Mit dem zyklenorientierten Zeitmodell und HSP-Baum, als dessen interne (physische) Repräsentationsform, haben wir ein solches zugrundliegendes logisches (Meta-) Modell eingeführt. Deren präzise Syntax und Semantik bildete auch die Grundlage dafür, dass Fehler bzw. semantisch falsche, in-

konsistente oder widersprüchliche Angaben erkannt und korrigiert werden können. Im Gegensatz zu bestehenden Ansätzen, wird auf eine Übermenge an Einzeloperationen, mit z.T. diffizilen und für Anwender kaum nachvollziehbaren Unterschieden verzichtet. Stattdessen wurde eine Grundmenge an generischen Operationen vorgestellt, die in ihrer Semantik klar voneinander abgegrenzt sind. Dies fördert nicht nur die Verständlichkeit sondern reduziert in gleichem Maße auch die Fehleranfälligkeit entsprechend beschriebener Periodizitätsschemas. Aufbauend auf dem eingeführten Zeitmodell und dem HSP-Baum können in entsprechenden Informationssystemen oder PAIS semantisch höherwertige Unterstützung (z.B. grafische Benutzerschnittstellen, Integration in Prozessmetamodell) realisiert werden. Diese Integration eröffnet den Weg hin zu einer intuitiven, verständlichen und nachvollziehbaren Beschreibung von periodischen Zusammenhängen in Informationssystemen.

Welche logische Repräsentation kann gleichzeitig eine effiziente Vergleichbarkeit periodischer Spezifikationen und eine kompakte (interne) Repräsentation gewährleisten?

Eine adäquate operationale Unterstützung ist unerlässlich für einen effektiven Umgang mit periodischen Spezifikationen. Hierbei zeigt sich, dass neben den Ableitungs- und Überprüfungsoperationen, die Vergleichsoperationen eine der größten Herausforderungen darstellen. Wie haben gezeigt, dass lediglich eine auf Wiederholungsmustern basierende Realisierung den Anforderungen nach operationaler Unterstützung begegnen kann (siehe Abschnitt 2.4). Vor diesem Hintergrund erwuchs die Anforderung nach Ableitbarkeit von Wiederholungsmustern für das zyklenorientierte Zeitmodell bzw. den HSP-Baum, als dessen kompakte und berechenbare Repräsentation. Es zeigte sich, dass Auswahl- und Projektionsebenen die eigentliche Herausforderung bei der Auswertung von HSP-Bäumen darstellen. Mit der eingeführten periodischen Projektion haben wir einen Lösungsansatz vorgestellt, der eine effiziente Auswertung entsprechender Projektionsebenen ermöglicht. Diese Theorie erlaubt eine direkte "Berechnung" der Wiederholungsmuster solcher Ebenen auch bei unterschiedlichen Instanziierungszeitpunkten. Darüber hinaus zeigte sich, dass die entwickelte Theorie auch eine hervorragende Grundlage für eine effiziente Auswertung von Auswahlebenen darstellt. Sie ermöglicht hierdurch einen einheitlichen Umgang mit Auswahl- und Projektionsebenen bei der Auswertung.

Wie können periodische Spezifikationen persistent und effizient gespeichert werden, so dass Vergleiche direkt auf dieser (internen) Repräsentationsform ausführbar sind?

Für den praktischen Einsatz ist eine effiziente operationale Unterstützung unumgänglich. Wie wir zeigen konnten, fehlt es an adäquaten Ansätzen einer musterbasierten Realisierung von Vergleichsoperationen für Wiederholungsmuster mit sehr langen Periodenlängen. Aufgrund des Repräsentationsumfangs und dem damit verbundenen operationalen Aufwand sind vorliegende Vorschläge für den praktischen Einsatz weniger geeignet. Kompaktere Repräsentationen – unter Beibehaltung der Vergleichbarkeit – sind sowohl bezüglich des Speicherbedarfs des Einzelmusters als auch bezüglich der Effizienz der operationalen Unterstützung für den praktischen Einsatz essentiell. Mit der Bitletrepräsentation ist ein neuer Lösungsansatz vorgestellt worden, der eine kompakte

Repräsentation und effiziente operationale Realisierung von Vergleichsoperation zwischen Wiederholungsmustern – auch bei langen Periodenlängen und großer Anzahl von diesen – realisiert. Die Bitletrepräsentation ermöglicht es, dem Hauptproblem heutiger Verfahren zu begegnen, das bei der notwendige Vereinheitlichung unterschiedlich langer Wiederholungsmuster entsteht. In heutigen Verfahren führt die Vereinheitlichung zu einem starken Anstieg sowohl des operationalen Aufwands als auch des Repräsentationsumfangs. Im Gegensatz dazu entfällt bei der Bitletrepräsentation der operationale Aufwand (d.h. unterschiedlich lange Muster müssen nicht explizit vereinheitlicht werden) und der Umfang der Repräsentation bleibt konstant. Des Weiteren konnten wir zeigen, dass eine Transformation von Wiederholungsmustern in entsprechende Bitletrepräsentationen und anders herum sehr effizient möglich ist. Die vorgestellten Verfahren weisen eine Komplexität auf, die einen praktischen Einsatz erlauben. Darüber hinaus kann die Bitletrepräsentation als eine Form der (verlustfreien) Komprimierung von Wiederholungsmustern verstanden werden. Wie gezeigt, kann bei geeigneter Wahl der Bitlets, die Repräsentation eines Wiederholungsmusters in Form einer Bitletrepräsentation äußerst kompakt sein.

Mir der Effizienz und Kompaktheit der Bitletrepräsentation ergibt sich allerdings auch die Problematik der Realisierbarkeit von Vergleichsoperationen. Zur Realisierung entsprechender Operationen im Bitletbereich kann nicht das gleiche Vorgehen wie bei Wiederholungsmustern im Zeitbereich verwendet werden. Allerdings konnten wir für jede relevante Vergleichsoperation auch eine Realisierung im Bitletbereich aufzeigen. Die vorgestellten Verfahren stellen jedoch nicht nur die grundsätzliche Realisierbarkeit der Operationen sicher, sondern es handelt sich hier um effiziente Verfahren, die sich gegenüber Realisierungen im Zeitbereich teilweise als deutlich effizienter erweisen. Zusammenfassend kann festgehalten werden, dass die Bitletrepräsentation einen ersten fundamentalen Schritt in Richtung einer effizienten operationalen Unterstützung von Wiederholungsmustern darstellt. Auch wenn es uns in der vorliegenden Arbeit nicht vollständig gelungen ist, alle spezifischen und allgemeinen Probleme der Bitletrepräsentation bzw. Transformation zu begegnen, so stellt diese dennoch einen lohnenswerten weiterzuverfolgenden Weg dar. Mit den in dieser Arbeit geschaffenen Grundlagen ist der erste Schritt dazu getan. Es bedarf jedoch Folgeansätze, um das Potential dieses Lösungsansatzes in Gänze auszuschöpfen.

Welche Zugriffs- und Indexstrukturen können verwendet werden, um die Effizienz und Anwendbarkeit der entwickelten Lösungen auch bei großer Anzahl an Daten sicherzustellen?

Für die Praxis ist es unumgänglich Betrachtungen vorzunehmen, wie periodische Zusammenhänge auch im Kontext einer großen Menge an Daten effizient verwaltet und ausgewertet werden können. Hierzu zählen auch Betrachtungen zur Entwicklung geeigneter Zugriffs- und Indexstrukturen für entsprechend repräsentierte periodische Spezifikationen. Wie wir gezeigt haben, sind entsprechende Strukturen nur im Zusammenhang mit eingeschränkten Vergleichsanfragen, d.h. Anfragen die lediglich einen bestimmten Zeitintervall betrachten, für den praktischen Einsatz sinnvoll (siehe Abschnitt 6.6). Die

Grundidee zur indizierten Auswertung von Vergleichsanfragen basiert auf der Eigenschaft, dass in einem bestimmten Zeitintervall nur einige wenige Bitlets einer Bitletrepräsentation relevant sind (siehe Abschnitt 6.5.2). Wir haben gezeigt, dass "klassische" (zweidimensionale) Index- bzw- Zugriffsstrukturen mit der Bitletrepräsentation kombiniert werden können, um die Spezifikationen zu ermitteln, die für einen gegebenen Zeitintervall (Ereignishorizont) relevante Bitlets enthalten. Alle anderen Spezifikationen sind nicht für den Ereignishorizont relevant und müssen etwa bei Vergleichen nicht berücksichtigt werden. Es zeigte sich jedoch, dass die Selektivität solcher Index- bzw. Zugriffsstrukturen, d.h. die Anzahl der "False-Hits" pro Anfrage, bei der Standardbitletbasis gegenüber alternativen Ansätzen nur geringfügig besser ist. Allerdings konnten wir diese Problematik auf ein allgemeines, noch offenes Problem der Bitletrepräsentation zurückführen. Die Lösung des allgemeinen Problems der Bitletrepräsentation (Lokalisationsproblem) bildet somit den Schlüssel für deutlich effektivere und effizientere Index- und Zugriffsstrukturen für periodische Spezifikationen in der Zukunft.

Insgesamt erlauben die von uns entwickelten Lösungen und Algorithmen eine durchgängige Unterstützung periodischer Zusammenhänge in Informationssystemen. Neben der ausdruckstarken Spezifikation entsprechender Zusammenhänge kann auch eine operationale Unterstützung gewährleistet werden. Insbesondere Vergleichsoperationen, welche das Zusammenwirken solcher Zusammenhänge bewerten, werden von uns nicht nur grundsätzlich unterstützt, sondern ihre Skalierbarkeit bzw. Effizienz kann auch bei großem Datenvolumen gewährleistet werden. Abbildung 8.1 zeigt abschließend das Zusammenspiel unserer Konzepte vor dem Hintergrund eines prozessorientierten Informationssystems (PAIS). Mit dem zyklen-orientierten Zeitmodell und seiner algebraischen Notationsform können bei der Modellierung von Prozessen (Build-Time) komplexe periodische Aktivitäten in Prozessen beschrieben werden. Die (interne) Repräsentation der beschriebenen periodischen Aktivitäten als HSP-Baum ermöglicht es, bei der Instanziierung eines Prozess bzw. für eine Prozessinstanz (Run-Time) das korrespondierende Wiederholungsmuster der (periodischen) Aktivität durch Auswertung des HSP-Baums abzuleiten. Gleichzeitig kann auch die entsprechende Bitletrepräsentation zur Verfügung gestellt werden. Die Bitletrepräsentation bzw. die zugehörigen Vergleichsoperationen können in entsprechenden Methoden in PAIS zur Verifikation, Koordination und Überwachung zeitlicher Beschränkungen in Prozessen verwendet bzw. integriert werden, so dass auch vorausschauende Überprüfungen vorgenommen bzw. Konflikte erkannt werden können. Aufgrund der Skalierbarkeit und Effizienz der Vergleichsoperationen auf Basis der Bitletrepräsentation ist dies auch bei großer Anzahl an Prozessen bzw. laufenden Prozessinstanzen möglich.

Auch wenn mit dieser Arbeit lediglich die ersten Schritte getan wurden, so zeigt sich dennoch, dass die vorgestellten Konzepte völlig neue Möglichkeiten in Bezug auf die Realisierung von Informationssystemen im Allgemeinen und prozessorientierten Informationssystemen (PAIS) im Besonderen eröffnet. Weitere Schritte müssen jedoch folgen, um das Potential dieser Ansätze voll zu entfalten, damit aktuellen und zukünftigen Anforderungen an Informationssystemen effektiv und effizient begegnet werden kann.

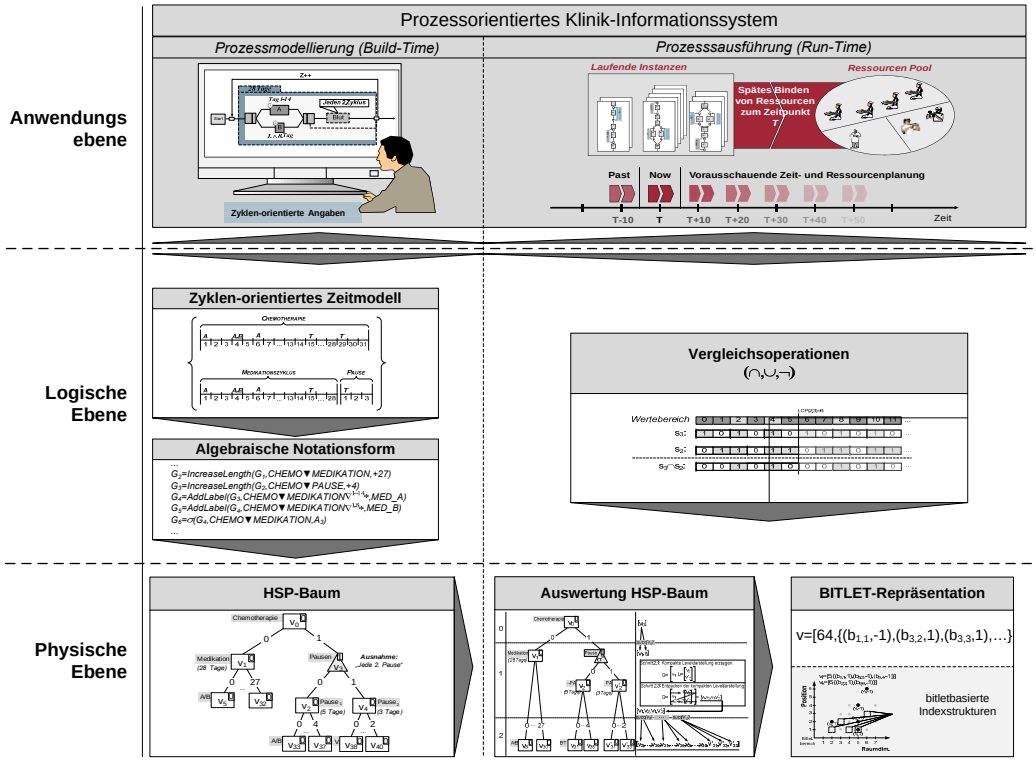


Abbildung 8.1: Zusammenspiel der entwickelten Konzepte am Beispiel eines prozessorientierten Informationssystems

Ausblick

Wie so oft, wirft die Auseinandersetzung mit einer komplexen Thematik, neue Fragestellungen auf, die Gegenstand von Folgearbeiten sein können. Aufbauend auf den Ergebnissen der vorliegenden Arbeit, bieten sich folgende Themen für weitergehende Untersuchungen an:

Granularitätsfähigkeit: Für die Realisierung sowohl von Granularitätsmodellen als auch granularitätsabhängiger, periodischer Spezifikationen bilden das zyklenorientierte Modell bzw. der HSP-Baum ein solides Fundament. Wir haben die Grundideen vorgestellt, wie die in der vorliegenden Arbeit eingeführten Modelle und Theorien zur Realisierung solch semantisch "höherwertigeren" Modelle bzw. Strukturen verwendet werden können. Dies gilt sowohl für die grundsätzliche Abbildbarkeit solcher Strukturen als auch für deren Auswertungsfähigkeit. Um jedoch der Anforderung nach Granularitätsfähigkeit bzw. die mit der Handhabung unterschiedlicher Granularität verbundenen zusätzlichen Anforderungen begegnen zu können, bedarf es weiterführender Forschungsarbeiten.

Weiterentwicklung der Bitletrepräsentation: Wie erwähnt, konnte in dieser Arbeit nicht allen Problemen der Bitletrepräsentation begegnet werden. In zukünftigen Arbeiten gilt

es, den vorgestellten Ansatz weiterzuentwickeln. Grundsätzlich sind noch mehrere Fragestellungen offen, etwa:

- Ein Hauptproblem der Bitletrepräsentation stellt das Lokalisationsproblem im Bitletbereich dar. Dieses tritt sowohl bei der eingeschränkten Auswertung von Vergleichsoperationen als auch bei der Entwicklung von Indexstrukturen auf. Auch wenn in der Literatur zu diesem allgemeinen Problem verschiedene Lösungsvorschläge bestehen, gilt es zu überprüfen, ob und wie diese im vorliegenden Fall eingesetzt werden können.
- Für eine effiziente Realisierung einer der Vergleichsoperationen (AND-Operation) wird ein anwendungsspezifisches Wissen zu den Wiederholungsmustern bzw. ihren Bitletrepräsentationen benötigt. Auch wenn das Verfahren grundsätzlich auch ohne solches Wissen auskommt, gilt es zu untersuchen, ob das benötigte Wissen für entsprechende Anwendungsdomänen ermittelt werden kann und wie sich dies auf die Effizienz des Verfahrens auswirkt.
- Die Konstruktion von Bitlets (bzw. von Bitletbasen) "auf Knopfdruck" und in Abhängigkeit bestimmter Mustereigenschaften, würde die Kompressionseigenschaften des Verfahrens verbessern und zu einer ausgeprägten Kompaktheit der Bitletrepräsentationen führen.

Integration des Beschreibungsmodells in Prozessmetamodelle: Die entwickelten Lösungen und Algorithmen sind an konkrete Formen von Informationssystemen zu adaptieren. Aus den in der Einleitung erwähnten Gründen, besteht vor allem für prozessorientierte Informationssysteme (engl. process-aware information systems, kurz PAIS) ein hoher Bedarf. Es gilt zu untersuchen, wie das vorgestellte zyklenorientierte Modell zur Beschreibung periodischer Spezifikationen in entsprechende Prozessmodelle integriert oder auf solche übertragen werden kann.

Bitletrepräsentation in der Laufzeitumgebung temporaler PAIS: Die systemseitige Unterstützung temporaler Prozesse im Allgemeinen und periodizitätsbehafteter Prozesse im Besonderen ist für viele Anwendungen essentiell. Ohne eine entsprechende Unterstützung ist es für Anwender oftmals sehr schwer, sich potenziell anbahnende Konflikte, wie etwa zeitliche Kollisionen in Termin- und Arbeitsplänen oder Engpässe im Ressourcenbereich, im Vorhinein zu erkennen. Hierzu wird in der Laufzeitumgebung eines PAIS die Ausführung aller (laufenden) Prozesse über ihre komplette Lebensdauer hinweg gesteuert und überwacht. Mit dem Vorliegen der Bitletrepräsentation, kann untersucht werden, wie dessen operationale Möglichkeiten (Vergleichsoperationen) bezüglich periodischer Aktivitäten mit den bestehenden Verfahren zur Überprüfung von Zeitbedingungen und Berechnung von Zeitplänen in (zeitabhängigen) Prozessen kombiniert werden können.

Anhang

A Algorithmus zur Ableitung von Wiederholungsmustern bei HSP-Bäumen

In den folgenden Algorithmen werden zum besseren Verständnis unterschiedliche Typen von Knotenlisten L, L^* und Q verwendet. Diese weisen die folgende Semantik auf:

- L: Eine Knotenliste, die nur H-Knoten enthält.
- L^*: Eine Knotenliste, die auch S-Knoten und P-Knoten enthält.
- Q: Diese Liste von Knotenlisten repräsentiert die kompakte Leveldarstellung.

A.1 Die Funktion $getPattern()$

Die Funktion $getPattern()$ entspricht der Hauptfunktion zur Ableitung von Wiederholungsmustern. Hierbei wird der vollständige Baum von Level zu Level durchlaufen und pro Level das Wiederholungsmuster des Levels bestimmt ($getLevelPattern()$) und die Knoten des nächsttieferen Levels bestimmt ($getNextLevelList()$). Die Implementierung der Funktion zeigt Algorithmus A.1.

Algorithmus A.1 $getPattern(T, z_T) \mapsto L$

input:
T: HSP-Baum
z_T: Instanziierungspunkt von T bzw. des entsprechenden Periodizitätsschemas
output:
L: Wiederholungsmuster des HSP-Baum als Knotenliste $L = (v_0, ..., v_k)$

1: $L = NewList()$
2: $InsertElement(L, root^T)$
3: **while** $\exists n \in L : n \neq$ Blattknoten **do**
4: $L_{tmp} = getLevelPattern(T, z_T, L)$
5: $L = getNextLevelList(T, L_{tmp})$
6: **end while**
7: **return** L

A.2 Bestimmung der Knoten des nächsttieferen Levels

In Abschnitt 5.2.2 haben wir gezeigt, dass bei der Leveltraversierung Blattknoten von einem Level zum nächsttieferen Level mit übertragen werden. Diese Semantik der Traversierung wird durch die Hilfsfunktion $getNextLevelList()$ sichergestellt. Die Implementierung der Funktion zeigt Algorithmus A.2.

Des Weiteren gehen wir von einer Funktion $UB(n) \mapsto (\#A, pos)$ aus, die für jeden Knoten n dessen Übertrag zur Berücksichtigung von Levelabhängigkeiten bestimmt. Die Übergabe des Übertrags an den nächsttieferen Level findet in der untenstehenden Funktion statt. Zur Vereinfachung des Algorithmus wird davon ausgegangen, dass jeder Knoten stets einen entsprechenden Übertrag aufweist. Wie gezeigt, existiert ein "neutraler" Übertrag, der diese Vereinfachung des Algorithmus ermöglicht.

Algorithmus A.2 $getNextLevelList(T, L) \mapsto L^*$

input:
T : HSP-Baum
L : Wiederholungsmuster eines Levels als Knotenliste $L = 0, ..., n$
output:
L^* : Levelknotenliste des nachfolgenden Levels

```
 1: L* := NewList()
 2: for i = 0 to n do
 3:    node = GetElement(L, i)
 4:    if node <> Blattknoten then
 5:       succ = succ(T, node)
 6:       UB^succ = UB^node
 7:       InsertElement(L*, succ)
 8:    else
 9:       InsertElement(L*, node)
10:    end if
11: end for
12: return L*
```

A.3 Erzeugung des Wiederholungsmusters pro Level

Die Funktion $getLevelPattern()$ erzeugt das Wiederholungsmuster eines Levels. Durch die (Hilfs-) Funktion $getCompactTransformedLevel()$ wird im ersten Schritt die kompakte und transformierte Leveldarstellung erzeugt. Diese wird dann im zweiten Schritt mit der Funktion $getUnpackedLevel()$ entpackt. Zur Berücksichtigung von gekoppelten Auswahl- oder Projektionsebenen werden diese Schritte solange wiederholt, bis in der

Algorithmus A.3 $getLevelPattern(T, z_T, L^*) \mapsto L$

input:
T: HSP-Baum T
z_T: Instanziierungspunkt von T bzw. des entsprechenden Periodizitätsschemas
L^*: Levelknotenliste $L^* = (v_0, ..., v_{m-1})$ eines Levels
output:
L: Wiederholungsmuster des Levels als Knotenliste L

1: **repeat**
2: $Q = getCompactTransformedLevel(T, z_T, L^*)$
3: $L^* = getUnpackedLevel(Q)$
4: **until** $\nexists n \in L^* : NT^n \in \{\texttt{SNODE}, \texttt{PNODE}\}$
5: **return** L^*

Ergebnisliste nach dem Entpacken keine S-Knoten oder P-Knoten mehr enthalten sind. Die Implementierung der Funktion zeigt Algorithmus A.3.

A.4 Erzeugung der kompakten transformierten Leveldarstellung

In der Funktion $getCompactTransformedLevel()$ werden für jede Ebene in einem Level im ersten Schritt die individuellen Wiederholungsmuster durch den rekursiven Aufruf der Funktion $getPattern()$) ermittelt und anschließend transformiert. Das Abbruchkriterium der Rekursion bilden HSP-Bäume, die nur aus H-Knoten bestehen. Wie leicht im Algorithmus zu erkennen ist, erfolgt bei H-Knoten kein weiterer rekursiver Aufruf der Funktion $getPattern()$. H-Knoten und 0-Knoten werden lediglich unverändert in die Ergebnisliste übernommen. Folglich fallen bei HSP-Bäumen, die nur aus H-Knoten bestehen, keine weiteren Rekursionsschritte an und die Rekursion kann terminieren.

Bei Auswahlebenen wird nach der Bestimmung des individuellen Wiederholungsmusters direkt die Projektionsfunktion $\mathcal{P}()$, wie in Abschnitt 5.5 beschrieben, aufgerufen. Bei Projektionsebenen hingegen wird die zusätzliche Funktion $getProjectionPattern()$ aufgerufen, die das individuelle Wiederholungsmuster transformiert. Das Ergebnis der Funktion bildet die transformierte kompakte Leveldarstellung Q. Die Implementierung der Funktion zeigt Algorithmus A.4.

Die Berücksichtigung von Levelabhängigkeiten erfolgt durch den zweiten Aufruf der Projektionsfunktion $\mathcal{P}$. Die Funktion $UB^v = (wdh, pos_{ub})$ bestimmt den Übertrag (wdh, pos_{ub}) des Knotens v, dessen Werte in der Projektionsfunktion verwendet werden (siehe Abschnitt 5.6). Zur Vermeidung von Fallunterscheidungen wird in den Funktion davon ausgegangen, dass jeder Knoten einen entsprechenden Übertrag aufweist. Wir zeigen später in Abschnitt A.5, weshalb diese Vereinfachung vorgenommen werden kann.

Algorithmus A.4 $getCompactTransformedLevel(T, z_T, L^*) \mapsto Q$

input:
T: HSP-Baum T
z_T: Instanziierungspunkt von T bzw. des entsprechenden Periodizitätsschemas
L^*: Levelknotenliste $L^* = (v_0, ..., v_{m-1})$ eines Levels
output:
Q: Kompakte Leveldarstellung des Levels

1: $Q := NewList()$
2: $i = 0$
3: **repeat**
4: $v = GetElement(L^*, i)$
5: **if** $NT^v \in \{\texttt{HNODE}, \texttt{ONODE}\}$ **then**
6: $InsertElement(Q, NewList(v))$
7: **else if** $NT^v = \texttt{SNODE}$ **then**
8: $SP^v = (z_1, l_P, l_A, POS = \{pos\})$
9: $L_{tmp} = getPattern(\sigma^v \cup \sigma^v_{leaf}, z_T)$
10: $L_{tmp} = \mathcal{P}(L_{tmp}, l_P, l_A, pos)$
11: $UB^v = (wdh, pos_{ub})$
12: $L_{tmp} = \mathcal{P}(L_{tmp}, wdh, l_A, pos_{ub})$
13: $InsertElement(Q, L_{tmp})$
14: **else if** $NT^v = \texttt{PNODE}$ **then**
15: $SP^v = (z_1, l_P, l_A, POS)$
16: $L_{tmp} = getPattern(\pi^v \cup \pi^v_{leaf}, z_1)$
17: $L_{tmp} = getProjectionPattern(L_{tmp}, SP^v, z_T)$
18: $UB^v = (wdh, pos_{ub})$
19: $L_{tmp} = \mathcal{P}(L_{tmp}, wdh, l_A, pos_{ub})$
20: $InsertElement(Q, L_{tmp})$
21: **end if**
22: i++
23: **until** $i > Size(L^*)$
24: **return** Q

A.4.1 Berücksichtigung von Projektionsebenen

Die Funktion $getProjectionPattern()$ bildet eine Hilfsfunktion, die das Wiederholungsmuster einer Projektionsebene im Fall einer mehrelementigen Menge an Positionsangaben bestimmt. Wie in Abschnitt 5.4.2 beschrieben, wird hierzu pro Position p_i eine Projektion des individuellen Wiederholungsmusters vorgenommen. Das Ergebnis bildet die kompakte Darstellung des Wiederholungsmusters der Projektionsebene, die im

Algorithmus A.5 $getProjectionPattern(L_i^*, SP, z_T) \mapsto L_o^*$

input:
L_i^*: Knotenlisten der Projektionsebene
SP Projektionsparameter $SP = (z_1, l_P, l_A, POS = \{p_0, ..., p_n\})$ der Projektionsebene
z_T Instanziierungspunkt von T bzw. des entsprechenden Periodizitätsschemas
output:
L_o^*: Knotenliste als Wiederholungsmuster der Projektionsebene

1: $Q := NewList()$
2: **for** $x = 0$ to n **do**
3: $pos = GetElement(POS, x)$
4: $L_{tmp} = \mathcal{P}(L_i^*, l_P, l_A, pos)$
5: $L_{tmp} = \mathcal{R}(L_{tmp}, z_T, z_1)$
6: $InsertElement(Q, L_{tmp})$
7: **end for**
8: $L_o^* = getUnpackedLevel(Q)$
9: **return** L_o^*

zweiten Schritt mit der Funktion $getUnpackedLevel()$ (siehe später) entpackt wird. Das Ergebnis L_o^* von $getProjectionPattern()$ bildet das Wiederholungsmuster der Projektionsebene. Die Implementierung der Funktion zeigt Algorithmus A.5.

A.4.2 Projektionsfunktion

Die Funktion $\mathcal{P}()$ entspricht einer Hilfsfunktion, welche die vollständige Projektion für das übergebene Muster bzw. Liste L_i^* an der Position i berechnet. Die Implementierung der Funktion zeigt Algorithmus A.6.

A.4.3 Rotationsfunktion

Die Funktion $\mathcal{R}()$ entspricht einer Hilfsfunktion, die ein Muster bzw. Liste L_i^* mit Instanziierungspunkt z auf den übergebenen Rotationspunkt r (z.B. Instanziierungspunkt eines zweiten Musters) rotiert. Das Ergebnis bildet das rotierte Muster L_o^*. Die Implementierung der Funktion zeigt Algorithmus A.7.

Algorithmus A.6 $\mathcal{P}(L_i^*, n, m, pos) \mapsto L_o^*$

input:
L_i^*: Knotenliste $L_i^* = (v_0, ..., v_{m-1})$ des Projektionsschemas
n: Musterlänge des Periodizitätsschema
m: Musterlänge des Projektionsschemas
pos: Musterposition $0 \leq pos < n$ der Projektion im Periodizitätsschemas
output:
L_o^*: Knotenliste des Projektionsschemas nach erfolgter periodischer Projektion

1: $x = 0$; $L_o^* = NewList()$
2: **repeat**
3: $j = (x * n + pos)\ mod\ m$ {Projektionsfunktion}
4: $InsertElement(L_o^*, x, GetElement(L_i^*, j))$
5: $x++$
6: **until** $x > \frac{m}{ggT(m,n)}$
7: **return** L_o^*

Algorithmus A.7 $\mathcal{R}(L_i^*, r, z) \mapsto L_o^*$

input:
L_i^*: Knotenliste des zu rotierenden Wiederholungsmuster $L_i^* = (v_0, ..., v_{m-1})$
r: Rotationspunkt $r \in \mathbb{Z}$
z: Instanziierungspunkt des zu rotierenden Periodizitätsschemas $z \in \mathbb{Z}$
output:
L_o^*: Wiederholungsmuster des rotierten Wiederholungsmustern als Knotenliste

1: $x = 0$; $L_o^* = NewList()$
2: **repeat**
3: $d = r - z$
4: $m = size(L_i^*)$
5: $j = (x + d)\ mod\ m$ {Rotationsfunktion}
6: $InsertElement(L_o^*, x, GetElement(L_i^*, j))$
7: $x++$
8: **until** $x > m$
9: **return** L_o^*

A.5 Entpacken der kompakten Leveldarstellung

Die Funktion $getUnpackedLevel(Q) \mapsto L$ entpackt die übergebene, kompakte Leveldarstellung Q. Die Implementierung der Funktion $getUnpackedLevel(Q) \mapsto L$ zeigt Algorithmus A.8.

Um die logische Bedingung beim Entpacken, dass alle Knoten mindestens einmal "besucht" werden, zu erfüllen, wird im ersten Schritt die hierfür notwendige Anzahl an Durchläufen ermittelt (siehe Abschnitt 5.2.5). Diese Anzahl an Durchläufen j ergibt sich durch die Bestimmung des kleinsten gemeinsamen Vielfachen der Längen der enthaltenen Muster bzw. Listen. Die Funktion `kgV()` ermittelt hierbei das kleinste gemeinsame Vielfache der übergebenen Menge an Zahlen. Nach dem Entpacken werden alle 0-Knoten aus der Ergebnisliste L gelöscht. Die Ergebnisliste L der Funktion bildet das Wiederholungsmuster des Levels.

Im letzten Schritt wird für jeden Knoten der Ergebnisliste L bzw. des Levelmusters der Übertrag zur Berücksichtigung von Levelabhängigkeiten ermittelt. Wir gehen hier von einer Funktion $getCarry(L, n) \mapsto \#Copy \times Pos$ aus, die für einen Knoten n der Liste L den Umfang der Menge an Kopien ($\#Copy$) von Knoten n und dessen Position (pos) in der Menge an Kopien ermittelt (Positionen beginnend bei 0). Dieser ermittelte Übertrag wird dem Knoten n zugewiesen bzw. der entsprechenden Funktion $UB^n = (\#Copy, pos)$. Zur Vereinfachung der Algorithmen wird jedem Knoten ein entsprechender Übertrag zugewiesen, d.h. auch solchen Knoten, die nicht kopiert wurden, sondern lediglich einzeln in der Liste enthalten sind. Für diese Knoten wird jedoch der "neutrale" Übertrag $(1, 0)$ verwendet. Wie leicht nachvollziehbar ist, führt die Anwendung der Projektionsfunktion für eine Liste L mit den neutralen Übertrag zu der gleichen Liste L, d.h. $\mathcal{P}(L, 1, l_A, 0) = L$.

In den folgenden Implementierungen gehen wir von einer besonderen Form von Zugriffsfunktion auf die Listen zur Speicherung von Elementen bzw. Knoten aus. Die Zugriffsfunktion $getNextElement(L)$ mit $L = (v_0, v_1, ..., v_n)$ startet beim initialen Aufruf beim ersten Element, d.h Element v_0. Die Liste "merkt" sich das letzte ausgewählte Element und beim erneuten Zugriff bzw. Aufruf der Funktion wird der Nachfolger des "gemerkten" Elements zurückgeliefert, d.h. beim zweiten Aufruf das Element v_1, beim dritten Aufruf das Element v_2 und so weiter. Nach dem Zugriff auf das letzte Element v_n der Liste wird zurück auf das erste Element v_0 "gesprungen".

Algorithmus A.8 $getUnpackedLevel(Q) \mapsto L$

input:
Q: Kompakte Leveldarstellung $Q = (L_0 = (v_0, ..., v_m), L_1, ..., L_h)$ als Liste von Knotenlisten
output:
L^*: Entpackte Darstellung des Levels als Levelknotenliste L^*

1: $i = 0$
2: $L = NewList()$
3: $j = Size(Q) * \texttt{kgV}(Size(L_0), ..., Size(L_h))$
4: **repeat**
5: $\quad L_{tmp} = GetNextElement(Q)$
6: $\quad node = GetNextElement(L_{tmp})$
7: $\quad InsertElement(L, node)$
8: $\quad i++$
9: **until** $i > j$
10: $\forall n \in L^* : NT^n = \texttt{ONODE}$ do $DeleteElement(L^*, n)$
11: $\forall n \in L^* : UB^n = getCarry(L^*, n)$
12: **return** L

B Beweise

B.1 Beweis von Korollar 5.2

Korollar B.1: *Gegeben zwei Muster* $M_1 = (a_0, \ldots, a_{n-1})$ *und* $M_2 = (b_0, \ldots, b_{m-1})$ *mit* $m, n \in \mathbb{N}$. *In dem Vektor* $v = (p(i,0), \ldots, p(i,k-1))$ *ist jedes Element von Muster* M_2 *maximal einmal enthalten, wobei für* k *gilt:* $k = \frac{|M_2|}{ggT(m,n)}$.

Beweis: Wir zeigen zunächst, dass $k = \frac{|M_2|}{ggT(m,n)}$, die minimale Periodenlänge der Positionen der beiden Mustern M_1 und M_2 darstellt.

$p(i,w) = (w \cdot n + i) \bmod m,\ m = |M_2|$

Angenommen es gibt eine Periode, also $p(i,x) = p(i,y)$ für irgendwelche x, y, dann haben wir:

$x \cdot n + i = y \cdot n + i \bmod m$

Wir bringen alles auf eine Seite:

$(x-y) \cdot n = 0 \bmod m.$

Also ist m ein Teiler von der linken Seite: Es gibt $z \in \mathbb{Z}$ mit $z \cdot m = (x-y) \cdot n$. Jetzt ist g der größte gemeinsame Teiler von m und von n, also $m = g \cdot m'$ sowie $n = g \cdot n'$, sodass folgt

$z \cdot g \cdot m' = (x-y) \cdot g \cdot n'.$

Nach Dividieren durch g auf beiden Seiten ergibt sich: $z \cdot m' = (x-y) \cdot n'$.

Da g aber der -größte- gemeinsame Teiler von m und n war sind die Restfaktoren m' und n' jetzt teilerfremd: $ggT(m', n') = 1$. m' ist ein Teiler der rechten Seite, aber nicht von n', also muss m' die Differenz $x - y$ teilen. Also ist $x - y$ mindestens so groß wie $m' = \frac{m}{ggT(m,n)} = k$, damit ist x mindestens k.

Also kann in den ersten k Elementen von v keine Periode auftreten.

Umgekehrt findet die erste Periode schon für $y = k$ und $x = 0$ statt, denn es gilt $p(i,y) = p(i,m') = (m' \cdot n) + i \bmod m = (m \cdot n') + i \bmod m = i$ und ebenso $p(i,x) = p(i,0) = i$.

Die Gleichung $m'n = mn'$ gilt, weil ein Versetzen des Kommas bedeutet, dass der gemeinsame Teiler g von einem Faktor in den anderen geschoben wird, was den Wert des Produkts nicht verändert.

Also ist $k = \frac{m}{ggT(m,n)}$ genau die minimale Periodenlänge. ∎

Da k die minimale Periodenlänge der Positionen der beiden Muster M_1 und M_2 darstellt, folgt daraus, dass die Elemente aus M_2 innerhalb dieser minimalen Periodenlänge nicht doppelt vorkommen können.

B.2 Beweis von Satz 6.1

Für die Zerlegung eines Vektorraums V^m in seine Teiler-Räume muss bewiesen werden, dass sich stets ein induktives Basensystem für V^m aus den Basen seiner Teiler-Räume ergibt bzw. konstruieren lässt. Aus beweistechnischen Gründen wird hierbei die Menge der Teiler von V^m nicht als Menge, sondern als ein Graph bzw. Baum verstanden, der wie folgt definiert ist.

Definition B.1 (Teilerbaum): *Ein Teilerbaum ist ein Graph $G = (\mathbb{N}, \|)$, wobei (a, b) eine Kante ist, falls a ein maximaler Teiler von b ist, d.h. falls $a|b$ aber $a \neq b$ ist, und a maximal mit dieser Eigenschaft ist. Dann ist G ein Baum mit Wurzel 1. Mit G_n bezeichnen wir den Teilgraph, der nur die Knoten a mit $a|n$ enthält.*

Ein Teilerbaum ist ein Baum, in dem eine Kante zwischen zwei Werten a und b kennzeichnet, dass a einen maximalen bzw. den größten Teiler von b darstellt.

Mit Hilfe des Teilerbaums kann ein (formales) Konstruktionsverfahren (induktive Basiswahl), angegeben werden, das zu einer induktiven Basis führt. Die induktive Basenwahl bildet die formale Entsprechung und Übertragung auf den Teilerbaum der Grundidee, dass eine induktive Basis für V^n aus seinen "kleineren" Teileräumen konstruiert werden kann.

Definition B.2 (Induktive Basiswahl): *Eine induktive Basiswahl ordnet jeder Kante (a, b) des Teilerbaums G eine Menge $B_{a,b}$ von Vektoren des V^b zu, so dass gilt:*

- *Die Mengen $B_{a,b}$ sind paarweise disjunkt, und jede endliche Vereinigung solcher Mengen ist linear unabhängig.*
- *Die Menge*

$$B_n = \bigcup_{(a,b)\in G_n} B_{a,b}$$

ist jeweils eine Basis des V^n.

Vereinfacht ausgedrückt, wird in der obigen Definition, durch Vereinigung, der einer jeden Kante im Baum G_n zugeordneten Basisvektoren, eine induktive Basis für den Raum V^n gebildet.

Satz B.1: *Es existiert eine induktive Basenwahl.*

Beweis: Diese Aussage wird mit vollständiger Induktion über den Graphen G bewiesen.

Induktionsanfang ist der Teilgraph G_1 (ein Knoten, keine Kante), für den die Aussage trivialerweise erfüllt ist.

Angenommen das System (B_n) (bzw. $B_{a,b}$) ist für einen Teilgraph G' schon induktiv konstruiert, und n ist ein Knoten von G dessen Kindknoten sämtlich in G' liegen, dann haben wir die Mengen $B_{a,n}$ zu konstruieren, wobei $a \parallel n$ über die maximalen Teiler von n läuft. Es sei

$$B' = \bigcup_{(a,b)\in G'} B_{a,b}$$

die Vereinigung aller Basisvektoren in G'. Diese sind nach Konstruktionsvorschrift linear unabhängig, da sie alle im $n-$dimensionalen Raum V^n liegen ist $|B'| \leq n$. Nach dem Ergänzungssatz der linearen Algebra für Vektorräume kann B'durch Hinzufügen von l-Vektoren aus dem V^n zu einer Basis des V^n ergänzt werden[40]. Diese l Vektoren verteilen wir beliebig auf die zu konstruierenden Mengen $B_{\frac{n}{p},n}$ (nicht notwendig zu gleichen Teilen, zudem können einige oder alle Mengen leer bleiben), und erhalten ein induktives Basensystem für den Teilgraph von G, der aus G' durch Hinzufügen von n entsteht. Der Induktionsschluss liefert die Aussage des Satzes, da jeder endliche Teilgraph von G in einem G_n enthalten ist. ■

Aus der Konstruktionsvorschrift folgt sofort die Induktivität der Basiswahl: Ist $a|b$, so ist $B_a \subseteq B_b$ nach Verlängerung. Mit der Existenz einer induktiven Basenwahl ist gleichzeitig die Existenz eines induktiven Basensystems für alle Vektorräume bewiesen.

B.3 Beweis von Satz 6.2

Beweis: Zu zeigen ist, dass in jedem Schritt die bereits vorhandenen Basisvektoren linear unabhängig zu den Einheitsvektoren sind, wenn man "von rechts" mit Einsen füllt.

Also sei n eine Zahl für die Standardbitletbasis B_n erzeugt werden soll. Wir haben schon alle Basisvektoren für Teiler $a|n$. Vor der Auffüllung sieht die Matrix so aus:

$$\begin{pmatrix} B_1 & B_1 & B_1 & \cdots & B_1 \\ B_2 & & B_2 & & B_2 \\ \vdots & & & & \vdots \end{pmatrix}$$

wobei die Blöcke B unterschiedlich groß sein können.

Nach Induktionsannahme sind die Zeilen dieser Matrix linear unabhängig. Wir fügen jetzt einen Einheitsvektor "von rechts" dazu und zeigen, dass das Ganze immer noch

linear unabhängig ist. Da jedes B_j invertierbar ist (weil die Zeilen linear unabhängig sind) gibt es Zeilenumformungen, die die ganze Matrix umbauen zu:

$$\begin{pmatrix} I_1 & I_1 & I_1 & \cdots & I_1 \\ I_2 & & I_2 & & I_2 \\ \vdots & & & & \vdots \end{pmatrix}$$

Die einzelnen I_j stellen jetzt Einheitsmatrizen dar. Bei dieser Umformung bleibt die lineare Unabhängigkeit erhalten [40].

Angenommen, man kann mit diesen Zeilen eine 1 ganz rechts linear kombinieren. Weil die I_j ihre unterste Eins immer unten RECHTS stehen haben, muss man dazu zwingend, die letzten Zeilen aus jedem Block verwenden, alle anderen tragen nichts bei. Sei a_j der Koeffizient in der Zeilenkombination zur letzten Zeile des Blocks B_j. Da wir eine 1 unten RECHTS kombinieren, muss $a_1 + \ldots + a_k = 1$ sein. Im kleinsten Block I_1 oben LINKS tritt eine Eins in der letzten Zeile des Blocks unten RECHTS auf. Diese Zeile trägt also mit Koeffizient a_1 zu einem Element in der neuen letzten Zeile bei. Die letzten Zeilen der anderen Blöcke haben aber in dieser Spalte keine 1 weil die I_j für $j > 1$ breiter sind als I_1. Also entsteht unten LINKS die Komponente a_1. Wir wollen aber den Einheitsvektor mit nur einer Eins RECHTS kombinieren, also muss $a_1 = 0$ sein. Das heißt die Blöcke I_1 sind an der Linearkombination nicht beteiligt, wir können sie streichen. Jetzt haben wir wieder eine Matrix vom gleichen Typ: Einheitsmatrizen, die nach unten breiter werden. Mit dem gleichen Argument ist nun der Koeffizient $a_2 = 0$. Per Induktion sind alle Koeffizienten Null. Also ist es nicht möglich, den Einheitsvektor (0...01) linear aus den vorhandenen Vektoren zu kombinieren, wir dürfen also damit auffüllen.

Nachdem wir den Einheitsvektor (0...01) hinzugefügt haben, müssen wir zeigen, dass der nächste hinzuzufügende Vektor (0..010) auch linear unabhängig ist. An einer Linearkombination kann der zuvor zugefügte Vektor (0....01) nicht teilhaben, weil er die 1 am falschen Platz hat, und die anderen Vektoren aus der Basismatrix auch nicht mit dem gleichen Argument wie zuvor. Das funktioniert solange, bis die eine 1 sich in der gleichen Spalte wie der erste Block I_1 befindet (obwohl sie in einer ganz anderen Zeile befindet), dann funktioniert das Argument nicht mehr weil Blöcke sich ab dieser Spalte "schief" überlappen, und das Füllen ist beendet. ■

Abbildungsverzeichnis

2.1 Übersicht der Klassen von Periodizitätsschemas 13
2.2 Bitvektordarstellung zweier Wiederholungsmuster 30
2.3 Schnittoperation von Wiederholungsmustern mit gleicher Periodenlänge 31
2.4 Schnittoperation von Wiederholungsmustern mit unterschiedlicher Periodenlänge 32

3.1 Darstellung von Granularitäten und periodischen Spezifikationen im Time-Granularity Modell 47
3.2 Darstellung von Granularitäten und periodischen Spezifikationen im Partitionen Modell 48
3.3 Grundstrukturen des zyklenorientierten Modells 51
3.4 Beispiels von Periodizitätsangaben und ihre Abbildung auf die Grundstrukturen des zyklenorientierten Modell 52
3.5 Verwendung von Labels zur Beschreibung eines Zyklus `Chemotherapie` der Länge 28 Tage sowie von Medikamentengaben `A` vom 1. bis zum 14. Tag sowie Gaben von Medikament `B` am 1. und 8. Tag 52
3.6 Beispiel für eine horizontale Strukturierung 54
3.7 Beispiel für die vertikale Strukturierung eines Chemotherapiezyklus mit Medikamentengaben A und B sowie Bluttest BT 55
3.8 Vertikale Strukturierung am Beispiel von Öffnungszeiten 56
3.9 Abbildung von Periodizitätsschemas im zyklenorientierten Modell . . . 57
3.10 Periodizitätsschemas im zyklenorientierten Modell 57
3.11 Übersicht zu Verkettungen in Labelausdrücke 58
3.12 Beispiele einer Zyklus-Zyklusverkettung 59
3.13 Beispiele einer Zyklus-Zyklusverkettung 60
3.14 Beispiele einer horizontalen Navigation bei Zyklus-Zyklusverkettung 61
3.15 Beispiel einer Top-Down Navigation 62
3.16 Beispiel einer Bottom-Up Navigation 62
3.17 Beispiel einer horizontalen Navigation 63
3.18 BNF-Notation für Labelausdrücke 64
3.19 Übersicht zu den Änderungsoperationen 65
3.20 Verlängern (Verkürzen) von Zyklenlängen 65
3.21 Einfügen (Löschen) von Teilzyklen 66
3.22 Einfügen (Löschen) von Labels 66
3.23 Verschieben von Labels 67
3.24 Einfügen (Löschen) von vertikalen Ebenen 67

3.25 Zusammenfassen (Aufteilen) von Teilzyklen 67
3.26 Beispiel für ein einfaches periodisches Auswahlschema 69
3.27 Beispiel für ein strukturiertes periodisches Auswahlschema 70
3.28 Beispiel für ein periodisches Auswahlschema mit Ausnahmen 71
3.29 Übersicht zu Auswahloperationen 71
3.30 Zyklusauswahl mit einfachen Auswahlschema 72
3.31 Zyklusauswahl bei strukturierten Periodizitätsschemas 73
3.32 Zyklspositionsauswahl mit strukturiertem Auswahlschema 74
3.33 Beispiel für eine Projektionsauswahl 76

4.1 Grundidee der Darstellung von Periodizitätsschemas durch HSP-Bäume 84
4.2 Grafische Darstellung der Knotentypen des HSP-Baums 87
4.3 Einfache Beispiele für HSP-Bäume 89
4.4 Darstellung eines Periodizitätsschemas im HSP-Baum und dessen entsprechende Darstellung im zyklenorientierten Modell 90
4.5 Beispiel für die Repräsentation einer horizontale Strukturierung im HSP-Baum 91
4.6 Beispiel einer vertikalen Strukturierung im HSP-Baum 92
4.7 Beispiel eines HSP-Baum für ein Periodiziätsschemas mit mehrfach strukturierten Zyklen 94
4.8 Beispiel einer Zyklusauswahl im zyklenorientierten Modell 95
4.9 Beispiel für die endliche Repräsentation einer Auswahlmenge und (periodischen) Auswahlschemas 96
4.10 Wiederholungsmuster des Auswahlschemas über der Auswahlmenge . . . 97
4.11 Beispiel für die Abbildung einer Auswahl im HSP-Baum 98
4.12 Abbildung einer Zyklusauswahl im HSP-Baum 99
4.13 Vereinfachte Darstellung von Auswahlebenen 102
4.14 Grundformen von Auswahlebenen 103
4.15 Beispiel eines HSP-Baums mit Faktor $\neq 1$ beim Wechsel der Schachtelungstiefe 105
4.16 Beispiel einer Auswahloperation mit mengenwertiger Auswahlmenge . . 105
4.17 Auswahlebenen im HSP-Baum bei mengenwertiger Auswahlmenge . . . 106
4.18 Zyklenorientierte Darstellung des Wiederholungsmusters des Auswahlschemas über der Auswahlmenge 107
4.19 Beispiel einer Projektionsauswahl im zyklenorientierten Modell 108
4.20 Beispiel für die Abbildung einer Projektionsauswahl im HSP-Baum . . . 108
4.21 Beispiel einer Projektion mit strukturierten Periodizitätsschema PS_2 . . 110
4.22 Vereinfachte Darstellung von Projektionsebenen 111
4.23 Verschiedene Grundformen von Projektionsebenen 112
4.24 Projektionsauswahl mit mengenwertiger Projektionsmenge 114
4.25 Beispiel eines HSP-Baums mit 0-Knoten 115
4.26 Übersicht zu Verkettungen in Labelausdrücke 118

4.27 Beispiel einer Top-Down Navigation im HSP-Baum und im zyklenorientierten Modell . . . 119
4.28 Beispiel einer Top-Down Navigation im HSP-Baum und im zyklenorientierten Modell bei einfacher Auswahlebene . . . 121
4.29 Beispiel für transitive Blätter einer Auswahlebene . . . 124
4.30 Komplexes Beispiel einer Top-Down Navigation . . . 128
4.31 Beispiel einer Zyklus-Zyklluspositionsverkettung im HSP-Baum und im zyklenorientierten Modell . . . 131
4.32 Problem der Zyklus-Zyklluspositionsverkettung im HSP-Baum . . . 132
4.33 Komplexes Beispiel einer Zyklus-Zyklluspositionsverkettung . . . 137
4.34 Beispiel für Dead-Leafs bei einer Projektionsauswahl . . . 140
4.35 Übersicht zu den Änderungsoperationen . . . 143
4.36 Einfaches Löschen von Zyklen im HSP-Baum . . . 144
4.37 Periodisches Löschen von Zyklen im HSP-Baum . . . 144
4.38 Aufsplitten eines Zyklus im HSP-Baum . . . 147
4.39 Übersicht zu Auswahloperationen . . . 149
4.40 Anpassung der Schachtelungstiefe und des Knotentyps des Auswahlschemas 150
4.41 Ersetzen der Blätter des Auswahlschemas durch Kopien des von der Auswahl betroffenen Knotens v_2 . . . 150
4.42 Einfügen der Auswahlebene mit ihren Blättern in den übergebenen HSP-Baum T_{in} . . . 151
4.43 Initialer HSP-Baum . . . 157
4.44 HSP-Baum nach Anpassen der Zykluslängen (nach Schritt 1 und 2) . . . 158
4.45 HSP-Baum nach Einfügen der Labels (nach Schritt 3 und 4) . . . 158
4.46 HSP-Baum des Auswahlschemas (nach Schritt 5.1 und 5.2) . . . 159
4.47 HSP-Baum nach erfolgter Zyklusauswahl (nach Schritt 6 und 7) . . . 160
4.48 HSP-Baum des Periodizitätsschemas G_{Woche} (nach Schritt 8.1 und 8.2) 160
4.49 HSP-Baum nach erfolgter Projektionsauswahl (nach Schritt 9) . . . 161
4.50 HSP-Baum nach Löschen aller Bluttests (`BT`), die auf einen Sonntag (`So`) fallen (nach Schritt 10) . . . 162
4.51 HSP-Baum mit verschiedenen (semantischen) Granularitätsebenen . . . 163
4.52 HSP-Baum mit periodischen Angaben auf mehreren Granularitätsebenen 164
4.53 Grundidee separater Granularitätsebenen und Kalender in Verbindung mit dem HSP-Baum . . . 165

5.1 Zusammenhang HSP-Baum und Wiederholungsmuster . . . 171
5.2 Ablaufübersicht des Algorithmus zur Ableitung von Wiederholungsmustern 173
5.3 Level-Order Traversierung von geordneten Bäumen . . . 174
5.4 Beispiel für die Level-Order Traversierung bei HSP-Bäumen . . . 175
5.5 HSP-Baum mit einzelner einfacher Auswahlebene . . . 176
5.6 Level mit einzelner einfacher Auswahlebene . . . 177
5.7 Level mit mehreren Auswahlebenen . . . 178
5.8 Level mit geschachtelter Auswahlebene . . . 178

5.9 Transformation der Muster einer Projektionsebene 179
5.10 Entpacken der kompakten Leveldarstellung mit einzelnem Wiederholungsmuster 180
5.11 Entpacken der kompakten Leveldarstellung mit mehreren Wiederholungsmustern 181
5.12 Level mit einzelner Auswahlebene und 0-Knoten 182
5.13 Entpacken der kompakten Leveldarstellung mit 0-Knoten 183
5.14 Level mit einer gekoppelten Auswahl 183
5.15 Auswertung eines Levels mit gekoppelter Auswahlebene 184
5.16 Traversierung eines HSP-Baums mit einfacher Auswahlebene 185
5.17 Beispiel für eine Projektionsebene im HSP-Baum 186
5.18 Problem der Projektion bei Wiederholungsmustern 187
5.19 Beispiel für Wiederholungsmuster mit unterschiedlichen Instanziierungspunkten und deren Vereinheitlichung 191
5.20 Projektion mit unterschiedlichen Instanziierungspunkten 194
5.21 Ableitung des korrekten Wiederholungsmusters einer Projektionsebene . 197
5.22 Vereinfachte Darstellung eines Levels ohne 0-Knoten zum Zeitpunkt der Projektionsoperation 198
5.23 Vereinfachte Darstellung eines Levels mit 0-Knoten zum Zeitpunkt der Projektionsoperation 198
5.24 Vereinfachte Darstellung eines Levels ohne 0-Knoten zum Zeitpunkt der Auswertung 199
5.25 Vereinfachte Darstellung eines Levels mit 0-Knoten zum Zeitpunkt der Auswertung 200
5.26 Beispiel für die Konstruktion des Ergebnismusters der Projektion 201
5.27 Transformation von Projektionsebenen bei mehrelementigen Positionsmengen 202
5.28 Level mit mehreren abhängigen Auswahlebenen 204
5.29 Logische Auswertung der kompakten Leveldarstellung eines Levels mit mehreren abhängigen Auswahlebenen 204
5.30 Semantisch korrekte Auswertung eines Levels nach dem Löschen einer der abhängigen Auswahlebenen 206
5.31 Beispiel einer periodischen Auswahl 207
5.32 Darstellung der periodischen Auswahl als Wiederholungsmuster 207
5.33 Transformation der kompakten Leveldarstellung mit abhängigen Auswahlebenen 210
5.34 Transformation der kompakten Leveldarstellung mit abhängigen Auswahlebenen und verkürzten Ergebnismustern 211
5.35 Darstellung des indirekten Kopierens von Teilbäumen 213
5.36 Semantisch korrektes Entpacken eines Levels bei bestehenden Levelabhängigkeiten 214
5.37 Entpacken der kompakten Leveldarstellung nach der Bereinigung von Levelabhängigkeiten 215

5.38 Übergabe des Übertrags auf den nächst tieferen Level 216
5.39 Übersicht der logischen Schritte des Algorithmus und die korrespondierenden Funktionen 218
5.40 HSP-Baum des Auswertungsbeispiels 219
5.41 Darstellung der Zwischenergebnisse der Auswertung pro Level 220
5.42 Ausgabe des Prototyps mit den Endergebnissen pro Level 222

6.1 Schnittoperation von Wiederholungsmustern mit gleicher Periodenlänge 229
6.2 Schnittoperation von Wiederholungsmustern mit unterschiedlicher Periodenlänge 230
6.3 Schematische Darstellung des Zusammenhangs von Bitlet- und Fourier-Transformation 233
6.4 Beispiel einer Funktion mit diskretem Wertebereich 235
6.5 Wiederholungsmuster als Stufenfunktion 236
6.6 Zusammenhang von Raumeinbettung und Verlängerung bei Mustern unterschiedlicher Periodenlänge 237
6.7 Beispiel für musterangepasste Basisvektoren 245
6.8 Ermittlung des geometrischen Mittel an Nicht-Null-Belegungen pro Spalte bei Standardbitletbasen verschiedener Dimension durch die prototypische Realisierung der Fast-Bitlet-Transformation 263
6.9 Schnittoperation von Wiederholungsmustern mit gleicher Periodenlänge 266
6.10 Auswertungsdiagramm der bitletbasierten Realisierung von Vergleichsoperationen 267
6.11 Relevante Bitlets für den Ereignishorizont = 17 284
6.12 Relevante Bitlets für den Ereignishorizont = 17 − 19 287
6.13 Eingeschränkte Vergleichsoperation bei Wiederholungsmustern und Ereignishorizont > Musterlänge 293
6.14 Eingeschränkte Vergleichsoperation bei Wiederholungsmustern und Ereignishorizont < Musterlänge 293
6.15 Beispiel einer Zugriffstruktur im Zeitbereich und Bitletbereich 298

7.1 Gesamtansicht des Prototyps 306
7.2 Implementierte Verkettungsfunktionen für Labelausdrücke 307
7.3 Repository zur Verwaltung der Labels 308
7.4 Implementierte Änderungsoperationen 308
7.5 Beispiel für eine Auswahloperation 309
7.6 Bitletdarstellung im Prototyp 311
7.7 Darstellung der Bitletkonsole 311
7.8 Initiale Periodizitätsschema 315
7.9 Periodizitätsschema mit eingefügten Labels 316
7.10 Periodizitätsschema (Ausschnitt) mit Zyklusauswahl 317
7.11 Berücksichtigung der Bestellzeitpunkte 318

7.12 Wiederholungsmuster (Ausschnitt) für die Bestellzeitpunkte von Medikament B 318

8.1 Zusammenspiel der entwickelten Konzepte am Beispiel eines prozessorientierten Informationssystems 324

Tabellenverzeichnis

3.1 Übersicht zu Eingabe- und Ausgabetypen von Verkettungen 63

4.1 Übersicht der verwendeten Basisfunktionen im HSP-Baum 88
4.2 Auswahlparameter und ihre Semantik 153
4.3 Projektionsparameter und ihre Semantik 155

5.1 Übersicht der verwendeten Basisfunktionen im HSP-Baum 195
5.2 Semantische Zuordnung der Projektionsparameter 196
5.3 Semantische Zuordnung der Auswahlparameter 209

6.1 Einzelvergleiche der beiden Repräsentationsvarianten 230
6.2 Gesamtmenge zweistelliger Boolescher Funktionen 266
6.3 Gegenbeispiele für die Verträglichkeit boolescher Funktion mit der Bitlettransformation bezüglich der Standardbitletbasis 269
6.4 Realisierungkomplexität der NOT-Operation im $\mathbb{Z}^2$-Vektorraum 289
6.5 Realisierungsaufwände bei unbeschränkten Vergleichsoperationen für unterschiedliche Repräsentationsvarianten 291
6.6 Realisierungsaufwände bei eingeschränkten Vergleichsoperationen für unterschiedliche Repräsentationsvarianten 295

Literaturverzeichnis

[1] L. Anselma. Recursive representation of periodicity and temporal reasoning. In *Proc. TIME'04, 11th Int. Symposium on Temporal Representation and Reasoning*, pages 52–59, Tatihou, France, July 2004.

[2] S. Bassil, R. Keller, and P. Kropf:. A workflow-oriented system architecture for the management of container transportation. In *BPM '04: Proceedings of the 2th International Conference on Business Process Management*, pages 116 –131, Potsdam, Germany, June 2004 2004.

[3] T. Bauer. Kooperation von Projekt- und Workflow-Management-Systemen. *Informatik - Forschung und Entwicklung*, 19(2):74–86, November 2004.

[4] T. Bauer, M. Reichert, and P. Dadam. Intra-subnet load balancing in distributed workflow management systems. *Int. Journal of Cooperative Information Systems*, 12(3):205–323, 2003.

[5] Alberto Belussi, Carlo Combi, and Gabriele Pozzani. Towards a formal framework for spatio-temporal granularities. In *Proc. TIME'08, 15th Int. Symposium on Temporal Representation and Reasoning*, pages 49–53, à Monteéal, Canada,, June 2008.

[6] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An acess control model supporting periodicity constraints and temporal reasoning. *IEEE Transaction on Database Systems*, 23(3):231–285, September 1998.

[7] Elisa Bertino, Dolores Cuadra, and Paloma Martínez. An object-relational approach to the representation of multi-granular spatio-temporal data. In *17th International Conference Advanced Information Systems Engineering (CAiSE'05)*, pages 119–134, Porto, Portugal, June 2005.

[8] C. Bettini, S. Jajodia, and S. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer-Verlag Berlin Heidelberg New York, 2000.

[9] C. Bettini, S. Jajodia, and S. Wang. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):296–306, 2002.

[10] C. Bettini and S. Mascetti. An efficient algorithm for minimizing time granularity periodical representations. In *Proc. TIME'05, 12th Int. Symposium on Temporal Representation and Reasoning*, pages 20– 25, Burlington, Vermont, USA, June 2005.

[11] C. Bettini, S. Mascetti, and X. Sean Wang. Mapping calendar expressions into periodical granularities. In *Proc. TIME'04, 11th Int. Symposium on Temporal Representation and Reasoning*, pages 96–102, Tatihou Island,France, July 2004.

[12] C. Bettini and R. de Sibi. Symbolic representation of user-defined time granularities. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):53–92, 2000.

[13] Christian Blatter. *Wavelets - Eine Einführung.* Vieweg & Sohn Verlag, 2003.

[14] D. Bresolin, A. Montanari, and G. Puppis. Time granularities and ultimately periodic automata. In *Proc. JELIA 2004, 9th European Conference on Logics in Artificial Intelligence*, volume LNAI 3229, pages 513–525, Lisbon, Portugal, September 2004.

[15] C. Bussler. Workflow instance scheduling with project management tools. In *Proc. DEXA 98, Int. Workshop on Database and Expert Systems Applications*, pages 753–758, Vienna, Austria, August 1998.

[16] Fabio Casati, Stefano Ceri, Stefano Paraboschi, and Giuseppe Pozzi. *Database Support for Workflow Management: The WIDE Project*, chapter 7, pages 141–168. Kluwer Academic Publishers, 1999.

[17] R. Chandra, A. Segev, and M. Stonebraker. Implementing calendars and temporal rules in next generation databases. In *Proc. Int. Conf. on Data Engineering*, pages 264–273, Houston, USA, February 1994.

[18] C. Combi, M. Franceschet, and A. Peron. A logical approach to represent and reason about calendars. In *Proc. TIME'02, 9th Int. Symposium on Temporal Representation and Reasoning*, pages 134–140, Manchester, UK, July 2002.

[19] C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 4(1):51–77, 2004.

[20] C. Combi, M. Gozzi, J.M. Juarez, B. Oliboni, and G. Pozzi. Conceptual modeling of temporal clinical workflows. In *Proc. TIME'07, 14th Int. Symposium on Temporal Representation and Reasoning*, volume 0, pages 70–81, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[21] C. Combi and B. Oliboni. Managing valid time semantics for semistructured multimedia clinical data. In *Current Trends in Database Technology, EDBT 2006*, volume 4254/2006 of *LNCS*, pages 375–386, Munich, Germany, March 2006. Springer Berlin / Heidelberg.

[22] C. Combi and R. Posenato. Controllability in temporal conceptual workflow schemata. In *BPM '09: Proceedings of the 7th International Conference on Business Process Management*, pages 64–79, Berlin, Heidelberg, 2009. Springer-Verlag.

[23] Carlo Combi, Matteo Gozzi, José M. Juárez, Roque Marín, and Barbara Oliboni. Querying clinical workflows by temporal similarity. In *11th Conference on Artificial Intelligence in Medicine (AIME 2007)*, volume 4594 of *Lecture Notes in Computer Science*, pages 469–478, Amsterdam, The Netherlands, Juli 2007. Springer.

[24] Carlo Combi and Giuseppe Pozzi. Task scheduling for a temporalworkflow management system. In *Proc. TIME'06, 13th Int. Symposium on Temporal Representation*

and Reasoning, pages 61–68, Budapest, Hungary, June 2006. IEEE Computer Society.

[25] J.F. Condotta, G. Ligozat, M. Saade, and S. Tripakis. Ultimately periodic simple temporal problems (upstps). In *Proc. TIME'06, 13th Int. Symposium on Temporal Representation and Reasoning*, pages 69–77, Budapest, Hungary, June 2006. IEEE Computer Society.

[26] D. Cukierman and J. Delgrande. Expressing time intervals and repetition within a formalization of calendars. *Journal of Computational Intelligence*, 14(4):563–597, 1998.

[27] D. Cukierman and J. Delgrande. A formalization of structured temporal objects and repetition. In *Proc. TIME'00, 7th Int. Workshop on Temporal Reasoning and Representation*, pages 83–87, Cape Breton, Canada, July 2000.

[28] D. Cukierman and J. Delgrande. The sql time theory: A formalization of structured temporal objects and repetition. In *Proc. TIME'04, 11th Int. Symposium on Temporal Representation and Reasoning*, pages 28–35, Tatihou, France, July 2004.

[29] P. Dadam and M. Reichert. Towards a new dimension in clinical information processing. In *Proc. MIE'00, 16th Int. Congress of the European Federation for Medical Informatics*, pages 295–301, Hanover, Germany, September 2000.

[30] P. Dadam, M. Reichert, and K. Kuhn. Clinical workflows - the killer application for process-oriented information systems? In *Proc. BIS 2000, 4th Int. Conf. on Business Information Systems*, pages 36–59, Poznan, Poland, April 2000.

[31] U. Dal Lago and A. Montanari. Calendars, time granularities, and automata. In *Proc. SSTD 2001, 7th Int. Symposium on Spatial and Temporal Databases*, volume LNCS 2121, pages 279–298, Los Angeles, US, July 2001.

[32] U. Dal Lago, A. Montanari, and G. Puppis. Towards compact and tractable automaton-based representation of time granularity. In *Proc. ICTCS'03, 8th Italien Conference on Theoretical Computer Science*, volume LNCS 2841, pages 72–85, Bertinoro, Italia, October 2003.

[33] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and applied Mathematics, Philadelphia, 1992.

[34] S. Demri. LTL over integer periodicity constraints (extended abstract). In *Proc. of 7th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2987 of *Lecture Notes in Computer Science*, pages 121–135, Barcelona, Spain, March 2004.

[35] M. Dumas, W. M. van der Aalst, and A.H. Ter Hofstede. *Process Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley-Interscience, September 2005.

[36] C.E. Dyreson, W.S. Evans, H. Lin, and R.T. Snodgrass. Efficiently supporting temporal granularities. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):568–587, 2000.

[37] J. Eder and E. Panagos. Managing time in workflow systems. In *in Workflow Handbook 2001, Layna Fischer (Ed.), Future Strategies Inc.*, pages 109–132, 2000.

[38] L. Egidi and P. Terenziani. A lattice of classes of user-defined symbolic periodicities. In *Proc. TIME'04, 11th Int. Symposium on Temporal Representation and Reasoning*, pages 13–20, Tatihou, France, July 2004.

[39] L. Egidi and P. Terenziani. A mathematical framework for the semantics of symbolic languages representing periodic time. In *Proc. TIME'04, 11th Int. Symposium on Temporal Representation and Reasoning*, pages 21–27, Tatihou, France, July 2004.

[40] Gerd Fischer. *Lineare Algebra*, volume 16. Vieweg+Teubner Verlag, August 2008.

[41] I.A. Goralwalla, Y. Leontiev, M. Oezsu, D. Szafron, and C. Combi. Temporal granularity: Completing the puzzle. *Journal of Intelligent Information Systems*, 16(1):41–63, 2001.

[42] G. H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, 6th edition, Juli 2008.

[43] Holger Hoheisel. *Temporale Geschäftsprozesmodellierung.* PhD thesis, Universität Bern, 2000.

[44] Barbara Burke Hubbard. *Wavelets. Die Mathematik der kleinen Wellen.* Birkhäuser Verlag, 1997.

[45] F. Kabanza, J.-M. Stévenne, and P. Wolper. Handling infinite temporal data. In *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 392–403, Nashville, US, April 1990.

[46] F. Kabanza, J.-M. Stévenne, and P. Wolper. Handling infinite temporal data. *Journal of Computer and System Sciences*, 51(1):3–17, 1995.

[47] J. P. Kahane. *Fourier Series and Wavelets.* Gordon & Breach Publishing Group, 1995.

[48] M. Kalb and P. Dadam. Unterstützung von Periodizität in Informationssystemen – Herausforderungen und Lösungsansätze. *Springer, Informatik - Forschung und Entwicklung*, Vol. 450(8):109–125, 2007. ISSN 1865-2034 (Print) 1865-2042 (Online).

[49] M. Kalb, C. Dittrich, and P. Dadam. Support of relationships among moving objects on networks. Technical Report 2008-5, Ulmer Informatik-Berichte, ISSN 0939-5091, Februar 2008.

[50] M. Kalb, K. Schneider, and G. Specht. T-XPath: Ein zeitliches Modell für XML-Datenbanken. In *In. Proc. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW'03)*, pages 157–166, Leipzig, Germany, 2003.

[51] M. Kalb and G. Specht. Zeitliche Verwaltung XML-basierter Metadaten in digitalen Bibliotheken. In *9. Kongress der IuK-Initiative der Wissenschaftlichen Fachgesellschaften in Deutschland (IuK 2003)*, pages 227–238, Osnabrück, Deutschland, März 2003.

[52] U.D. Lago, A. Montanari, and G. Puppis. On the equivalence of automaton-based representations of time granularities. In *Proc. TIME'07, 14th Int. Symposium on Temporal Representation and Reasoning*, volume 0, pages 82–93, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[53] B. Leban, D. McDonald, and D. Forster. A representation for collections of temporal intervals. In *Proc. 5th National Conference on Artificial Intelligence*, pages 367–371, Philadelphia, US, August 1986.

[54] R. Lenz and M. Reichert. It support for healthcare processes - premises, challenges, perspectives. *Data & Knowledge Engineering*, 61(1):39–58, 2007.

[55] H. Li and Y. Yang. Dynamic checking of temporal constraints for concurrent workflows. *Electronic Commerce Research and Applications, Elsevier*, 4(2):124–142, 2005.

[56] Stephan G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Machine Intell.*, 11:674–693, 1989.

[57] O. Marjanovic and M. Orlowska. On modeling and verification of temporal constraints in production workflows. *Knowledge & Inf. Systems*, 1, 1999.

[58] Andreas Meister. *Numerik linearer Gleichungssysteme.* Vieweg Verlag, Wiesbaden, 2. Auflage edition, 2005.

[59] S. Miksh and R. Kosara. Communicating time-oriented, skeletal plans to domain experts lucidly. In *Proc. DEXA 99, Int. Workshop on Database and Expert Systems Applications*, LNCS 1677, pages 1041–1051, Florence, Italy, August 1999.

[60] Robert Müller. *Event-Oriented Dynamic Adaptation of Workflows: Model, Architecture and Implementation.* PhD thesis, Universität Leipzig, 2002.

[61] A. Montanari, A. Peron, and A. Policriti. Decidable theories of omega-layered metric temporal structures. *Logic Journal of the IGPL*, 7(1):79–102, January 1999.

[62] D. Müller, J. Herbst, M. Hammori, and M. Reichert. It support for release management processes in the automotive industry. In *BPM '06: Proceedings of the 4th International Conference on Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 368–377, Vienna, Austria, September 2006.

[63] B. Mutschler, M. Reichert, and J. Bumiller. Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors, and implications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(3):280–291, 2008.

[64] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proc. First Int. Conference on Information and Knowledge Management*, pages 161–168, Baltimore, US, November 1992.

[65] P. Ning, X.S. Wang, and S. Jajodia. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):5–38, 2002.

[66] H.J. Ohlbach. The role of labeled partitionings for modeling periodic temporal notions. In *Proc. TIME'04, 11th Int. Symposium on Temporal Representation and Reasoning*, pages 60–63, Tatihou, France, July 2004.

[67] Oracle. *Oracle8i Time Series User's Guide, Release 8.1.5, Part Number A67294-01.* Oracle Corporation, 1999.

[68] T. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen*, volume Auflage: 4. Spektrum Akademischer Verlag, Januar 2002.

[69] M. Reichert, T. Bauer, T. Fries, and P. Dadam. Realisierung flexibler, unternehmensweiter workflow-anwendungen mit adept. In *Proc. Gemeinsame Arbeitskonferenz GI, VOI BITKOM, OCG, Tele TrusT, Elektronische Geschäftsprozesse*, pages 217–228, Klagenfurt, Austria, September 2001.

[70] M. Reichert and P. Dadam. ADEPTflex - supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, March/April 1998.

[71] M. Reichert, S. Rinderle-Ma, and P. Dadam. Flexibility in process-aware information systems. *T. Petri Nets and Other Models of Concurrency*, 2:115–135, 2009.

[72] Manfred Reichert. *Dynamische Ablaufänderungen in Workflow-Management-Systemen.* PhD thesis, Universität Ulm, 2000.

[73] Stefanie Rinderle. *Schema Evolution in Process Management Systems.* PhD thesis, Universität Ulm, 2004.

[74] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* 2nd edition. SIAM Society for Industrial & Applied Mathematics, 2003.

[75] S. Sadiq, O. Marjanovic, and M. Orlowska. Managing change and time in dynamic workflow processes. *Intl. Journal of Cooperative Information Systems*, 9(1+2):1–25, 2000.

[76] Uwe Schöning. *Logik für Informatiker.* 5.Auflage. Spektrum Akademischer Verlag, 2000.

[77] Uwe Schöning. *Theoretische Informatik - kurzgefasst.* 4.Auflage. Spektrum Akademischer Verlag, 2001.

[78] Stephanie Spranger. *Calendars as Types - Data Modeling, Constraint Reasoning, and Type Checking with Calendars.* PhD thesis, Ludwig Maximilians Universität München, 2005.

[79] P. Terenziani. Integrating calendar dates and qualitative temporal constraints in the treatment of periodic events. *IEEE Knowledge and Data Engineering*, 9(5):763–783, 1997.

[80] P. Terenziani. Temporal reasoning with class and instances of events. In *Proc. TIME'02, 9th Int. Symposium on Temporal Representation and Reasoning*, pages 100–107, Manchester, UK, July 2002.

[81] P. Terenziani. Symbolic user-defined periodicity in temporal relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 15(2):489–509, 2003.

[82] P. Terenziani and L. Anselma. Towards an integrated approach dealing with part-of, instance-of and periodicity constraints. In *Proc. TIME'03, 10th Int. Symposium on Temporal Representation and Reasoning and 4th Int. Conf. on Temporal Logic*, pages 37–46, Cairns, Australia, July 2003.

[83] P. Terenziani and L. Anselma. Temporal reasoning about composite and/or periodic events. *Journal of Experimental & Theoretical Artificial Intelligence*, 18(1):87–115, March 2006.

[84] P. Terenziani, L. Anselma, and S. Montani. Periodicity-based temporal constraints. In *AI*IA 2005: Advances in Artificial Intelligence 9th Congress of the Italian Association for Artificial Intelligence*, number 3673 in LNCS, pages 62–65, Milan, Italy, September 2005.

[85] P. Terenziani, C. Carlini, and S. Montani. Towards a comprehensive treatment of temporal constraints in clinical guidelines. In *Proc. TIME'02, 9th Int. Symposium on Temporal Representation and Reasoning*, pages 20–27, Manchester, UK, July 2002.

[86] D. Toman, J. Chomicki, and D. S. Rogers. Datalog with integer periodicity constraints. In *Proc. of the 1994 Int. Symposium Logic Programming*, pages 189–203, Melbourne, Australia, November 1994.

[87] A. Tuzhilin and J. Clifford. On periodicity in temporal databases. *Information Systems*, 20(8):619–639, 1995.

[88] Wil M. P. van der Aalst. Tomtom for business process management (tomtom4bpm). In *21th International Conference Advanced Information Systems Engineering (CAiSE'09)*, volume 5565 of *LNCS*, pages 2–5, Amsterdam, The Netherlands, June 2009. Springer-Verlag Berlin Heidelberg New York.

[89] M. Y. Vardi. A temporal fixpoint calculus. In *Conference Record of the Conference on Principles of Programming Languages*, pages 250–259, San Diego, California, United States, January 1988.

[90] W. v.d. Aalst and K. v.d. Hee. *Workflow Management.* MIT Press, 2002.

[91] B. Weber, S.W. Sadiq, and M. Reichert. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23(2):47–65, 2009.

[92] Mathias Weske. *Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects., Habilitationsschrift.* Fachbereich Mathematik und Informatik, Universität Münster, 2000.

[93] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer-Verlag, Berlin Heidelberg, 2007.

[94] J. Wijsen. A string-based model for infinite granularities. In *Proc. of the AAAI-2000 Workshop on Spatial and Temporal Granularity*, pages 9–16, Austin, US, July 2000.

[95] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, January/February 1983.