Universität Ulm
Fakultät für Informatik
Abteilung Datenbanken und Informationssysteme (DBIS)

Semantische Konflikte in adaptiven Prozess-Management-Systemen

Diplomarbeit

vorgelegt von

Michael Nahler

1. Gutachter: Dr. S. Rinderle

2. Gutachter: Prof. Dr. P. Dadam

31. Oktober 2005

Zusammenfassung

Prozess-Management-Systeme (PMS) kommen bei der Steuerung von Unternehmensabläufen zum Einsatz. Dazu werden die Abläufe in Prozesse des PMS abgebildet. Um wettbewerbsfähig zu sein, müssen Unternehmen flexibel auf Marktgeschehnisse und Änderungen der Bedingungen reagieren können. Adaptive PMS bieten diese Flexibilität, indem sie Änderungen an den ausgeführten Prozessen und festgelegten Prozesstypen ermöglichen. Änderungen erfordern besondere Maßnahmen, um die Konsistenz und Qualität der Prozesse nicht zu gefährden. Dazu wurde ein umfassendes Rahmenwerk für Prozesstyp- und Instanzänderungen entwickelt, um die syntaktische Korrektheit von Änderungen sicherzustellen. Nicht berücksichtigt werden dabei bisher jedoch semantische Aspekte von Prozessen. Deren Einbeziehung ist von entscheidender Bedeutung, um eine weitere Ebene für Korrektheitsgarantien einzuführen. Die Bestimmung der semantischen Korrektheit erfordert die Erfassung semantischer Informationen zu einem Prozess. Anhand dieser Daten kann die semantische Korrektheit nach festgelegten Kriterien bewertet werden. Diese Bewertung der semantischen Korrektheit darf nicht zu langanhaltenden Verzögerungen in der Ausführung der Prozesse führen. Deshalb ist es wichtig, dass die Auswertung der semantischen Korrektheit effizient abläuft.

Die Diplomarbeit stellt Betrachtungen über die Beschreibung der Semantik von Prozessen an. Es wird untersucht, unter welchen Bedingungen ein Prozess semantisch nicht korrekt ist, wobei insbesondere auf die Auswirkungen von Änderungen auf die semantische Korrektheit eingegangen wird. Änderungen können den Charakter von Ad-hoc-Änderungen auf Instanz-Ebene oder von Prozesstyp-Änderungen aufweisen. Die Besonderheiten von Ad-hoc-Änderungen werden genauso untersucht, wie die Änderung von Prozesstypen. Desweiteren werden Untersuchungen über die effiziente Umsetzbarkeit der eingeführten Konzepte zur Sicherstellung der semantischen Korrektheit angestellt. Betrachtungen zur Laufzeit werden dabei ebenso berücksichtigt wie die Auswirkungen auf die Speicherbelastung.

Die Sicherstellung der semantischen Korrektheit ist ein konsequenter Schritt in der Weiterentwicklung adaptiver PMS und stellt einen Meilenstein auf dem Weg zu adaptiven PMS dar, die eine noch höhere Qualität der Prozesse garantieren.

Vorwort

Mit dieser Diplomarbeit durfte ich ein sehr interessantes, herausforderndes und weitgreifendes Thema untersuchen. An erster Stelle möchte ich mich bei meiner Betreuerin Frau Dr. Stefanie Rinderle bedanken, die mir die Bearbeitung dieses Themas ermöglicht hat. Insbesondere in der nervenaufreibenden Schlussphase, als ich mich in den vielfältigen Aspekten der Thematik zu verlieren drohte, half sie mir, die wesentlichen Punkte nicht aus den Augen zu verlieren.

Ein weiterer besonderer Dank gilt Herrn Prof. Dr. P. Dadam, in dessen Abteilung ich meine Diplomarbeit schreiben durfte, und der sich bereit erklärt hat, die Funktion des zweiten Gutachters zu übernehmen.

Ganz besonders möchte ich mich bei meinen Eltern Christa und Leopold, meiner Freundin Steffi, meiner Schwester Christine und meinem Schwager Konrad bedanken, die immer an mich geglaubt und mich immer wieder aufgebaut haben. Eure Liebe, Unterstützung und viel Verständnis haben diese Arbeit erst ermöglicht. Auch meine zwei kleinen Patenkinder Florian und Johanna, über deren Fortschritte ich auch im Laufe der Arbeit genauestens informiert wurde, haben mir geholfen, die Bodenhaftung in schwierigen Momenten nicht zu verlieren.

Einen besonderen Dank möchte ich Hilmar Acker aussprechen. Die anregenden Diskussionen haben mir sehr geholfen und meinen Blick für die Probleme der behandelten Thematik geschärft. Ein weiterer besonderer Dank gilt Markus Lauer, der immer ein offenes Ohr für mich hatte.

Auch allen Freunden und Mitstudenten, die mich auf meinem Weg zur Erstellung dieser Arbeit unterstützt haben, sei es durch fachliche Gespräche, nervliche "Aufbauhilfe" oder durch Ablenkung, möchte ich an dieser Stelle meinen Dank aussprechen. Ohne euch wäre das Unternehmen "Diplomarbeit" ungleich schwieriger geworden.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Betrachtung der Unternehmensabläufe als Geschäftsprozesse ist ein nicht mehr wegzudenkender Bestandteil der heutigen Unternehmenskultur. Die explizite Erfassung der Geschäftsprozesse stellt die Basis für die Unterstützung der Unternehmensprozesse durch ein Prozess-Management-System (PMS) dar. Ein Prozess-Management-System organisiert die Verteilung der innerhalb der verschiedenen Prozesse anfallenden Aufgaben an entsprechende Bearbeiter und verwaltet den Fortschritt der Prozesse. Der Nutzen beim Einsatz eines PMS in einem Unternehmen ist mit der Erwartung der Kostensenkung, sowie der Produktivitäts- und Qualitätssteigerung der angebotenen Produkte und Dienste verknüpft [Aal01]. Um diese Ziele zu erreichen und die Konkurrenzfähigkeit zu sichern, sind die Qualität und Anpassbarkeit der abgebildeten Geschäftsprozesse ein entscheidender Faktor. Gerade im Hinblick auf den global immer härter werdenden Wettbewerb mit häufig wechselnden Anforderungen kann die Geschwindigkeit, mit der Unternehmen auf veränderte Situationen und Ausnahmebedingungen reagieren, oftmals den entscheidenden Marktvorteil sichern. Flexibilität und Anpassungsfähigkeit sind die Schlagworte, die eine entscheidende Säule für den Unternehmenserfolg darstellen.

Die meisten am Markt eingesetzten PMS bilden zur Modellierzeit lediglich die typisch auftretenden Situationen im Ablauf eines Prozesses ab. Seltene und besonders zu behandelnde Ausnahmesituationen sind häufig nicht abgedeckt. Nicht nur Ausnahmesituationen führen zu Änderungen in bestehenden Abläufen. Beispielsweise sich ändernde juristische Rahmenbedingungen können eine Änderung der unternehmensinternen Prozesse nach sich ziehen. Die Unterstützung für Änderungen und Ausnahmesituationen ist deshalb ein entscheidendes Kriterium für den breiten Einsatz von PMS in Unternehmen. PMS, die evolutionäre Geschäftsprozesse unterstützen, werden adaptive PMS genannt. Das in der Abteilung DBIS an der Universität Ulm entwickelte PMS ADEPT

stellt einen Vertreter der adaptiven PMS dar und legt damit einen Schwerpunkt auf die Unterstützung von Prozessänderungen.

Ein Geschäftsprozess wird in einem PMS durch ein so genanntes Prozessschema repräsentiert, das alle relevanten Informationen über die Struktur und die Daten eines Prozesses enthält. Die Ausführung eines Prozesses geschieht in Instanzen des Prozessschemas. Die Anpassung modellierter Prozesse durch auftretende Ausnahmesituationen betrifft häufig die Instanzen. Dies resultiert in Instanzänderungen, auch Ad-hoc-Änderungen genannt, die das Prozessschema unberührt lassen. Dauerhafte Änderungen an der Basisstruktur eines Prozesses hingegen haben das Prozessschema zum Ziel. Das in [RRD04, RRD04a, RRD04b, RRD04c, Rin04] vorgestellte Rahmenwerk zur Unterstützung von Änderungen des Prozessschemas und zur Behandlung von Ausnahmesituationen ist applikationsneutral und sichert die Konsistenz von Prozessen. Arbeitsschritte bzw. Aktivitäten von Prozessen werden dabei lediglich als "Black Boxes" betrachtet, die Vor- und Nachbedingungen besitzen und einen bestimmten Teilausschnitt des gesamten Geschäftsprozesses kapseln. Die tatsächliche Aufgabenstellung einer Aktivität wird dabei nicht berücksichtigt. Es kann beispielsweise Änderungen geben, die auf der strukturellen Ebene verträglich zu einem Prozess sind, nicht jedoch in Hinblick auf ihre Semantik. In Abbildung 1.1 findet sich ein Beispiel für diesen Zusammenhang, der in [Rin04, S. 115] an einem ähnlichen Beispiel eingeführt wird. Dabei handelt es sich um einen stark vereinfachten Prozess, wie er bei der Einlieferung eines Patienten in ein Krankenhaus aussehen könnte. In einem konkreten Fall des dargestellten Prozesses, d.h. in einer bestimmten Instanz, entscheidet der behandelnde Arzt, dass der Patient zusätzlich zur ersten Medikation eine weitere Medikation bekommen soll. Die aktivierte und damit die nächste auszuführende Aktivität der Instanz ist das Entlassen des Patienten. Das Einfügen einer Aktivität vor der momentan aktivierten ist möglich, wobei die Aktivierung der Aktivität `Entlassung` entfernt wird und die neu eingefügte Aktivität stattdessen aktiviert wird. Auf semantischer Ebene kann es nun zu zwei möglichen Situationen kommen.

- Der Arzt hat alle Wechselwirkungen des Medikaments B mit anderen Medikamenten bedacht und ist zu der Feststellung gelangt, dass Medikament B keinerlei Wechselwirkungen mit Medikament A hat.
- Der Arzt hat übersehen, dass Medikament B und Medikament A nicht zusammen verabreicht werden dürfen, da starke Wechselwirkungen zwischen den beiden Medikamenten bestehen.

Im ersten Fall gibt es kein Problem, wenn die neue Aktivität `Medikation B` eingefügt wird, im zweiten Fall kann dies jedoch gesundheitliche Folgen für den Patient haben. Es gibt also im zweiten Fall einen semantischen Konflikt zwischen den zwei Aktivitäten `Medikation A` und `Medikation B`. Die Änderungsoperation des Einfügens von `Medikation B` ist in diesem Fall semantisch unverträglich zur Instanz I.

Die in [Rin04] eingeführten Konzepte werden durch die Einbeziehung der Semantik ergänzt, was einen Meilenstein in der Weiterentwicklung adaptiver PMS darstellt. Dies

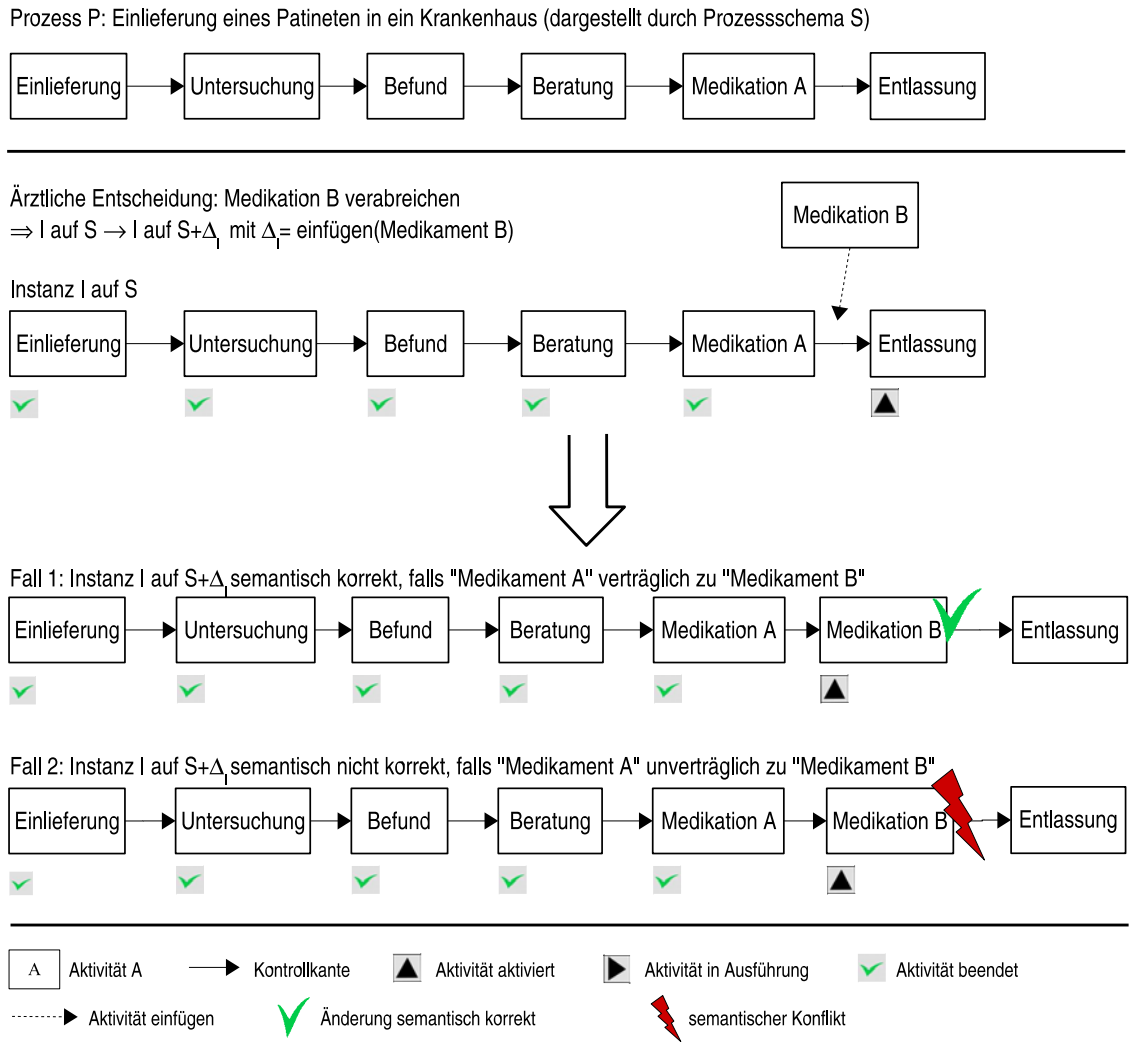


Abbildung 1.1.: Semantischer Konflikt durch Einfügeoperation

umfasst die Erfassung und Auswertung semantischer Informationen zu Prozessaktivitäten. Da Semantik dabei der zentrale Begriff ist, wird dieser nun näher beleuchtet.

1.1. Der Semantik-Begriff

Der Begriff Semantik wird in unterschiedlichen Kontexten benutzt und erscheint dadurch als ungenau, da oft nicht klar ist, was der Begriff Semantik in einem bestimmten Zusammenhang bedeutet. Deshalb ist es notwendig, den Begriff Semantik für das jeweilige Umfeld genau festzulegen.

Der Semantik-Begriff muss also im Kontext PMS definiert werden, wobei der Schwerpunkt der Betrachtungen auf die Semantik von Aktivitäten in Geschäftsprozessen gelegt wird. Aktivitäten beziehen sich auf ein zu bearbeitendes Objekt. Es kann sich dabei um einen materiellen Gegenstand, wie zum Beispiel ein bestelltes Produkt, als auch um einen immateriellen, virtuellen Gegenstand wie ein elektronisches Dokument handeln. Dieser einer Aktivität zugrundeliegende Gegenstand kann auf verschiedene Arten bearbeitet bzw. manipuliert werden. Abbildung 1.2 illustriert dies am Beispiel eines einfachen Prozesses, der die Abwicklung einer Bestellung bei einem Versandhändler repräsentiert. In der Aktivität `Packliste erstellen` wird ein elektronisches Dokument angelegt, das alle direkt vorrätigen bestellten Artikel auflistet. Im folgenden Arbeitsschritt wird anhand dieser Packliste das Paket für den Versand an den Kunden zusammengestellt. Beiden Aktivitäten liegt also das Dokument Packliste zugrunde, jedoch wird es in den Aktivitäten auf unterschiedliche Weise benutzt.

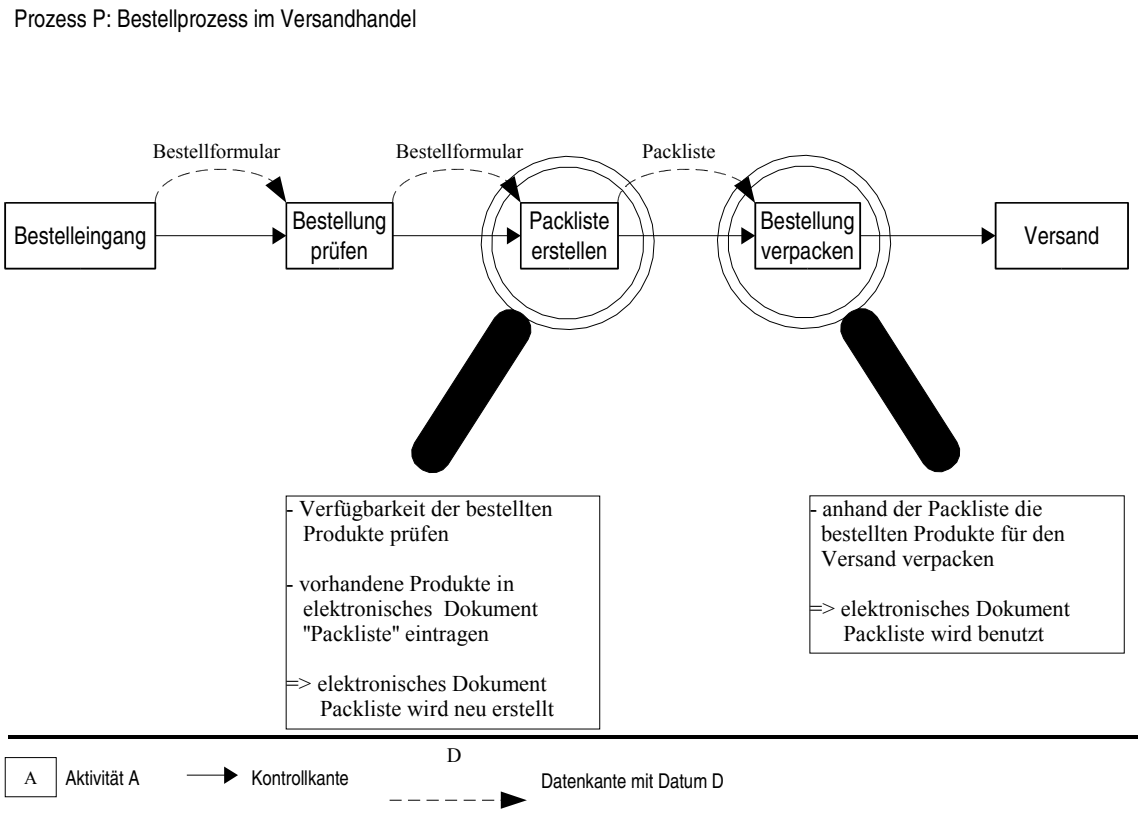


Abbildung 1.2.: Bestellvorgang

In einer ersten Näherung kann der Begriff Semantik in Bezug auf die Aktivitäten in einem Prozess folgendermaßen informal definiert werden:

Definition 1.1 (Semantik einer Aktivität - erste Näherung)

Die Semantik einer Aktivität erschließt sich durch die innerhalb der Aktivität zu manipulierenden materiellen oder immateriellen Gegenstände, also den Objekten der Aktivität, und der Art der Manipulation dieser Objekte.

1.2. Semantik in adaptiven PMS

Adaptive PMS müssen für die Konsistenz der Kontroll- und Datenflüssen bei Änderungen auf der Prozessschema- und auf der Instanzebene Sorge tragen. Ebenso muss die Zustandskonsistenz von Instanzen berücksichtigt werden. In bestehenden Systemen wird jedoch nicht die semantische Widerspruchsfreiheit berücksichtigt. So ist nicht sichergestellt, dass die festgelegte Abfolge der Aktivitäten tatsächlich zum erwarteten Ergebnis führt, ohne unerwünschte Seiteneffekte zu erzeugen oder sogar den gesamten Prozess zu korrumpieren. Semantische Konflikte können in jedem Stadium der Lebenszeit eines Prozesses verursacht werden, angefangen bei der Modellierung des Prozesses zur Designzeit, bis hin zu Ad-hoc-Änderungen einzelner Instanzen. Die Komplexität von realen Geschäftsprozessen kann dazu führen, dass der Designer eines Geschäftsprozesses diesen nicht mehr vollständig überblickt und somit die semantische Korrektheit des Prozesses gefährdet wird. Weiter verschärft wird dieses Problem durch die Möglichkeit von Änderungen bei adaptiven PMS, insbesondere wenn Instanzänderungen von verschiedenen Personen durchgeführt werden können. Gerade in diesem Fall kann es gehäuft vorkommen, dass einzelne Personen, die Änderungen an einer Instanz durchführen, nur einen Teil des Geschäftsprozesses überblicken und damit die semantische Korrektheit der geänderten Instanz verstärkt gefährdet ist.

Ein weiteres großes Problem tritt auf, wenn eine Änderung auf der Prozessschema-Ebene stattgefunden hat, die auf alle laufenden Instanzen propagiert werden soll und geänderte, noch nicht beendete Instanzen existieren, die zu dieser Prozessschema-Änderung semantisch unverträglich sind. In diesem Szenario berücksichtigt der Prozess-Designer im schlimmsten Fall nur das unverzerrte Prozessschema und damit nicht die Änderungen, die in einzelnen Instanzen durchgeführt wurden. Unterschiedliche Lösungansätze wären für diesen Fall denkbar. Beispielsweise könnte der Prozess-Designer bei der Änderung des Prozessschemas vom System auf mögliche semantische Konflikte mit geänderten Instanzen hingewiesen werden, so dass die Entscheidung über die Behandlung dieser Instanzen beim Prozess-Designer liegt. Eine andere denkbare Lösung besteht darin, semantisch unverträgliche Instanzen nicht auf das neue Prozessschema zu migrieren, sondern die alte Version des Schemas für diese Instanzen bis zu deren Beendigung vorzuhalten.

Die Einbeziehung der Semantik stellt die nächste wichtige Herausforderung für die Weiterentwicklung der Mechanismen adaptiver PMS dar. Erst dadurch kann sichergestellt werden, dass Prozesse auf allen Ebenen konsistent und gültig sind, so dass unerwünschte Seiteneffekte ausgeschlossen werden können. Ein Durchbruch auf diesem Gebiet kann zur Entwicklung von adaptiven PMS führen, die einen noch höheren Grad an Zuverlässigkeit

bieten. Dies hat in der Folge eine positive Wirkung auf die Qualität der unterstützten Prozesse.

1.3. Ziel dieser Arbeit

Die Ziele dieser Arbeit können durch die folgende Aufstellung umrissen werden:

- Identifikation der Beschreibungselemente für die Semantik
- Identifikation der Typen semantischer Beziehungen und Konflikte
- Entwicklung einer Organisationsform für semantische Informationen
- Betrachtung der Besonderheiten von Instanz-Änderungen und Änderungen auf der Prozessschema-Ebene
- Untersuchung der effizienten Umsetzbarkeit der entwickelten Konzepte

Bevor näher auf die neuen Konzepte und Mechanismen zur Einbeziehung der Semantik eingegangen werden kann, sind in einem eigenen Kapitel die in dieser Diplomarbeit verwendeten grundlegenden Begriffe und Formalismen aus dem Bereich PMS zu klären. Dazu dient Kapitel 2, in dem die Begriffe eingeführt und definiert werden.

In Kapitel 3 wird der bereits informal eingeführte Begriff der Semantik in den Kontext PMS übertragen und formalisiert. Darauf aufbauend müssen die möglichen Konfliktsituationen herausgearbeitet werden. Deren Identifikation erfordert ein schrittweises Vorgehen, wobei ausgehend von der Bestimmung der grundlegenden Arten semantischer Beziehungen untersucht wird, welche Spezialisierungen existieren. In einem weiteren Schritt werden die verschiedenen Änderungsoperationen und ihre Auswirkungen auf die semantische Korrektheit eines Prozesses untersucht, womit das Konzept der Adaptivität aufgegriffen wird. Im Anschluss daran findet sich die Untersuchung, wie sich Verzweigungen in der Ausführung eines Prozesses auf semantische Konflikte auswirken.
Nachdem die grundlegenden Fragen geklärt sind, wird in Kapitel 4 diskutiert, wie die semantischen Informationen organisiert werden können, um die Übersicht zu wahren und maximalen Nutzen aus den Daten zu ziehen.

Damit sind die theoretischen Grundlagen geschaffen, auf denen Kapitel 5 aufsetzt. Dieses Kapitel beleuchtet die Besonderheiten, die durch die verschiedenen Ziele von Änderungsoperationen, Prozessschemata und Instanzen, berücksichtigt werden müssen. Kapitel 6 beschäftigt sich damit, wie die behandelten Konzepte effizient umgesetzt werden können. Dabei wird das Hauptaugenmerk auf die Laufzeiteffizienz gelegt. Es wird jedoch ebenso auf die Frage eingegangen, ob die entwickelten Konzepte unter dem Gesichtspunkt der Speicherbelastung sinnvoll umsetzbar sind.

Kapitel 7 dient der Diskussion verwandter Arbeiten. Den Abschluss dieser Arbeit bildet Kapitel 8 mit einer Zusammenfassung der dargestellten Konzepte und einem Ausblick

auf weitere Fragen, die im Rahmen dieser Diplomarbeit nicht erschöpfend behandelt werden konnten.

2
Grundlagen

Dieses Kapitel dient dazu, die im weiteren vorausgesetzten Begriffe und Konzepte aus dem Kontext Prozess-Management einzuführen.

2.1. Prozess-Metamodell

Den Ausführungen dieser Arbeit liegt das ADEPT Prozess-Metamodell zugrunde. Die Ausdrucksmächtigkeit des Modells erlaubt die Abbildung realer Prozesse, wobei die beschriebenen Prozessschemata übersichtlich und verständlich bleiben. Die Modellierung umfasst alle relevanten Aspekte eines Arbeitsprozesses. Darunter fallen die Modellierung von Kontroll- und Datenflüssen, Bearbeiterzuordnungen, zeitliche Abhängigkeiten oder planbare Abweichungen [RRD02]. Das ADEPT-Metamodell hat sich jedoch auch in weiteren Aspekten, wie Ad-hoc-Änderungen einzelner Prozessinstanzen oder der Schemaevolution bewährt.

2.1.1. Entwurfsaspekte - Prozessschema

Ein Arbeitsprozess wird durch graphische Modellierung des Prozessschemas festgelegt. Dieses bestimmt u.a. die Ausführungsreihenfolge und -bedingungen von Aktivitäten. Die dazu notwendigen Kontrollflüsse werden intern durch attributierte Prozessgraphen repräsentiert, die sich durch unterscheidbare Knoten- und Kantentypen auszeichnen. Definition 2.1 formalisiert aufbauend auf den in [RDB03] getroffenen Aussagen den in dieser

Ausarbeitung verwendeten Begriff des Prozessschemas. Für die grundlegenden Untersuchungen in dieser Ausarbeitung wird gegenüber [RDB03] das Metamodell vereinfacht, indem weder Loop-Kanten, noch Sync-Kanten berücksichtigt werden.

Definition 2.1 (Prozessschema)

S = (N,E,D) sei ein Prozessschema. N bezeichne die Knoten, E die Kontroll- und Datenkanten und D die Datenelemente des Prozesses. Jedem Knoten $n \in N$ wird ein Knotentyp NT(n) $\in$ {STARTFLOW, ENDFLOW, ACTIVITY, AND-SPLIT, XOR-SPLIT, AND-JOIN, XOR-JOIN}, jeder Kante $e \in E$ wird ein Kantentyp ET(e) $\in \{CONTROL\_E, DATA\_E\}$ zugewiesen.

Auf der Grundlage von Definition 2.1 können Sequenzen, parallele Verzweigungen und Alternativ-Verzweigungen modelliert werden. Für ein Prozessschema gelten verschiedene Korrektheitskriterien [Rei00, RDB03]. Die drei wichtigsten Kriterien stellen sich folgendermaßen dar:

- Kontrollblöcke dürfen geschachtelt werden, sich aber nicht überlappen.
- Für jedes Datenelement, das gelesen wird, muss eine Vorgängeraktivität existieren, die das Datenelement schreibt.
- Der Prozessgraph muss zyklenfrei in Bezug auf die Kontrollkanten sein.

2.1.2. Laufzeitaspekte - Prozessinstanzen

Das Prozessschema bildet die Basis für die Erzeugung neuer Instanzen. Instanzen benötigen Zustandsinformationen, damit der PMS-Server feststellen kann, welche Aktivitäten für eine bestimmte Instanz zur Ausführung anstehen. Deshalb wird auf Instanzebene jeder Aktivität und jeder Kante eine Zustandsmarkierung zugewiesen. Knoten und Kanten der Ablaufpfade, die nicht mehr durchlaufen werden, werden als abgewählt markiert. Eine Aktivität durchläuft während der Abarbeitung einer Instanz verschiedene Zustände. Anfangs ist eine Aktivität als NOT_ACTIVATED markiert. Wird sie aktiviert geht ihr Zustand nach ACTIVATED über. Wird eine aktivierte Aktivität gestartet, so wird ihr neuer Zustand als RUNNING repräsentiert. Nach erfolgreicher Beendigung wechselt der Zustand der Aktivität von RUNNING nach COMPLETED. Eine Aktivität, für die feststeht, dass sie in der Ausführung nicht erreicht wird, wird als SKIPPED markiert. Kontrollkanten besitzen jeweils einen von drei möglichen Zuständen. Anfangs wird ihnen der Zustand NOT_SIGNALED zugewiesen. Im Zuge der Ausführung werden die Kanten dann entweder mit FALSE_SIGNALED oder mit SIGNALED markiert. Definition 2.2 fasst diese Aussagen nochmals zusammen.

Definition 2.2 (Prozessinstanz)

Eine Instanz I wird repräsentiert durch ein Tupel (S,$\triangle$I,M) mit

- *S = (N,E,D) sei das der Instanz I zugrundeliegende Prozessschema.*

- *$\triangle I$:= (m_1,...,m_k) repräsentiert die auf der Instanz I durchgeführten Ad-hoc-Änderungen. Die Änderungen resultieren in einem instanz-spezifischem Schema S' := S + $\triangle I$ = (N',E',D')* [1]
- *M = (NS,ES) ist die Markierung der Instanz I. Jeder Aktivität n ∈ N' wird dadurch der aktuelle Zustand NS(n) ∈ {NOT_ACTIVATED, ACTIVATED, RUNNING, COMPLETED, SKIPPED} und jeder Kontrollkante e ∈ E' ihre aktuelle Markierung ES(n) ∈ {NOT_SIGNALED, SIGNALED, FALSE_SIGNALED} zugewiesen.*

Abbildung 2.1 zeigt beispielhaft ein Prozessschema und eine Prozessinstanz. Das Prozessschema beinhaltet sowohl ein Beispiel für parallele Ausführung durch einen AND-Split in A1, als auch ein Beispiel für bedingte Ausführung durch einen XOR-Split in A3. Die den jeweiligen Splits zugeordneten Joins, also A11 und A7, verdeutlichen die Blockstruktur des Metamodells. Die Betrachtung des Datenflusses zeigt, dass das dargestellte Prozessschema auch unter diesem Gesichtspunkt korrekt ist. Beide modellierten Datenelemente werden zuerst von einer Aktivität geschrieben, bevor sie gelesen werden. Die in Abbildung 2.1 dargestellte Prozessinstanz macht die Wirkungsweise von AND- und XOR-Split deutlich. Im ersten Fall werden die beiden ausgehenden Ausführungspfade beide ausgeführt, was die Markierungen der Kontrollkanten und Aktivitäten belegen. Beim XOR-Split wird im Beispiel die obere Ausführungsalternative gewählt. Die Kontrollkanten der nicht ausgewählten Alternative werden mit FALSE_SIGNALED markiert, der Aktivität A6 wird der Status SKIPPED zugewiesen.

2.2. Änderungsoperationen

Prozesse unterliegen aus verschiedenen Gründen Änderungen [AJ00]. Dies umfasst sowohl Änderungen auf der Prozesstyp-Ebene, als auch Änderungen einzelner Instanzen. Änderungen entstehen beispielsweise durch die Notwendigkeit, Aktivitäten in einen Prozess einzufügen, Aktivitäten aus einem Prozess zu entfernen oder einen Prozess durch Verschieben von Aktivitäten zu reorganisieren. Die Änderungsoperationen können wie folgt formuliert werden:

Definition 2.3 (Einfügeoperation)

S = (N,E,D) sei das Schema des Prozesses P, A_i und $A_j \in N$ seien zwei Aktivitäten von P.

$\Longrightarrow$ *Einfügeoperation* $insert(S, A_{insert}, A_i, A_j, mode)$ *mit*

- *S: Schema des Zielprozesses der Einfügeoperation*
- A_{insert}*: einzufügende Aktivität*
- A_i, A_j*: Bezugsaktivitäten für das Einfügen*

[1] Für $\triangle$I=$\emptyset$ gilt N'=N, E'=E und D'=D

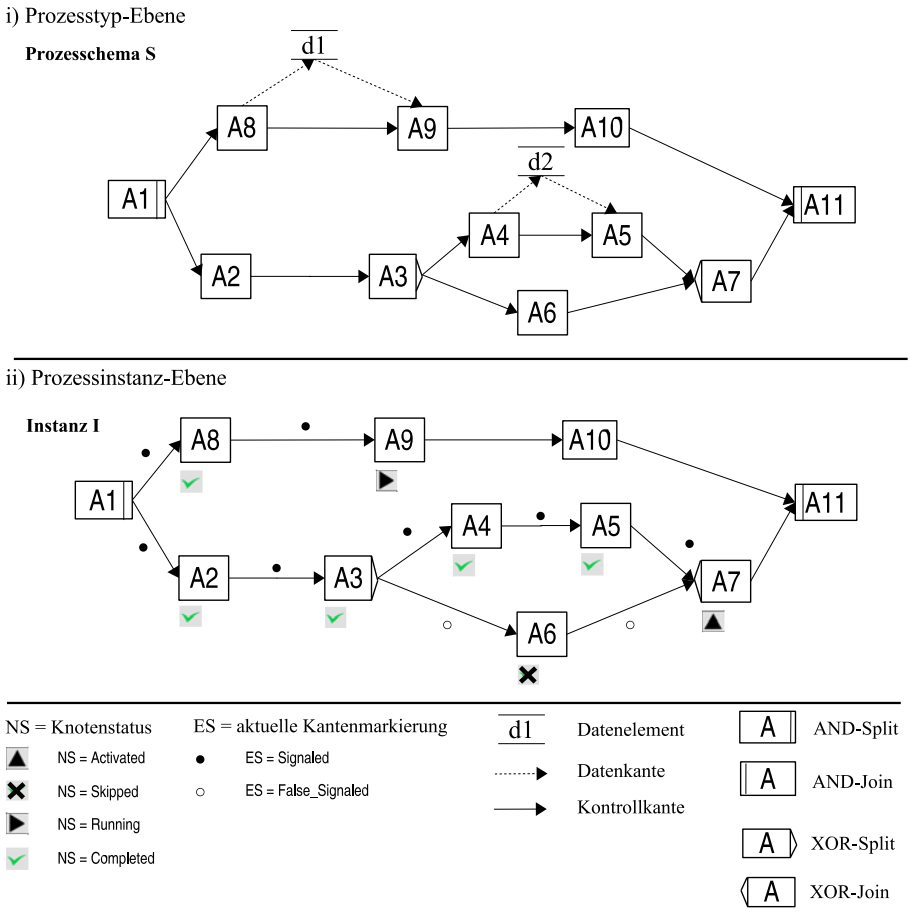


Abbildung 2.1.: Prozessschema und Instanz

- *mode* $\in$ {*seq,sel,par*}*: Parameter für das sequentielle, das parallele und das Einfügen als bedingter Ausführungsalternative*

Die Operation `insert` fügt die Aktivität A_{insert} in den Prozess ein, der durch das übergebene Schema S repräsentiert wird. Je nach übergebenem Modus wird A_{insert} sequentiell, parallel oder als bedingte Ausführungsalternative an die durch A_i und A_j spezifizierte Position eingefügt.

Definition 2.4 (Löschoperation)

S = (N,E,D) sei das Schema des Prozesses P. $A_{delete} \in N$ *sei eine Aktivität des Prozesses P.*
$\Longrightarrow$ *Löschoperation* $delete(S, A_{delete})$ *mit*

- *S: Schema des Zielprozesses der Löschoperation*
- A_{delete}*: zu löschende Aktivität*

Die Operation `delete` entfernt die Aktivität A_{delete} aus dem Prozess, der durch das übergebene Schema S repräsentiert wird. Entsteht wie in Abbildung 2.2 dadurch ein Bruch im Kontrollfluss, werden die Bezugsaktivitäten kurzgeschlossen.

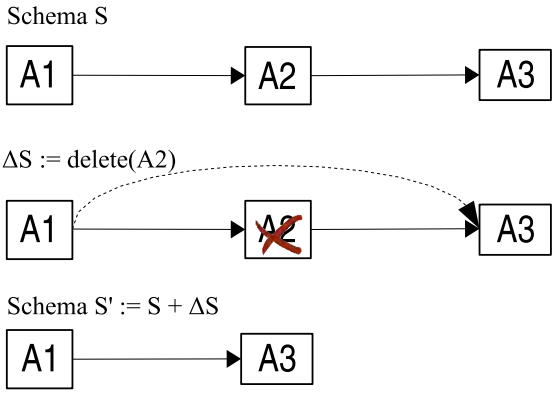


Abbildung 2.2.: Löschoperation - Bruch des Kontrollflusses

Definition 2.5 (Verschiebeoperation)

S = (N,E,D) sei das Schema des Prozesses P, $A_i, A_j, A_{move} \in N$ seien Aktivitäten von P.
$\Longrightarrow$ *Verschiebeoperation* $move(S, A_{move}, A_i, A_j, mode)$

- *S: Schema des Zielprozesses der Verschiebeoperation*
- A_{move}*: zu verschiebende Aktivität*
- A_i, A_j*: Bezugsaktivitäten für die neue Position*
- $mode \in \{seq,sel,par\}$*: Parameter für das sequentielle, das parallele und das Einfügen als bedingter Ausführungsalternative an der neuen Position*

Die Operation `move` verschiebt die Aktivität A_{move} an die durch A_i und A_j angegebene neue Position innerhalb des Prozesses, der durch das übergebene Schema S repräsentiert wird. An der alten Position werden die dortigen Bezugsaktivitäten analog zur Löschoperation gegebenenfalls kurzgeschlossen.

Änderungen können nicht beliebig durchgeführt werden, da dies die strukturelle Korrektheit eines Prozesses gefährden kann. Daher ist es im Zuge von Änderungsoperationen notwendig, deren Verträglichkeit zum Prozessschema zu prüfen. Es existieren zwei grundlegende Ansätze:

1. Änderung anwenden und anschließend eine Korrektheitsprüfung durchführen.
2. Die Änderungsoperation stellt die Korrektheit der Vor- und Nachbedingungen selbst sicher.

Im ADEPT-System wird der zweite Ansatz verfolgt. Abbildung 2.3 stellt eine Änderungsoperation dar, die die Vorbedingung einer Aktivität verletzt. Durch das Löschen der Aktivität A2 wird der für A3 benötigte Eingabeparameter nicht mehr versorgt. Die Änderung ist damit nicht verträglich.

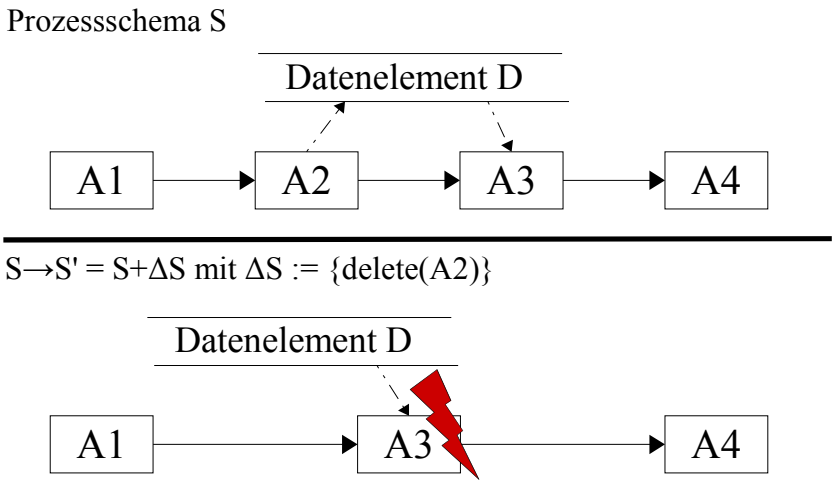


Abbildung 2.3.: Schemaänderung: verletzte Datenintegrität

2.3. Änderungen auf Instanz-Ebene

Bei der Ad-hoc-Änderung einer Instanz I können die folgenden zwei Situationen auftreten:

- Die Instanz I ist unverzerrt. $\Rightarrow \triangle I = \emptyset$
- Die Instanz I ist verzerrt. $\Rightarrow \triangle I \neq \emptyset$

Im ersten Fall kann bei der Prüfung der statischen Korrektheit der Kontroll- und Datenflüsse auf das der Instanz zugeordnete Prozessschema S zurückgegriffen werden. Dies ist für verzerrte Instanzen nicht ausreichend, da sie individuelle Änderungen gegenüber dem Prozessschema aufweisen. Deshalb muss das abweichende Schema S' der verzerrten Instanz als Grundlage der Korrektheitsprüfung dienen.

Für beide Fälle gilt, dass zusätzlich zur Überprüfung der statischen Korrektheit die aktuelle Markierung der Prozessinstanz auf Verträglichkeit mit der Änderung geprüft wird. Dieser Sachverhalt wird durch ein Beispiel in Abbildung 2.4 verdeutlicht. Das Beispiel zeigt eine Instanz, die bereits weit fortgeschritten ist. Durch eine Instanzänderung soll zwischen A1 und A2 die Aktivität A5 eingefügt werden. Dies resultiert darin, dass A5 in der Ausführung nie erreicht werden kann. Damit ergibt sich nach Beendigung der Instanz ein Zustand, in dem eine Aktivität existiert, die nicht erreicht wurde. Dies stellt

einen inkonsistenten Zustand dar. In [Rin04, RD98, RRD04] werden Mechanismen zur Sicherstellung der Korrektheit auf der strukturellen und Zustands-Ebene beschrieben.

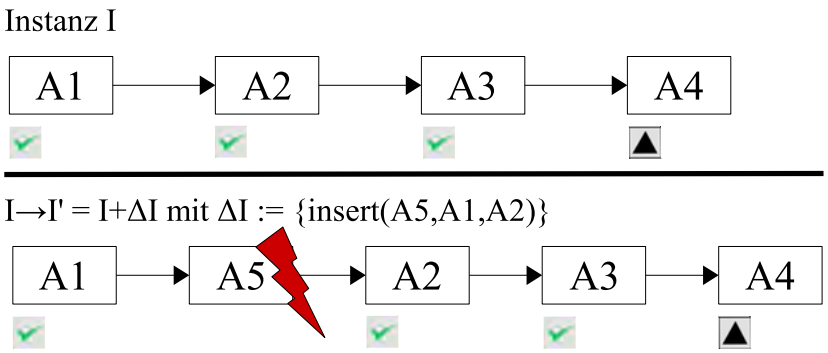


Abbildung 2.4.: Instanzänderung: inkonsistente Markierung

2.4. Änderungen auf Prozesstyp-Ebene

Bei der Änderung eines Prozessschemas muss die Konsistenz der Kontroll- und Datenflüsse geprüft werden. Ist die Änderung verträglich zum Prozessschema, so wird eine neue Version des Prozessschemas angelegt, die die Änderung widerspiegelt.
Existieren keine Instanzen des Schemas, die momentan ausgeführt werden, so müssen keine weiteren Überprüfungen vorgenommen werden. Neu anzulegende Instanzen werden dem geänderten Schema zugeordnet.
Existieren hingegen bereits gestartete Instanzen des Schemas, so müssen diese auf Verträglichkeit mit der Änderung des Schemas und damit auf Migrierbarkeit auf das neue Schema geprüft werden. Dabei müssen für jede Instanz folgende Situationen unterschieden werden:

- Die Instanz I ist unverzerrt.
- Die Instanz I ist verzerrt.

Im ersten Fall ist die Instanz I aus der statischen Sicht verträglich zur Änderung des Prozessschemas, da die Instanz keine Änderungen gegenüber dem Prozessschema enthält. Dies stellt sich im zweiten Fall anders dar, da für die Instanz $\triangle I \neq \emptyset$ gilt. Die Änderungen der Instanz müssen beim Korrektheitstest berücksichtigt werden.

Zusätzlich muss in beiden Fällen die aktuelle Markierungssituation der Instanzen miteinbezogen werden, um eine Entscheidung über die tatsächliche Migrierbarkeit der Instanzen treffen zu können.

Diese Überprüfungen der Migrierbarkeit einer Instanz kann zu folgenden Ergebnissen führen:

- Die Instanz ist verträglich zur Änderung des Prozessschemas.
- Die Instanz ist unverträglich zur Änderung des Prozessschemas.

Im ersten Fall kann die Instanz auf das geänderte Schema migriert werden. Im zweiten Fall ist dies nicht möglich. Die Instanz bleibt weiter der alten Version des Schemas zugeordnet, das mindestens bis zur Beendigung der letzten zugeordneten Instanz vorgehalten werden muss.

3

Semantik im Kontext Prozess-Management

Prozesse, die in ein Prozess-Management-System abgebildet werden, können sehr komplex sein. Je größer und verzweigter sich ein Prozess darstellt, desto höher ist das Risiko, dass die Übersichtlichkeit verloren geht. Es besteht insbesondere bei Änderungen die Gefahr, dass Inkonsistenzen im Prozess verursacht werden. Deshalb müssen Konsistenztests durchgeführt werden. Existierende Konsistenztests berücksichtigen:

- die syntaktische Korrektheit
- die Zustandskonsistenz von Instanzen

Die nächste große Herausforderung besteht darin, mit der Semantik eine neue Ebene für die Vergabe von Korrektheitsgarantien zu entwickeln. Abbildung 1.1 verdeutlicht am Beispiel einer Medikamentenunverträglichkeit die Notwendigkeit der Einbeziehung semantischer Aspekte. Die alleinige applikationsneutrale Betrachtung kann zu gesundheitlichen Schäden beim Patient führen. Über die Prüfung der syntaktischen Korrektheit hinaus müssen die Prozesse bzw. die Instanzen anhand der in den Prozess eingebrachten semantischen Informationen nach festgelegten Korrektheitskriterien bewertet werden. Die Herausforderung besteht darin, die mit einem Prozess assoziierten semantischen Informationen deutlich von der reinen Beschreibung der Struktur und der Datenflüsse des Prozesses abzuheben. Die Grundlage der Semantik muss sich vielmehr an den real ausgeführten Arbeitsvorgängen der Prozesse orientieren.

Im Mittelpunkt dieses Kapitels steht die Entwicklung eines formalen Rahmenwerks für die Integration semantischer Aspekte. Dazu werden folgende Untersuchungen angestellt:

- Beschreibung der Semantik von Aktivitäten

- Identifikation der existierenden semantischen Beziehungen
- Identifikation der Arten semantischer Konflikte
- Auswirkungen von Änderungsoperationen auf die Semantik

Die Untersuchung der genannten Punkte legt die Grundlage, um Prozesse mit semantischer Information zu erweitern und die semantische Korrektheit auswerten zu können.

3.1. Semantik von Aktivitäten

Ein Prozess besteht aus einzelnen Prozessschritten. Die einzelnen Prozessschritte in ihrer vorgegebenen Reihenfolge stellen zusammen die Gesamtwirkung des Prozesses dar, d.h. die tatsächliche Manipulation von Daten und Objekten, die im Kontext des Prozesses bearbeitet werden. Diese Gesamtwirkung der Prozessschritte kann als die Semantik des vollständigen Prozesses betrachtet werden. Bei dieser Betrachtungsweise wird deutlich, dass sich die Semantik eines Prozesses mit jeder Anpassung des Prozesses ändert, d.h. bei Änderungen an der Prozessstruktur, aber auch bei Änderungen innerhalb einzelner Prozessschritte. Dabei kann es zu Änderungen kommen, die unerwünschte Effekte auf die Semantik des Prozesses haben. Deutlich wird dies an folgendem Beispiel:
Im Bankwesen gibt es einen Prozess, der den Ablauf beim Abheben eines Betrages von einem Kundenkonto beschreibt. Die Semantik dieses Prozesses kann auf einer abstrakten Ebene folgendermaßen beschrieben werden:

- Der Kunde erhält seinen gewünschten Betrag in bar.
- Das Konto wird entsprechend dem abgehobenen Betrag belastet.

In den Prozess wird nun die Änderung eingebracht, dass der Betrag sofort nach dem Belasten des Kontos wieder gutgeschrieben wird. Die realen Auswirkungen des Prozesses stellen sich dann wie folgt dar.

- Der Kunde erhält seinen gewünschten Betrag in bar.
- Das Konto wird entsprechend dem abgehobenen Betrag belastet.
- Auf dem Konto wird der abgehobene Betrag gutgeschrieben.

Es ist offensichtlich, dass diese Änderung ein von der Bank nicht erwünschtes Verhalten im Prozess verursacht und die Semantik des Prozesses durch die Änderung ad absurdum geführt wird.
Es liegt nahe, dass die einzelnen Prozessschritte bzw. Aktivitäten der Gegenstand der semantischen Beschreibung sind, damit insbesondere bei Änderungen festgestellt werden kann, welche Auswirkungen die Aktivitäten auf die Semantik eines Prozesses haben. Im Folgenden muss die Frage beantwortet werden, wie eine Aktivität semantisch beschrieben werden kann. Wie bereits in Definition 1.1 eingeführt, bezieht sich die Semantik

einer Aktivität auf die innerhalb der Aktivität manipulierten bzw. benutzten materiellen oder immateriellen Objekte. Im normalen Sprachgebrauch versteht man unter einem Objekt einen materiellen, klar abgegrenzten Gegenstand. Das Einbeziehen von immateriellen Objekten erweitert dabei die Vorstellung des Objektbegriffes insofern, dass auch Gegenstände, die nicht physisch vorhanden sind, trotzdem jedoch das Konzept des Objekts widerspiegeln, unter diesen Begriff fallen. Ein Vertreter dieser Art ist beispielsweise ein elektronisches Dokument. Für die Beschreibung von Aktivitäten muss dieser Begriff abermals erweitert und abstrahiert werden. Das Objekt einer Aktivität muss sich nicht unbedingt auf einen klar abgegrenzten Gegenstand beziehen, sondern stellt in Anlehnung an die Grammatik der Sprache das Akkusativ-Objekt eines Satzes dar. Dieser Zusammenhang kann am Beispiel des Satzes "Der Arzt führt die Operation durch" verdeutlicht werden. "Operation" repräsentiert das Akkusativ-Objekt dieses Satzes. Einer Aktivität `Operation durchführen`, die diesen Satz in einen Prozessschritt überträgt, kann dann das Objekt `Operation` zugeordnet werden, wodurch die Semantik der Aktivität als Operation festgelegt wird. Es existieren also auch Objekte, die nicht innerhalb einer Aktivität manipuliert werden, sondern die eine Umschreibung der Tätigkeit der Aktivität darstellen.

Die Objekte bilden die Basis, um die Semantik einer Aktivität zu beschreiben. In späteren Kapiteln wird näher untersucht, was für Elemente benötigt werden, um detailliertere Möglichkeiten zur Beschreibung der Semantik einer Aktivität anzubieten. Beispielsweise könnte es in einer Aktivität `Operation durchführen` relevant sein, von wem die Operation durchgeführt werden soll, da verschiedene Ärzte eventuell nur bestimmte Eingriffe vornehmen dürfen. So wäre es vermutlich nicht ratsam, einem Zahnarzt einen chirurgischen Eingriff in die inneren Organe zu erlauben. Bevor jedoch näher auf diese zusätzlichen beschreibenden Komponenten eingegangen wird, muss untersucht werden, welche weiteren Mechanismen benötigt werden, um die für einen Korrektheitstest notwendigen Informationen zu erhalten bzw. diese auszuwerten.

3.2. Semantische Beziehungen

Im vorigen Kapitel wurde die grundlegende semantische Beschreibung von Aktivitäten eingeführt. Dies beinhaltet einen wichtigen Schritt auf dem Weg zu einem semantischen Korrektheitstest. Es fehlen jedoch die Mechanismen zur Auswertung dieser Informationen, um zu entscheiden, was semantisch korrekt ist und was zu einer semantischen Inkonsistenz führt. Zwischen verschiedenen Aktivitäten eines Prozesses können unterschiedliche semantische Beziehungen bestehen. Im bereits eingeführten Beispiel, das den Vorgang beim Abheben eines Geldbetrages von einem Konto beschreibt, besteht eine Beziehung zwischen der Aktivität `Auszahlung an den Kunden` und `Gutschrift buchen`. Diese Beziehung besagt, dass die zweite Aktivität nicht im Zusammenhang mit der ersten Aktivität ausgeführt werden darf. Aus diesem Beispiel kann abgeleitet werden, dass eine solche Beziehung in Form einer Regel ausgedrückt werden kann, z.B. folgendermaßen:

- `Auszahlung an Kunden` schließt aus `Gutschrift buchen`

Eine solche Regel müsste nun dem Prozess "Abhebevorgang" zugeordnet werden, so dass bei einer Änderungsoperation, die zum Ziel hat, die Aktivität `Gutschrift buchen` nach der Aktivität `Auszahlung an den Kunden` in den Prozess einzufügen, festgestellt werden kann, dass diese Änderungsoperation gegen eine Regel verstößt und damit zu einem semantischen Konflikt führt. Die mit dem Prozess assoziierte Regel setzt also zwei Aktivitäten direkt miteinander in Beziehung. Es stellt sich die Frage, wo dabei die im vorigen Kapitel eingeführte semantische Beschreibung von Aktivitäten ins Spiel kommt. Solche Regeln stellen offensichtlich ebenso eine Art der semantischen Beschreibung einer Aktivität dar. Die Regeln, die die Beziehungen zwischen den Aktivitäten herstellen, sind auf das Prozessumfeld gerichtet, in das die Aktivität eingebettet ist, wohingegen die Beschreibung einer Aktivität durch Zuordnung von Objekten auf die Aktivität selbst gerichtet ist.
Aktivitäten als Ganzes wurden in obigem Beispiel als Gegenstand der semantischen Beziehungen betrachtet. Dies bringt den Vorteil mit sich, dass der Wirkungsbereich der Regeln klar abgegrenzt und schnell zu überblicken ist, nämlich die an der Regel beteiligten Aktivitäten. Solche Regeln sind jedoch nicht für andere Aktivitäten wiederverwendbar. Eine Regel, die beispielsweise eine Aktivität `Gutschrift buchen` vom Prozess ausschließen soll, muss für verschiedene Quellaktivitäten, für die dieser Sachverhalt zutrifft, jedesmal neu definiert werden. In einem Prozess "Bearbeitung einer Überweisung" gilt der selbe Sachverhalt wie beim Abheben eines Betrages. Das Konto des Kunden wird mit dem überwiesenen Betrag belastet und dieser Betrag darf nach der Buchung nicht wieder auf dem Konto des überweisenden Kunden gutgeschrieben werden. Für den Überweisungsprozess muss dieser Sachverhalt erneut in einer eigenen Regel ausgedrückt werden:

- `Überweisung buchen` schließt aus `Gutschrift buchen`

Betrachtet man beispielhaft einen möglichen Ausschnitt der Domäne des Bankwesens, so könnten die folgenden Regeln innerhalb der Domäne existieren:

- `Auszahlung an Kunden` schließt aus `Gutschrift buchen`
- `Überweisung buchen` schließt aus `Gutschrift buchen`

Mehrere verschiedene Regeln mit dem selben Ziel, d.h. die Aktivität `Gutschrift buchen` vom Prozess auszuschließen, sind in der Domäne definiert. Es kann in einer Domäne eine große Menge an Regeln geben, die jeweils eine unterschiedliche Quellaktivität, jedoch immer die selbe Zielaktivität besitzen. Man stelle sich vor, dass ein durch solche Regeln ausgedrückter, für eine Domäne global gültiger Sachverhalt, geändert werden muss, da sich beispielsweise gesetzliche Bestimmungen oder die Geschäftspolitik des Unternehmens ändern. Dann müssen unter Umständen sehr viele Regeln angepasst werden. Dies bedeutet einen hohen Aufwand und bietet Potential für Fehler. So ist es durchaus denkbar, dass bei der Anpassung eine Regel übersehen wird und somit eine Inkonsistenz im Regelwerk der Domäne entsteht.
Ein zweiter Fallstrick tritt bei der Definition neuer Aktivitäten in den Vordergrund.

Sämtliche bereits definierten Aktivitäten müssen in die Entscheidung miteinbezogen werden, welche semantischen Beziehungen für die neue Aktivität bestehen.

Diese Probleme haben ihre Ursache in der Betrachtung der Aktivitäten als Gegenstand der semantischen Beziehungen. Es ist viel natürlicher und übersichtlicher, eine Aktivität als Menge ihrer Objekte zu betrachten und ihre semantischen Beziehungen an diesen Objekten auszurichten. Die Beschreibung der Semantik einer Aktivität bewegt sich damit auf zwei Ebenen:

- Beschreibung der Semantik einer Aktivität über die Objekte, die die Aktivität ausmachen.
- Implizite Beschreibung der Beziehungen zwischen Aktivitäten durch Beziehungen, die auf der Objektebene definiert werden.

Damit kann die Definition der zu verwendenden Objekte und die Festlegung der Beziehungen zwischen verschiedenen Objekten unabhängig von der Definition der Aktivitäten vorgenommen werden. Eine einmal definierte Beziehung, die als Regel zwischen zwei Objekten repräsentiert wird, ist durch die Zuordnung von Objekten zu Aktivitäten in mehreren Aktivitäten verwendbar und folgt einem zentralen Ansatz. Obiges Beispiel aus dem Bankenwesen kann auf die geänderten Gegebenheiten übertragen werden. Abbildung 3.1 stellt die Assoziationen zwischen Aktivitäten-, Objekt- und Regelmenge graphisch dar.

In einem Prozess existiert die Aktivität `Auszahlung an Kunden`. Der Prozess soll durch Einfügen der Aktivität `Gutschrift buchen` direkt nach der Auszahlung erweitert werden. Durch die Aktivität `Auszahlung an Kunden` und ihre Objektzuordnungen befindet sich auch das Objekt `Belastung` mit der zugehörigen Regel `Belastung schließt aus Gutschrift` im Prozess. Durch das Einfügen der Aktivität `Gutschrift buchen` würde das assoziierte Objekt `Gutschrift` in den Prozess aufgenommen. Die bereits im Prozess vorhandene Regel `Belastung schließt aus Gutschrift` steht jedoch im Konflikt zu dieser Einfügung, so dass das Einfügen der Aktivität nicht möglich ist. Es wird also durch die Regel verhindert, dass eine Aktivität in den Prozess eingefügt werden kann, die die Semantik des gesamten Prozesses gefährdet. Tauscht man in diesem Prozess die Aktivität `Auszahlung an Kunden` mit der Aktivität `Überweisung buchen` aus und versucht, diesen Prozess ebenfalls mit der Aktivität `Gutschrift buchen` zu erweitern, so wird die Änderung aus den selben Gründen wiederum zurückgewiesen.
Ändert sich ein globaler Sachverhalt, d.h. die Beziehung zwischen zwei Objekten ändert sich für eine ganze Domäne, so müssen die entsprechenden Regeln nur an einer Stelle angepasst werden. Die Regeln werden dann implizit auf die Aktivitäten propagiert, die mit ihren Quellobjekten assoziiert sind. Durch diesen Mechanismus kann vermieden werden, dass bei der Anpassung vieler Regeln Fehler bei einzelnen Regeln gemacht werden, oder einige Regeln bei der Änderung übersehen werden.

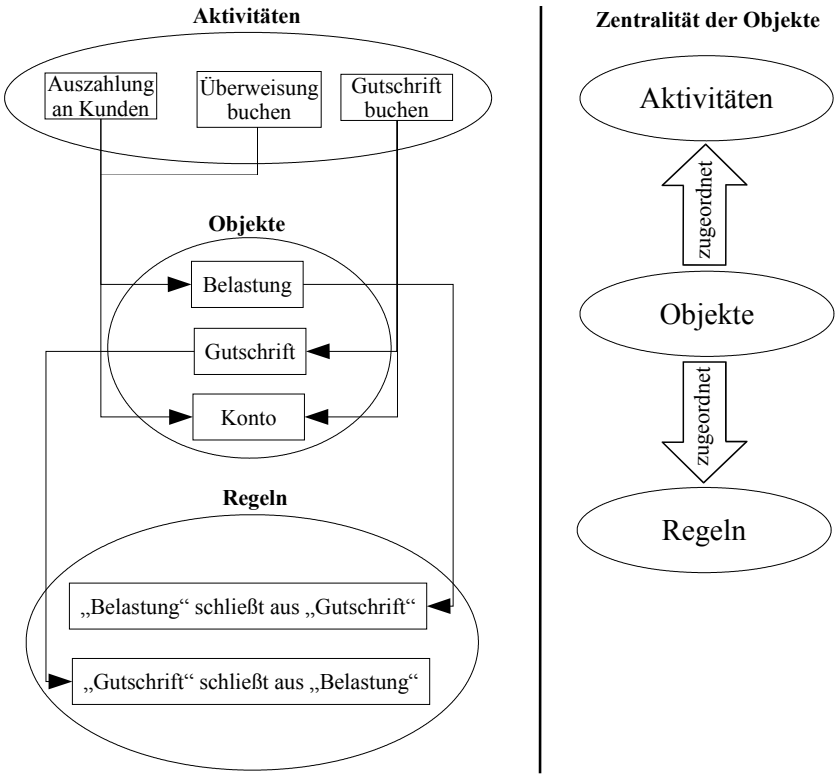


Abbildung 3.1.: Zusammenhang: Aktivitäten, Objekte und Regeln

3.2.1. Typen semantischer Beziehungen

Das Grundkonzept zur Sicherstellung der semantischen Korrektheit durch die Beschreibung von Beziehungen zwischen Objekten setzt die Identifizierung der existierenden Arten semantischer Beziehungen voraus. Es gibt drei grundlegende Typen von Beziehungen

- Zwei Objekte dürfen nicht gemeinsam im selben Anwendungskontext vorkommen. Diesem Sachverhalt wird durch eine Ausschlussbeziehung zwischen den beteiligten Objekten Rechnung getragen. Im weiteren Verlauf dieses Dokuments wird von einer Ausschlussbeziehung auch als einer `excludes-`Beziehung gesprochen. Einfache Ausschlussbeziehungen der Art `Object1 schließt aus Object2` sind symmetrische Beziehungen, d.h. eine solche Regel impliziert ebenso die Umkehrregel, nämlich `Object2 schließt aus Object1`. Diese Eigenschaft ergibt sich aus der Überlegung, dass die erste Regel festlegt, dass `Object2` nur im Prozess vorhanden sein darf, wenn nicht gleichzeitig `Object1` im Prozess vorkommt. Dies impliziert, dass, wenn immer `Object2` im Prozess vorhanden ist, kein Vorkommen von `Object1` im Prozess sein kann, woraus sich die Umkehrregel ergibt.

- Ein Objekt darf nur dann in einem Anwendungskontext verwendet werden, wenn auch ein bestimmtes anderes Objekt im selben Anwendungskontext auftritt. In diesem Fall kann von einer Abhängigkeitsbeziehung gesprochen werden, die nicht unbedingt auf Gegenseitigkeit beruhen muss. Diese Beziehung erfüllt damit nicht das Kriterium der Symmetrie, was bedeutet, dass das Zielobjekt, welches die geforderte Abhängigkeit auflöst, nicht auch vom Quellobjekt abhängig sein muss. Ähnlich wie bei der Ausschlussbeziehung wird im weiteren teils von einer `dependsOn`-Beziehung anstelle von einer Abhängigkeitsbeziehung gesprochen, was als synonym zu betrachten ist.
- Abgesehen vom Ausschluss und der Abhängigkeit gibt es den dritten Fall, dass zwischen zwei Objekten keinerlei Beziehung besteht, die die Benutzung eines der beiden Objekte mit dem anderen verbietet oder eine Abhängigkeit zwischen den Objekten fordert. Dies stellt den trivialen Fall dar und kann als das Komplement der beiden erstgenannten Fälle, dem Ausschluss und der Abhängigkeit, betrachtet werden. Dieser Fall kann als Nullbeziehung bezeichnet werden.

Interessant für die Erfassung in Prozessen sind nur die Ausschlussbeziehungen sowie die Abhängigkeitsbeziehungen zwischen Objekten, da es hier zu semantischen Konflikten kommen kann, sei es durch eine verletzte Ausschlussbedingung, sei es durch eine nicht erfüllte Abhängigkeit. Im Gegensatz dazu haben die Nullbeziehungen keine Relevanz für die Korrektheit von Prozessen, da die Natur dieser Beziehungsart vorgibt, dass keine semantischen Wechselwirkungen zu anderen Objekten existieren und deshalb keinerlei semantische Konflikte durch Nullbeziehungen im Prozess verursacht werden können.

3.2.2. Semantische Richtungsbeziehung

Es ist wünschenswert auch komplexere Sachverhalte als generellen Ausschluss und allgemeine Abhängigkeit ausdrücken zu können. Abbildung 3.3 liefert einen Hinweis auf eine spezifischere semantische Beziehung. Die Aktivität `Operation durchführen` verursacht die Abhängigkeit im Prozess, dass dem Patient ein Anästhetikum gegeben werden muss. Es ist offensichtlich, dass das Anästhetikum verabreicht werden muss, bevor die Operation durchgeführt wird. Die mit dem Objekt `Operation` assoziierte Regel `Operation dependsOn Anästhetikum` besitzt jedoch keine Information, wo im Prozess das Objekt `Anästhetikum` angesiedelt sein muss. Es würde kein semantischer Konflikt durch eine unerfüllte Abhängigkeit erkannt, solange das Objekt `Anästhetikum` überhaupt im Prozess vorkommt, also auch in einer Aktivität, die erst nach der durchgeführten Operation im Prozess ausgeführt wird. Dieses Verhalten ist nicht akzeptabel.
Das Beispiel hat die benötigte Zusatzinformation der relativen Position des Zielobjekts einer Regel zu ihrem Quellobjekt aufgedeckt, im Folgenden auch bezeichnet als `directionTarget`. Dies bedeutet, dass in der Regel vorgegeben werden kann, ob das Zielobjekt einer Aktivität zugeordnet sein muss, die vor oder nach der Aktivität des Quellobjekts ausgeführt wird, damit die Regel erfüllt wird.

Objekte und semantische Beziehungen bilden die Grundlage für die semantische Beschreibung einer Domäne, in der Prozesse festgelegt werden können. Dieser Begriff wird in Definition 3.1 formal eingeführt.

Definition 3.1 (Domäne - Semantik)

Eine Anwendungsdomäne $\mathcal{D}$ wird durch ein Tupel ($\mathcal{N}$,$\mathcal{O}$, $\mathcal{R}$, SEM_ACT, SEM_RELS) semantisch repräsentiert:

- *$\mathcal{N}$ ist die Menge aller Aktivitäten in $\mathcal{D}$*
- *$\mathcal{O}$ ist die Menge der Objekte, die $\mathcal{D}$ semantisch beschreiben*
- *$\mathcal{R}$:= {SR(s,t) | s,t $\in \mathcal{O}$} ist die Menge der semantischen Beziehungen in $\mathcal{D}$, mit SR(s,t) := ruleType(s,t [, directionTarget]) mit*
  - *ruleType $\in$ {excludes,dependsOn}*
  - *s ist Quellobjekt der Semantikregel*
  - *t ist Zielobjekt der Semantikregel*
  - *directionTarget $\in$ {pre,post} gibt die relative Position des Zielobjekts an*
- *SEM_ACT: $\mathcal{N} \rightarrow \mathcal{P}(\mathcal{O})$ stellt die semantische Aktivitätenbeschreibung dar*
- *SEM_RELS: $\mathcal{O} \rightarrow \mathcal{P}(\mathcal{R})$ ordnet den Objekten die semantischen Beziehungen zu*

Mit Definition 3.1 sind die Voraussetzungen geschaffen, um das in Kapitel 2 eingeführte Prozessmetamodell zu erweitern.

Definition 3.2 (Prozessschema - Zuordnung einer Domäne)

S = (N,E,D,Dom) sei ein Prozessschema in der Domäne Dom = ($\mathcal{N}$, $\mathcal{O}$, $\mathcal{R}$,) mit

- *Dom ordnet das Schema einer Domäne zu*
- *N$\subseteq \mathcal{N}$ ist die Menge der dem Schema zugeordneten Aktivitäten der Domäne*

3.3. Semantische Konflikte

Die Festlegung von Regeln zwischen Objekten, die den in Definition 3.1 eingeführten Beziehungen entsprechen, hat das Ziel, semantische Konflikte und Inkonsistenzen innerhalb von Prozessen zu vermeiden. Um einen semantischen Konflikt bei einem Korrektheitstest erkennen zu können, muss festgehalten werden, wann genau von einem semantischen Konflikt gesprochen wird. Definition 3.3 stellt formal die existierenden semantischen Konfliktsituationen dar, wobei mit $c\_pred^*(S, A)$ und $c\_succ^*(S, A)$ Vorgänger- bzw. Nachfolger-Funktionen Verwendung finden, die im Anhang in Definition A.1 eingeführt werden.

Definition 3.3 (Semantische Konflikte)

Ein semantischer Konflikt ist dadurch gekennzeichnet, dass die Bedingung einer Regel eines Objektes verletzt wird. Formal kann dies folgendermaßen ausgedrückt werden:
i) semantischer Konflikt durch verletzte Ausschlussbedingung:
S = (N,E,D,Dom) sei ein gegebenes Prozessschema in der Domäne
Dom = ($\mathcal{N}$*,*$\mathcal{O}, \mathcal{R}, SEM\_ACT, SEM\_RELS$*). Weiter gelte* $O_m, O_n \in \mathcal{O}$.
Ein semantischer Konflikt durch eine verletzte Ausschlussbedingung, bezeichnet als brokenExclusion(O_m, O_n)*, ist wie folgt charakterisiert:*
a.) ohne Angabe von directionTarget: brokenExclusion$(O_m, O_n) \Longleftrightarrow$

$$(\exists\ A_i \in N \text{ mit } O_m \in SEM\_ACT(A_i))$$
$$\wedge\ (\exists\ r \in SEM\_RELS(O_m) \text{ mit } r = excludes(O_m, O_n))$$
$$\wedge\ (\exists\ A_j \in N \text{ mit } O_n \in SEM\_ACT(A_j))$$

b.) mit directionTarget = pre: brokenExclusion$(O_m, O_n) \Longleftrightarrow$

$$(\exists\ A_i \in N \text{ mit } O_m \in SEM\_ACT(A_i))$$
$$\wedge\ (\exists\ r \in SEM\_RELS(O_m) \text{ mit } r = excludes(O_m, O_n, pre))$$
$$\wedge\ (\exists\ A_j \in c\_pred^*(S, A_i) \text{ mit } O_n \in SEM\_ACT(A_j))$$

c.) mit directionTarget = post: brokenExclusion$(O_m, O_n) \Longleftrightarrow$

$$(\exists\ A_i \in N \text{ mit } O_m \in SEM\_ACT(A_i))$$
$$\wedge\ (\exists\ r \in SEM\_RELS(O_m) \text{ mit } r = excludes(O_m, O_n, post))$$
$$\wedge\ (\exists\ A_j \in c\_succ^*(S, A_i) \text{ mit } O_n \in SEM\_ACT(A_j))$$

ii) semantischer Konflikt durch verletzte Abhängigkeitsbedingung:
S = (N,E,D,Dom) sei ein gegebenes Prozessschema in der Domäne
Dom = ($\mathcal{N}$*,*$\mathcal{O}, \mathcal{R}, SEM\_ACT, SEM\_RELS$*). Weiter gilt* $O_m, O_n \in \mathcal{O}$.
Ein semantischer Konflikt durch eine nicht erfüllte Abhängigkeit , bezeichnet als brokenDependency(O_m, O_n)*, ist wie folgt charakterisiert:*
a.) ohne Angabe von directionTarget: brokenDependency$(O_m, O_n) \Longleftrightarrow$

$$(\exists\ A_i \in N \text{ mit } O_m \in SEM\_ACT(A_i))$$
$$\wedge\ (\exists\ r \in SEM\_RELS(O_m) \text{ mit } r = dependsOn(O_m, O_n))$$
$$\wedge\ (\nexists\ A_j \in N \text{ mit } O_n \in SEM\_ACT(A_j))$$

b.) mit directionTarget = pre: brokenDependency$(O_m, O_n) \Longleftrightarrow$

$$(\exists\ A_i \in N \text{ mit } O_m \in SEM\_ACT(A_i))$$
$$\wedge\ (\exists\ r \in SEM\_RELS(O_m) \text{ mit } r = dependsOn(O_m, O_n, pre))$$
$$\wedge\ (\nexists\ A_j \in c\_pred^*(S, A_i) \text{ mit } O_n \in SEM\_ACT(A_j))$$

c.) mit directionTarget = post: brokenDependency$(O_m, O_n) \Longleftrightarrow$

$$(\exists\ A_i \in N \text{ mit } O_m \in SEM\_ACT(A_i))$$
$$\wedge\ (\exists\ r \in SEM\_RELS(O_m) \text{ mit } r = dependsOn(O_m, O_n, post))$$
$$\wedge\ (\nexists\ A_j \in c\_succ^*(S, A_i) \text{ mit } O_n \in SEM\_ACT(A_j))$$

Die in Definition 3.3 eingeführten semantischen Konflikte sollen an zwei Beispielen verdeutlicht werden [1]. Das erste Beispiel, dargestellt in Abbildung 3.2, zeigt dabei das Auftreten eines semantischen Konflikts infolge einer verletzten Ausschlussbeziehung. Der semantische Konflikt tritt zwischen den Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen` auf. Die Aktivität `Aspirin verabreichen` wird über das ihr zugeordnete Objekt `Aspirin` semantisch beschrieben, die Aktivität `Marcumar verabreichen` hingegen wird mit dem Objekt `Marcumar` assoziiert. Für diese beiden Objekte sind wechselseitige Ausschlussbeziehungen festgelegt. Die Unverträglichkeit hat zur Folge, dass die zwei Objekte nicht zusammen im Prozess vorkommen dürfen und deshalb die beiden Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen` nicht gleichzeitig im Prozess verwendet werden können. Da dies dennoch der Fall ist kommt es zu einem semantischen Konflikt.

Das zweite Beispiel aus Abbildung 3.3 veranschaulicht, wie ein semantischer Konflikt durch eine nicht erfüllte Abhängigkeit in einem Prozess entstehen kann. Den Ausgangspunkt des Konfliktes bildet die Aktivität `Operation durchführen`. Die semantische Beschreibung dieser Aktivität beinhaltet das Objekt `Operation`. Für `Operation` existiert eine Regel, die eine Abhängigkeitsbeziehung zum Objekt `Anästhetikum` modelliert. Das bedeutet, dass ein Patient nur dann operiert werden darf, wenn er zuvor mittels eines Anästhetikums betäubt wurde. Die einzige in der zugrundeliegenden Domäne definierte Aktivität, in deren semantischer Beschreibung das Objekt `Anästhetikum` vorkommt, ist die Aktivität `Anästhetikum verabreichen`. Diese Aktivität ist jedoch nicht Bestandteil des Prozesses, so dass die geforderte Abhängigkeit nicht erfüllt wird und somit der semantische Konflikt durch eine nicht aufgelöste Abhängigkeit besteht. Dieser Konflikt kann mit Hilfe von Definition 3.3 formal hergeleitet werden:

S = (N,E,D,Dom) sei das Schema zum Prozess "Behandlungsprozess" in der Domäne Dom = ($\mathcal{N}$,$\mathcal{O}$,$\mathcal{R}$, SEM_ACT, SEM_RELS). A_i entspricht der Aktivität `Operation durchführen`, $O_m \in SEM\_ACT(A_i)$ sei das Objekt `Operation`.
$SEM\_RELS(O_m)$:= {dependsOn(Operation, Anästhetikum)}
$\Longrightarrow \exists$ r $\in SEM\_RELS(O_m)$ mit
r=dependsOn(O_m,O_n)=dependsOn(Operation,Anästhetikum)
Aber: $\nexists A_j \in$ N mit `Anästhetikum` $\in SEM\_ACT(A_j)$
$\Longrightarrow$ Es existiert ein semantischer Konflikt `brokenDependency` durch die nicht aufgelöste Abhängigkeit in der Aktivität `Operation durchführen`.

Die regelbasierten semantischen Beziehungen bilden den Ausgangspunkt für die Sicherstellung der semantischen Korrektheit eines Prozesses. Es bleibt jedoch zu untersuchen, inwiefern verschiedene Ausführungspfade von Prozessen das tatsächliche Auftreten eines semantischen Konflikts beeinflussen. Ebenso muss untersucht werden, welche semantischen Konflikte durch die verschiedenen Änderungsoperationen verursacht werden können.

[1] In den folgenden Betrachtungen wird vereinfachend von einer Unverträglichkeit zwischen den Medikamenten Aspirin und Marcumar ausgegangen.

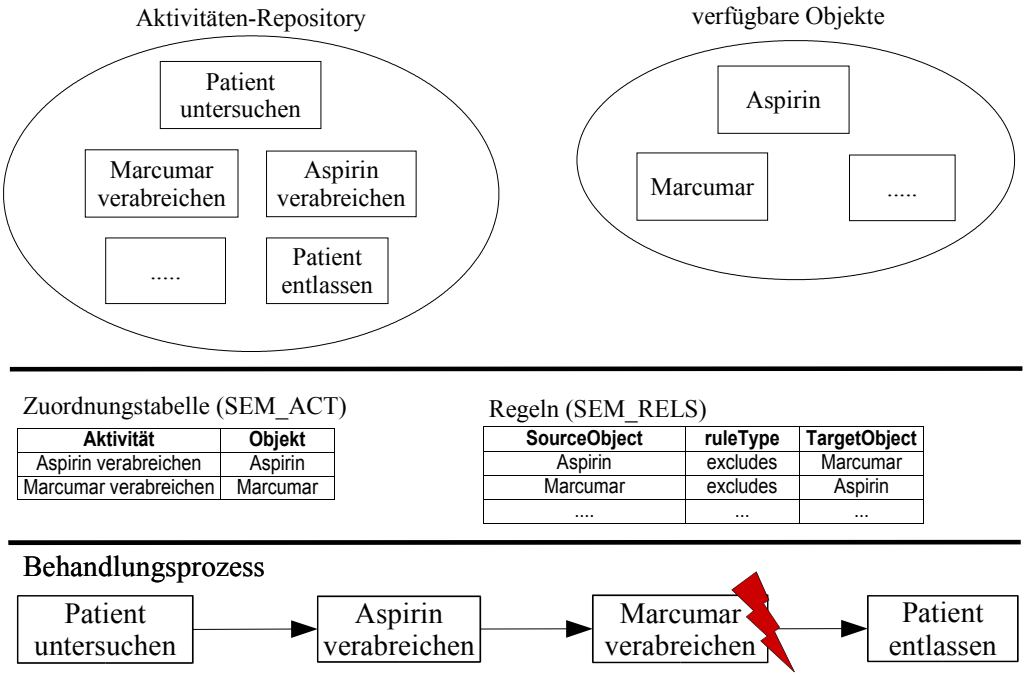


Abbildung 3.2.: Semantischer Konflikt - brokenExclusion

3.4. Änderungsoperationen und Semantik

In den bisherigen Ausführungen wurde nicht explizit auf die verschiedenen Änderungsoperationen eingegangen, denen eine Instanz oder ein Prozessschema unterzogen werden kann. Wie in Kapitel 2 beschrieben, gibt es unterschiedliche Änderungsoperationen:

- Einfügen: In einen Prozess bzw. eine Instanz wird eine Aktivität eingefügt.
- Löschoperation: Eine Aktivität wird aus einem Prozess bzw. einer Instanz entfernt.
- Verschiebeoperation: Eine Aktivität wird innerhalb des Prozesses bzw. einer Instanz an eine neue Position verschoben.

Es stellt sich die Frage, was für semantische Konflikte in Folge der jeweiligen Änderung im Prozess auftreten können.

Durch das Einfügen einer neuen Aktivität im Zuge einer Einfügeoperation kommen neue Objekte aus der semantischen Beschreibung der Aktivität und neue Regeln, die die semantischen Beziehungen der einzufügenden Aktivität repräsentieren, in den Prozess. Um die semantische Korrektheit einer Einfügeoperation zu überprüfen, müssen zwei Tests durchgeführt werden:

- Es muss geprüft werden, ob die Objekte der einzufügenden Aktivität mit Regeln von Objekten in Konflikt geraten, die Aktivitäten des Prozesses zugeordnet sind.

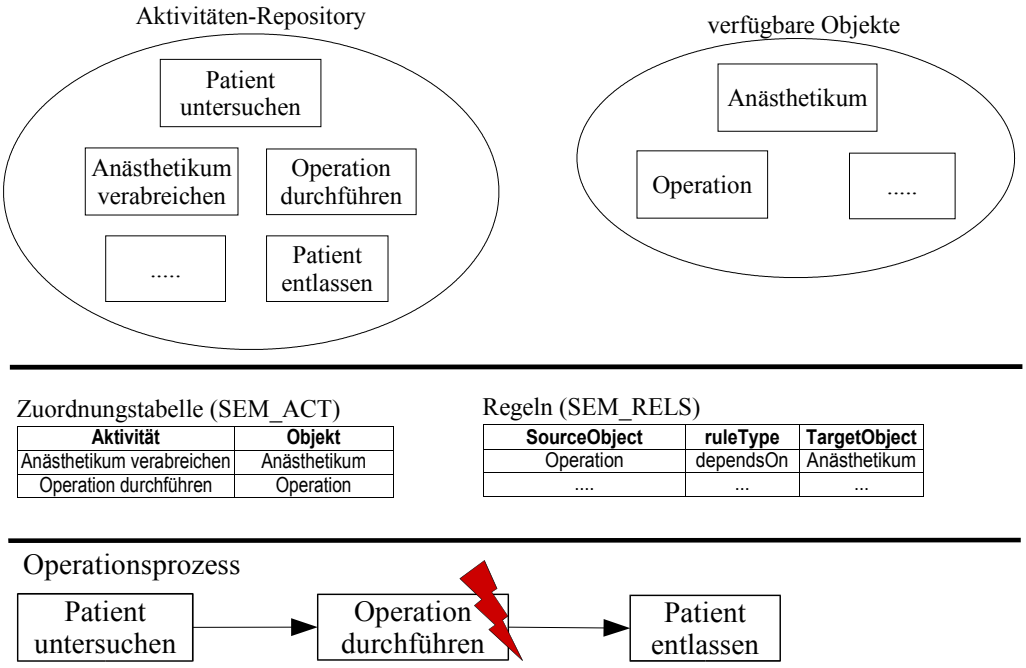


Abbildung 3.3.: Semantischer Konflikt - brokenDependency

- Die Regeln der einzufügenden Aktivität müssen ausgewertet werden, um festzustellen, ob sie mit Objekten der im Prozess vorhandenen Aktivitäten in Konflikt stehen.

Beim ersten Schritt des Korrektheitstests kann es zu semantischen Konflikten kommen, wenn Objekte der einzufügenden Aktivität im Prozess vorhandene Regeln verletzen. Objekte einer einzufügenden Aktivität können jedoch keine Abhängigkeitsbedingungen des Prozesses verletzen, da durch das Hinzufügen eines Objekts kein Einfluss auf die bereits erfüllten Abhängigkeiten des Prozesses genommen wird. Im Gegensatz dazu können Ausschlussbedingungen der Regeln des Prozesses durch Objekte der einzufügenden Aktivität verletzt werden.
Anders stellt sich der zweite Schritt beim Korrektheitstest dar. Die Regeln der Objekte der einzufügenden Aktivität können beide Arten von semantischen Konflikten verursachen, da sowohl eine geforderte Abhängigkeit eines Objekts durch den Prozess unerfüllt bleiben, als auch eine festgelegte Ausschlussbedingung durch die Objekte des Prozesses verletzt werden kann.

Beim Löschen wird eine Aktivität zusammen mit den Objekten ihrer semantischen Beschreibung und den zugehörigen Regeln aus dem Prozess entfernt. Da keine neuen Objekte in den Prozess eingefügt werden, kann kein neuer semantischer Konflikt durch eine verletzte Ausschlussbeziehung als Folge des Löschens entstehen. Anders sieht es mit einem neuen semantischen Konflikt durch eine nicht erfüllte Abhängigkeit aus. Wird eine

Abhängigkeit eines Objektes des Prozesses durch ein Objekt der zu löschenden Aktivität aufgelöst, so ist diese Abhängigkeit nach dem Löschen der Aktivität nicht länger erfüllt. Beim Löschen muss also folgendes getestet werden:

- Ist ein Objekt der zu löschenden Aktivität Zielobjekt einer im Prozess vorhandenen Regel, die nicht zur zu löschenden Aktivität gehört?
- Gibt es eine solche Regel, handelt es sich dann um eine Abhängigkeitsbeziehung? Um dann zu entscheiden, ob das Löschen der Aktivität die identifizierte Abhängigkeit unaufgelöst zurücklässt, muss das eventuell vorhandene `directionTarget`-Attribut der Regel untersucht werden. Damit kann festgestellt werden, ob die Position der zu löschenden Aktivität dem in der Regel vorgegebenen `directionTarget`-Attribut entspricht, so dass entschieden werden kann, ob die zu löschende Aktivität die geforderte Abhängigkeit erfüllt hat.

Bei der dritten zu untersuchenden Änderungsoperation handelt es sich um das Verschieben einer Aktivität des Prozesses an eine andere Stelle im Prozess. Durch das Verschieben kommen keine neuen Objekte und Regeln in den Prozess und es werden keine Objekte und Regeln entfernt. Die Objekte und Regeln der zu verschiebenden Aktivität werden jedoch in einen neuen Kontext gesetzt. Das bedeutet, dass eine Neuauswertung stattfinden muss. Diese stellt sich wie folgt dar:

- Es muss geprüft werden, ob das Verschieben Abhängigkeiten unerfüllt zurücklässt, die bisher durch die Aktivität aufgelöst wurden.
- Es muss geprüft werden, ob die Aktivität an der neuen Position Ausschlussregeln des Prozesses verletzt, so dass semantische Konflikte durch verletzte Ausschlussbedingungen entstehen.
- Die Regeln der zu verschiebenden Aktivität müssen auf verletzte Ausschlussbedingungen oder nicht länger erfüllte Abhängigkeiten geprüft werden.

Vergleicht man den Ablauf bei der Verschiebeoperation mit den Abläufen der Einfüge- und Löschoperation, so fällt auf, dass die Verschiebeoperation eine Kombination der Korrektheitstests der Einfügeoperation und der Löschoperation erfordert. Das Einfügen und das Löschen können deshalb als atomare Operationen betrachtet werden, das Verschieben als zusammengesetzte Operation.

Satz 3.1 fasst die semantischen Konflikte, die als Folge der Einfüge- und Löschoperation auftreten können, formal zusammen. Die Betrachtungen gelten aus den genannten Gründen ebenso für die Verschiebeoperation.

Satz 3.1 (Semantische Konflikte bei Änderungsoperationen)

S = (N,E,D,Dom) sei das Schema des Prozesses P in der Domäne
Dom = ($\mathcal{N}$*,*$\mathcal{O}, \mathcal{R}, SEM\_ACT, SEM\_RELS$*).*

i) Einfügeoperation (insert): insert(S,A_{insert}*,*A_i*,*A_j*,mode)*

a.) semantischer Konflikt brokenExclusion(O_m,O_n) wenn gilt:
$(O_m \in SEM\_ACT(A_k)) \wedge (A_k \in N) \wedge (O_n \in SEM\_ACT(A_{insert}))$

b.) semantischer Konflikt brokenExclusion(O_m,O_n) wenn gilt:
$(O_m \in SEM\_ACT(A_{insert})) \wedge (O_n \in SEM\_ACT(A_k)) \wedge (A_k \in N)$

c.) semantischer Konflikt brokenDependency(O_m,O_n) wenn gilt:
$(O_m \in SEM\_ACT(A_{insert})) \wedge (\nexists\ A_k \in N \text{ mit } O_n \in SEM\_ACT(A_k))$

ii) Löschoperation (delete): delete(S,A_{delete})

semantischer Konflikt brokenDependency(O_m,O_n) wenn gilt:
$(O_m \in SEM\_ACT(A_k)) \wedge (A_k \in N) \wedge (O_n \in SEM\_ACT(A_{delete})) \wedge (A_k \neq A_{delete})$

In den angestellten Betrachtungen zu den Auswirkungen von Änderungsoperationen wurde implizit die Annahme vorausgesetzt, dass zwischen den einer Aktivität zugeordneten Objekten keine semantischen Beziehungen bestehen. Diese Vereinfachung hatte das Ziel, die grundlegenden Auswirkungen von Änderungsoperationen für diesen Normalfall klar herausarbeiten zu können. Im Folgenden wird kurz darauf eingegangen, welche Konsequenzen sich ergeben, wenn diese Vereinfachung aufgehoben wird.

Existiert eine `excludes`-Beziehung zwischen zwei der Aktivität zugeordneten Objekten, würde dies zu einer semantisch nicht korrekten Aktivität führen. Dies kann bei der Beschreibung der Aktivitäten durch Zuordnung von Objekten algorithmisch geprüft und damit verhindert werden. Im Falle von `dependsOn`-Beziehungen kann es jedoch erwünscht sein, zwei Objekte zwischen denen eine Abhängigkeit besteht, der selben Aktivität zuzuordnen. Bei der Einfügeoperation könnte es zu der Situation kommen, dass eine geforderte Abhängigkeit durch die Objekte der Aktivitäten des Prozesses nicht aufgelöst wird und deshalb ein semantischer Konflikt signalisiert wird. Diese Abhängigkeit wird jedoch durch die der einzufügenden Aktivität zugeordneten Objekte selbst aufgelöst. Als Konsequenz ergibt sich daraus, dass bei einer Einfügeoperation zusätzlich zu den Objekten der Aktivitäten des Prozesses auch die Objekte der einzufügenden Aktivität in die Auswertung der Semantikregeln der einzufügenden Aktivität miteinbezogen werden müssen. Es wird sozusagen dem Abschluss der Einfügeoperation auf logischer Ebene bereits vorgegriffen und die Objekte der einzufügenden Aktivität als bereits im Prozess vorhanden betrachtet.

3.5. Änderungsoperationen und Teilprozesse

In adaptiven PMS sind häufig nicht nur einzelne Aktivitäten Ziel einer Änderungsoperation, sondern es muss ermöglicht werden, dass auch komplette Teilprozesse Gegenstand einer Änderungsoperation sein können. Dies umfasst das Einfügen eines vollständigen

Teilprozesses in einen Zielprozess, das Entfernen eines Teilprozesses aus einem Zielprozess oder das Verschieben eines Teilprozesses innerhalb eines Zielprozesses. Auf den ersten Blick erscheint es, als ob Aufgaben dieser Art einfach durchgeführt werden können, indem über die einzelnen Aktivitäten des Teilprozesses iteriert und jede einzelne Aktivität mit dem entsprechenden Algorithmus der Änderungsoperation behandelt wird. Je nach Änderungsoperation gibt es jedoch einige Fallstricke zu beachten.

Für das Einfügen eines Teilprozesses gilt, dass Objekte einer Aktivität des Teilprozesses Abhängigkeiten besitzen können, die durch Objekte von anderen Aktivitäten des einzufügenden Teilprozesses aufgelöst werden. Beim Zerlegen des Teilprozesses in einzelne Aktivitäten und durch das Einfügen jeder einzelnen Aktivität, kann der Fall eintreten, dass bei einer Aktivität ein semantischer Konflikt durch eine verletzte Abhängigkeitsbedingung signalisiert wird. Diese Abhängigkeit wird jedoch eventuell durch eine im einzufügenden Teilprozess erst später auftretende Aktivität aufgelöst. Um dieses Problem zu lösen, gibt es zwei unterschiedliche Vorgehensweisen:

- Vor dem Einfügen in den Zielprozess wird der Teilprozess untersucht und nur die Abhängigkeitsregeln beim Einfügen in den Prozess geprüft, die nicht durch Aktivitäten des Teilprozesses selbst aufgelöst werden.
- Die einzelnen Aktivitäten des Teilprozesses werden in den Zielprozess eingefügt. Tritt ein vermeintlicher Konflikt durch eine verletzte Abhängigkeitsbeziehung auf, so wird dieser Konflikt nicht sofort gemeldet und das Einfügen als nicht möglich eingestuft, sondern dieser Konflikt wird in einer Liste potentiell unaufgelöster Abhängigkeiten vorgemerkt. Beim Einfügen jeder weiteren Aktivität wird diese Liste durchsucht. Löst die einzufügende Aktivität eine der dort vorhandenen Abhängigkeiten auf, so wird diese aus der Liste gelöscht. Ist nach dem Test der letzten einzufügenden Aktivität die Liste leer, so besteht kein semantischer Konflikt. Befindet sich jedoch weiterhin eine nicht erfüllte Abhängigkeit in der Liste, so scheitert das Einfügen des gesamten Teilprozesses.

Die erste genannte Strategie verlangt nach einem Algorithmus, der dem Einfügealgorithmus vorgeschaltet ist. Der Nachteil dabei ist, dass dieser Algorithmus in Fällen, in denen kein semantischer Konflikt durch eine nicht erfüllte Abhängigkeit einer Aktivität des Teilprozesses besteht, trotzdem komplett durchlaufen werden muss und deswegen einen erhöhten und unnötigen Laufzeitaufwand verursacht.
Die zweite dargestellte Möglichkeit hingegen verzichtet auf ein gesondertes Prüfen des einzufügenden Teilprozesses. Gibt es keinen semantischen Konflikt durch eine verletzte Abhängigkeitsbedingung, so bleibt der Laufzeitaufwand für den Test eines vollständigen Teilprozesses linear zur Anzahl der einzufügenden Aktivitäten, wenn als Bezugsgröße das Einfügen einer Einzelaktivität herangezogen wird.
Die zweite Strategie stellt die effizientere Lösung dar, da eine Änderungsoperation auf einem Teilprozess keine gesonderte Behandlung des Teilprozesses erfordert und im Falle des erfolgreichen Abschlusses der Änderungsoperation kein Zusatzaufwand entsteht. Die zweite Methode stellt damit die Alternative der Wahl für eine Umsetzung dar.

Die Objekte und Regeln der aktuell untersuchten Aktivität eines sequentiell einzufügenden Teilprozesses müssen nach erfolgreichem Korrektheitstest der Aktivität in die entsprechenden Datenstrukturen des Zielprozesses eingefügt werden. Da bei einem Konflikt, verursacht durch eine Aktivität des Teilprozesses, bereits Objekte und Regeln von anderen Aktivitäten des Teilprozesses in den entsprechenden Datenstrukturen des Prozesses vorhanden sein können, ist ein "Rollback" notwendig. Folgende Möglichkeiten stellen Alternativen zum Umgang mit diesem Problem dar:

- Objekte und Regeln, die zu Aktivitäten des einzufügenden Teilprozesses gehören, werden in den entsprechenden Datenstrukturen des Prozesses als temporär gekennzeichnet. Schlägt das Einfügen des Prozesses fehl, können die Datenstrukturen durchsucht und als temporär markierte Einträge entfernt werden.
- Anlegen von temporären Objekt- und Regel-Caches für die einzufügenden semantischen Informationen, die zusätzlich bei den durchzuführenden Tests geprüft werden. Bei einem semantischen Konflikt besteht das "Rollback" im Löschen dieser Caches. Tritt kein Konflikt auf, so werden die Daten aus diesen temporären Caches in die entsprechenden Datenstrukturen des Prozesses übertragen.

Der erste Vorschlag hat den Vorteil, dass die Objekte und Regeln nach dem vollständigen Durchlaufen des Einfüge-Algorithmus bereits in den korrekten Datenstrukturen stehen. Es muss lediglich die Markierung entfernt werden. Der zusätzliche Laufzeitaufwand im Erfolgsfall ist damit sehr gering. Andererseits ist ein "Rollback" relativ aufwendig, da die markierten Objekte und Regeln aus den entsprechenden Datenstrukturen des Prozesses entfernt werden müssen. Im Gegensatz dazu ist die Beendigung des Einfüge-Algorithmus aufwendiger, wenn man der zweiten Möglichkeit folgt, da die Objekte und Regeln in die endgültigen Datenstrukturen übertragen werden müssen. Der Aufwand dabei hängt stark von der Organisation der Datenstrukturen ab. Ausgeglichen wird dieser Nachteil damit, dass ein "Rollback" trivial zu bewerkstelligen ist. Beide Alternativen bieten die Möglichkeit, das Konzept von Transaktionen bei Änderungsoperationen auf die Semantikinformationen zu übertragen. Dabei handelt es sich um ein "Alles-Oder-Nichts"-Prinzip, d.h. entweder werden die semantischen Informationen des Prozesses entsprechend der jeweiligen Änderungsoperation aktualisiert oder der Zustand vor der Anwendung der Änderungsoperation ist eindeutig wiederherstellbar.

Beide Strategien sind praxistauglich. Die Entscheidung für eine der Strategien hängt davon ab, ob der Schwerpunkt auf einen einfachen, unaufwendigen "Rollback" gelegt wird, oder ob der Laufzeitaufwand für die Beendigung des Einfügens im Erfolgsfall minimiert werden soll.

Es bleibt die Frage zu beantworten, ob auch das Löschen eines Teilprozesses besondere Maßnahmen erfordert. Beim Löschen können semantische Konflikte auftreten, wenn eine zu löschende Aktivität Ziel einer Abhängigkeitsbeziehung ist und diese Abhängigkeit auch tatsächlich auflöst. Ist nun ein kompletter Teilprozess Ziel einer Löschoperation, so kann ein durch das Löschen auftretender Konflikt sein Quellobjekt innerhalb des zu löschenden Teilprozesses haben. In diesem Fall ist der identifizierte semantische Konflikt

kein real existierender Konflikt, da die Aktivität, zu der die nicht länger aufgelöste Abhängigkeit gehört, ebenfalls aus dem Prozess entfernt wird. Dieser Fall ist einfach zu erkennen, da über die Regel, die verletzt wird, das zugehörige Objekt und damit die konfliktverursachende Aktivität identifiziert werden können. Liegt die so erhaltene Aktivität innerhalb des zu löschenden Teilprozesses, so hat der aufgetretene semantische Konflikt keinerlei Relevanz und kann ignoriert werden. Ist die Aktivität hingegen nicht innerhalb des zu löschenden Teilprozesses angesiedelt, so verursacht das Löschen einen Konflikt im Zielprozess. Das Löschen kann nicht durchgeführt werden. Erst nach einem erfolgreichen Korrektheitstest für den kompletten Teilprozess werden die Objekte und Regeln tatsächlich aus dem Prozess entfernt.

Das Verschieben stellt wie bereits dargelegt eine Kombination aus dem Einfügen und dem Löschen dar. Demzufolge gelten auch die Maßnahmen, die beim Einfügen und Löschen eines Teilprozesses ergriffen werden müssen. Durch die neue Position können sowohl innerhalb als auch außerhalb des Teilprozesses unaufgelöste Abhängigkeiten entstehen. Ebenso können durch die neue Position Ausschlussbeziehungen verletzt werden. Wird ein semantischer Konflikt verursacht, kann das Verschieben des Teilprozesses nicht durchgeführt werden.

Ein letzter Aspekt, der in den bisherigen Betrachtungen über semantische Konflikte nicht klar herausgearbeitet wurde, ist der Einfluss verschiedener Ausführungspfade auf das tatsächliche Eintreten semantischer Konflikte. Dieser Aspekt ist Gegenstand des nächsten Kapitels.

3.6. Weitere Aspekte semantischer Konflikte

Wird ein semantischer Konflikt innerhalb eines Prozesses identifiziert, beispielsweise durch eine Änderungsoperation, ist noch nicht sichergestellt, dass dieser potentielle Konflikt auch tatsächlich zum Tragen kommt. Dabei spielen die Positionen der an dem Konflikt beteiligten Aktivitäten eine Rolle. Prozesse bestehen nicht lediglich aus einer sequentiellen Abfolge von Aktivitäten, sondern besitzen durch AND- oder XOR-Splits auch unterschiedliche Ausführungspfade. Bei einem AND-Split werden die ausgehenden Pfade parallel ausgeführt, wohingegen die Pfade, die in einem XOR-Split ihren Ursprung haben, selektiv ausgeführt werden, d.h. nur einer der Pfade kommt tatsächlich zur Ausführung. Abbildung 3.4 und Abbildung 3.5 illustrieren dies an Beispielen. In Abbildung 3.4 ist der aus Abbildung 3.2 bekannte Prozess dahingehend abgeändert, dass die Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen` nicht länger in einer sequentiellen Abfolge liegen, sondern vielmehr in zwei unterschiedlichen Pfaden eines XOR-Splits. Die sequentielle Abfolge aus Abbildung 3.2 führte zu einem semantischen Konflikt zwischen den zwei genannten Aktivitäten. Die in den Objekten `Aspirin` und `Marcumar` festgelegten wechselseitigen Ausschlussbeziehungen gelten nach wie vor auch in Abbildung 3.4. Durch den XOR-Split wird jedoch immer nur eine der beiden in Konflikt stehenden Aktivitäten ausgeführt. Es bleibt folgendes zu klären:

- Gilt eine Ausschlussbeziehung generell, wenn zwei Aktivitäten die miteinander in Konflikt stehen, in ein und demselben Prozess verwendet werden?
- Hat eine Ausschlussbeziehung nur dann eine tatsächliche Auswirkung, wenn die in Konflikt stehenden Aktivitäten in einem Ausführungspfad oder zumindest in parallelen Ausführungspfaden des Prozesses liegen?

Für den ersten Punkt spricht, dass der Test auf semantische Korrektheit nicht dadurch komplizierter wird, dass zusätzlich die verschiedenen Ausführungspfade eines Prozesses berücksichtigt werden müssen und damit die Zugehörigkeit von Aktivitäten zu einem Ausführungspfad effizient festgestellt werden muss. Gegen die erste Alternative spricht jedoch der Sachverhalt, dass die semantischen Beziehungen zwischen den Objekten auf globaler Ebene festgelegt werden, d.h. ohne das Wissen über eine bestimmte Prozessstruktur. Dabei liegt das Augenmerk darauf, dass die definierten Beziehungen für Objekte gelten, die im Zuge des Ablaufs eines Prozesses tatsächlich zur Ausführung kommen. Die zweite genannte Möglichkeit trifft aus diesen Gründen auf die Behandlung semantischer Beziehungen zu, so dass Ausschlussbeziehungen nur dann relevant sind und semantische Konflikte verursachen, wenn die in Konflikt stehenden Aktivitäten tatsächlich alle ausgeführt werden.

In Abbildung 3.4 wird dies deutlich. Einem Patienten dürfen nicht beide Medikationen, `Aspirin` und `Marcumar`, zuteil werden. Dies soll die definierte Ausschlussbeziehung zwischen den Objekten `Aspirin` und `Marcumar` verhindern. Nun sorgt der XOR-Split in dem Beispiel dafür, dass die beiden Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen`, die einen potentiellen Konfliktfall darstellen, in unterschiedlichen Ausführungspfaden liegen, so dass nur eine der beiden Aktivitäten auch tatsächlich ausgeführt wird. Das bedeutet, dass dem Patient entweder die Medikation `Aspirin` oder die Medikation `Marcumar` gegeben wird. Somit besteht der potentielle Konflikt in der Ausführung nicht und der Prozess ist semantisch korrekt.

Abbildung 3.5 zeigt nun den selben Prozess, in dem jedoch der XOR-Split durch einen AND-Split ersetzt wurde. Folgt man der Argumentation, die für den XOR-Split gilt, so erhärtet sich der potentielle Konflikt zwischen den Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen` zu einem tatsächlichen semantischen Konflikt, so dass der in Abbildung 3.5 dargestellte Prozess semantisch nicht korrekt ist. Die beiden Aktivitäten unterliegen einer parallelen Ausführung. Damit greift die semantische Ausschlussbeziehung, da dem Patienten beide Medikationen verabreicht werden.

Die selben Untersuchungen können für Abhängigkeitsbeziehungen vorgenommen werden. Abbildung 3.6 greift den bereits bekannten Operationsprozess aus Abbildung 3.3 wieder auf. Nun ist jedoch die Aktivität `Anästhetikum verabreichen` im Prozess vorhanden, die die geforderte Abhängigkeit des Objektes `Operation` aus der Aktivität `Operation durchführen` auflösen kann. Diese Aktivität ist jedoch in einem alternativen Ausführungspfad des XOR-Splits angesiedelt, so dass in dem Prozess nun entweder die Aktivität `Anästhetikum verabreichen` oder die Aktivität `Operation durchführen` zur Ausführung kommt. Dies erzeugt nicht den gewünschten Effekt, denn wenn

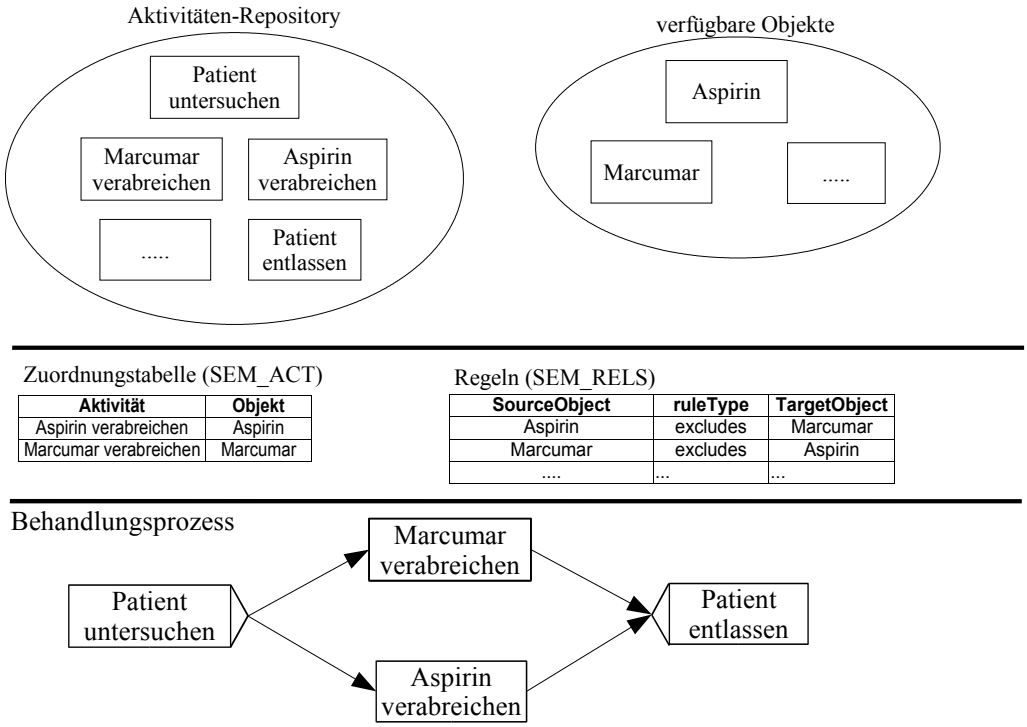


Abbildung 3.4.: XOR-Split und Auswirkungen auf Ausschlussbeziehungen

der Patient operiert wird, wurde in diesem Prozess nicht die Betäubung durchgeführt. Ein XOR-Split zwischen zwei abhängigen Aktivitäten bedeutet, dass die geforderte Abhängigkeit nicht aufgelöst ist, so dass im Gegensatz zu einer Ausschlussbeziehung ein XOR-Split bei einer Abhängigkeitsbeziehung zu einem semantischen Konflikt führt.

Das Verhalten einer Abhängigkeitsbeziehung bei einem AND-Split zwischen den abhängigen Aktivitäten bleibt schließlich noch zu klären. In Abbildung 3.7 wird die Aktivität `Anästhetikum verabreichen` parallel zu der Aktivität `Operation durchführen` ausgeführt. Nach der bisherigen Argumentation wird der dargestellte Prozess als semantisch korrekt eingestuft. Es ist offensichtlich, dass in einem Operationsprozess sichergestellt sein sollte, dass eine Aktivität `Anästhetikum verabreichen` vor der eigentlichen Operation durchgeführt werden muss. Der Prozess in Abbildung 3.7 besitzt also einen semantischen Konflikt der nicht erkannt wird. Dieser Konflikt ist nicht darauf zurückzuführen, dass die Mechanismen nicht korrekt sind oder versagen, sondern darauf, dass die Abhängigkeitsregel nicht präzise genug formuliert wurde.
Abhängigkeitsbeziehungen sind sehr häufig gerichtete Beziehungen, d.h. oftmals ist es notwendig, dass eine geforderte Abhängigkeit durch eine Aktivität, die vor oder nach der Aktivität mit dem zugeordneten Objekt, das die Abhängigkeit verursacht, im Prozess ausgeführt wird. Die im Beispiel modellierte Regel müsste eine solche Richtungsbeziehung festlegen, so dass die Abhängigkeit nur dann aufgelöst wird, wenn die Aktivität

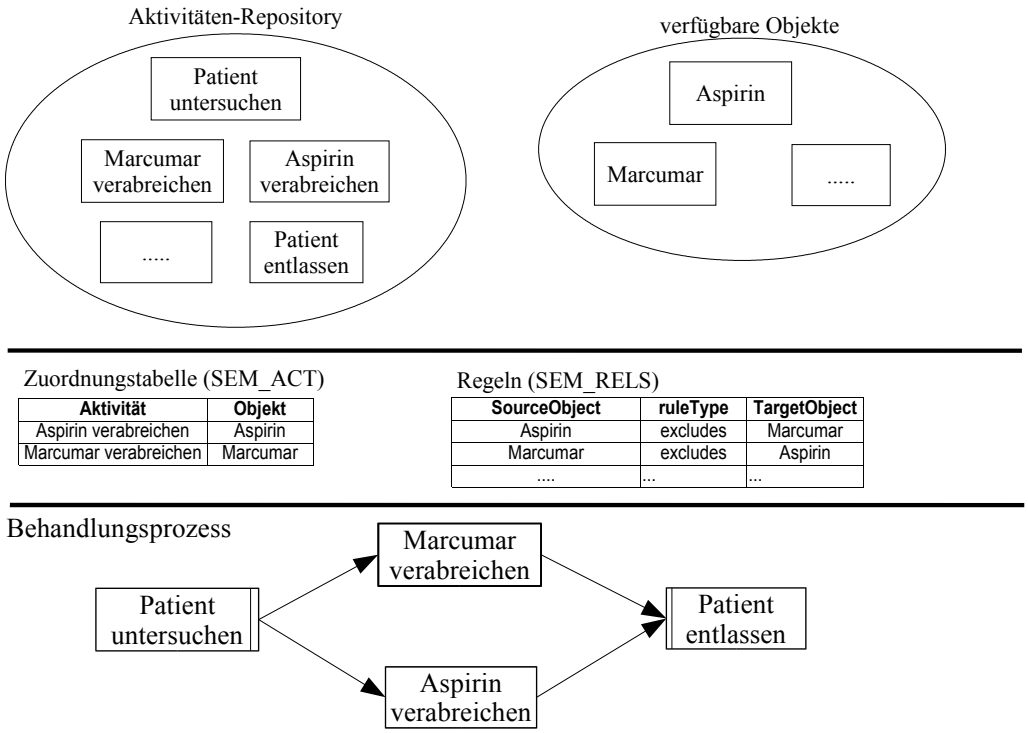


Abbildung 3.5.: AND-Split und Auswirkungen auf Ausschlussbeziehungen

`Anästhetikum verabreichen` vor der Aktivität `Operation durchführen` ausgeführt wird.
Die Regel muss `dependsOn(Operation,Anästhetikum,pre)` lauten, anstelle von `dependsOn(Operation,Anästhetikum)`. Mit dieser Regel würde im Prozess von Abbildung 3.7 korrekterweise ein semantischer Konflikt durch eine nicht aufgelöste Abhängigkeit erkannt, da über die Reihenfolge der Aktivitäten in parallelen Ausführungssträngen keine Aussage möglich ist.

Die Position von Aktivitäten und damit von Objekten innerhalb unterschiedlicher Ausführungsstränge hat je nach Art der Aufspaltung, XOR- oder AND-Split, sowie je nach Art der semantischen Beziehung Einfluss darauf, ob ein semantischer Konflikt vorliegt oder nicht. Folgende Vereinfachung gilt:
Ist innerhalb einer Regel eine relative Positionsangabe über das Attribut `direction-Target` angegeben, so ist die Regel nicht erfüllt, wenn sich die zur Regel gehörenden Objekte und damit Aktivitäten in unterschiedlichen Ausführungspfaden befinden, unabhängig davon, ob es sich um XOR- oder AND-Splits handelt. Für Ausschlussbeziehungen hat dieser Sachverhalt eine andere Bedeutung als für Abhängigkeitsbeziehungen, da im ersten Fall kein semantischer Konflikt vorliegt, im zweiten Fall jedoch sehr wohl.

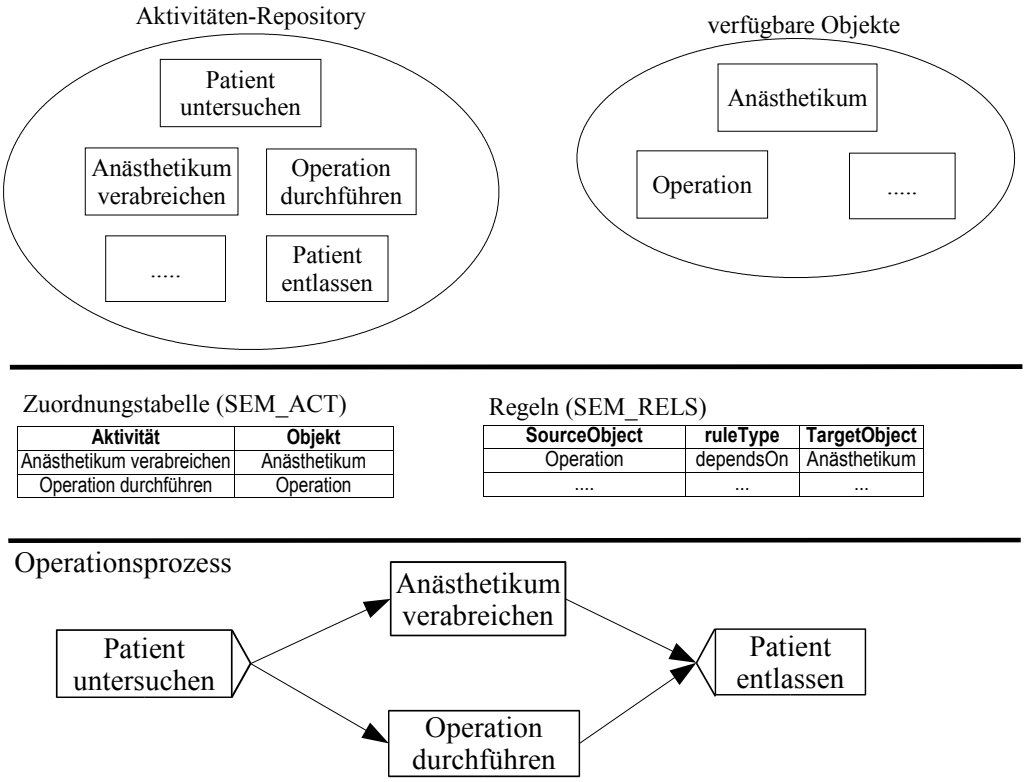


Abbildung 3.6.: XOR-Split und Auswirkungen auf Abhängigkeitsbeziehungen

Im Folgenden wird ein Algorithmus skizziert, der die Bestimmung des tatsächlichen Eintretens eines semantischen Konflikts zulässt. Die zentralen Fragestellungen dabei lauten:

- Für den Fall einer scheinbar verletzten `excludes`-Beziehung: Existiert ein Ausführungspfad im untersuchten Prozess, auf dem die potentiell konfliktverursachenden Aktivitäten zusammen ausgeführt werden?
- Bei einer scheinbar aufgelösten Abhängigkeit: Wird die Aktivität, die die Abhängigkeit auflöst, gemeinsam mit der Aktivität ausgeführt, die die Abhängigkeit verursacht?

S = (N,E,D,Dom) sei ein gegebenes Prozessschema in der Domäne
Dom = ($\mathcal{N}$,$\mathcal{O}$, $\mathcal{R}$, SEM_ACT, SEM_RELS). Ausgangspunkt stellt eine Semantikregel SR(s,t) dar. Beide Objekte, s und t treten im Prozess auf. Die Regel wird also potentiell erfüllt. Folgende Schritte sind nun durchzuführen:

1. sn sei die potentiell konfliktverursachende Aktivität mit $s \in SEM\_ACT(sn)$. targetNodeList := $\{n \in N$ mit $t \in SEM\_ACT(n)\}$ sei die Menge aller Aktivitäten in S, denen das Objekt t in der Aktivitätenbeschreibung zugeordnet ist.

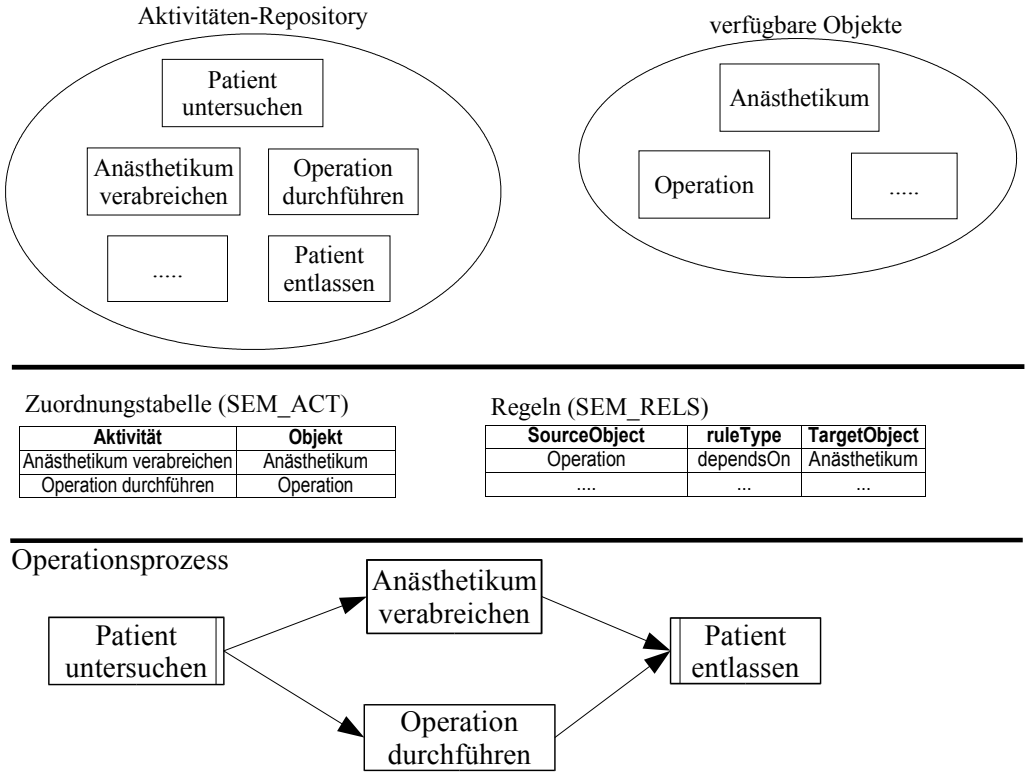


Abbildung 3.7.: AND-Split und Auswirkungen auf Abhängigkeitsbeziehungen

2. $\forall tn \in targetNodeList$ führe folgende Anweisungen aus:
   Kein weiteres Element tn vorhanden. ⇒ Gehe zu Schritt 6.
   Weiteres Element tn vorhanden:
   Falls $tn \in c\_pred^*(S, sn) \vee tn \in c\_succ^*(S, sn)$ gehe zu Schritt 5.
   Andernfalls weiter mit Schritt 3.

3. a) Bestimme die Menge aller direkten und indirekten Vorgängerknoten von sn, deren Knotentyp einen AND- oder XOR-Split darstellt und in deren Menge der direkten und indirekten Nachfolgerknoten jeweils tn enthalten ist:
   splitNodes:= $\{n \in c\_pred^*(S, sn) \mid (tn \in c\_succ^*(S, n)) \wedge ((\text{NT(n)} = \text{AND-Split}) \vee (\text{NT(n)} = \text{XOR-Split}))\}$
   Falls splitNodes = $\emptyset$: nächste Iteration von Schritt 2.
   Andernfalls weiter bei Schritt 3.b).

   b) Bestimme $splitTar \in splitNodes$, so dass folgende Bedingung erfüllt ist:
   $\forall split \in splitNodes \setminus splitTar$ gelte: $splitTar \in c\_succ^*(S, split)$

   Falls $(tn \in c\_succ^*(S, splitTar)) \wedge (NT(splitTar) = XOR - Split)$ gehe zu Schritt 4.

Falls $(tn \in c\_succ^*(S, splitTar)) \wedge (NT(splitTar) = AND - Split)$ gehe zu Schritt 5.

4. Die Aktivitäten sn und tn werden im Ablauf des Prozesses nie gemeinsam ausgeführt.

   Ist SR(s,t) eine Regel der Form excludes(s,t), so existiert kein semantischer Konflikt zwischen sn und tn.
   Ist SR(s,t) eine Regel der Form dependsOn(s,t), so löst tn die Abhängigkeit nicht auf.

   Weiter mit nächster Iteration in Schritt 2.

5. Es existiert eine Ausführungsfolge, in der die Aktivitäten sn und tn gemeinsam ausgeführt werden.

   Ist SR(s,t) eine Regel der Form excludes(s,t), so existiert der semantische Konflikt tatsächlich.
   Ist SR(s,t) eine Regel der Form dependsOn(s,t), so ist die Abhängigkeit aufgelöst und es existiert kein semantischer Konflikt.

   Gehe zu Schritt 7.

6. Es wurde keine Aktivität $n \in N$ mit $t \in SEM\_ACT(n)$ gefunden, die in einer möglichen Ausführung des Prozesses zusammen mit sn ausgeführt wird.

   Ist SR(s,t) eine Regel der Form excludes(s,t), so tritt der semantische Konflikt nicht ein.
   Ist SR(s,t) eine Regel der Form dependsOn(s,t), so wird die Abhängigkeit von keiner Aktivität des Prozesses aufgelöst und es existiert ein semantischer Konflikt.

   Gehe zu Schritt 7.

7. Ende des Algorithmus.

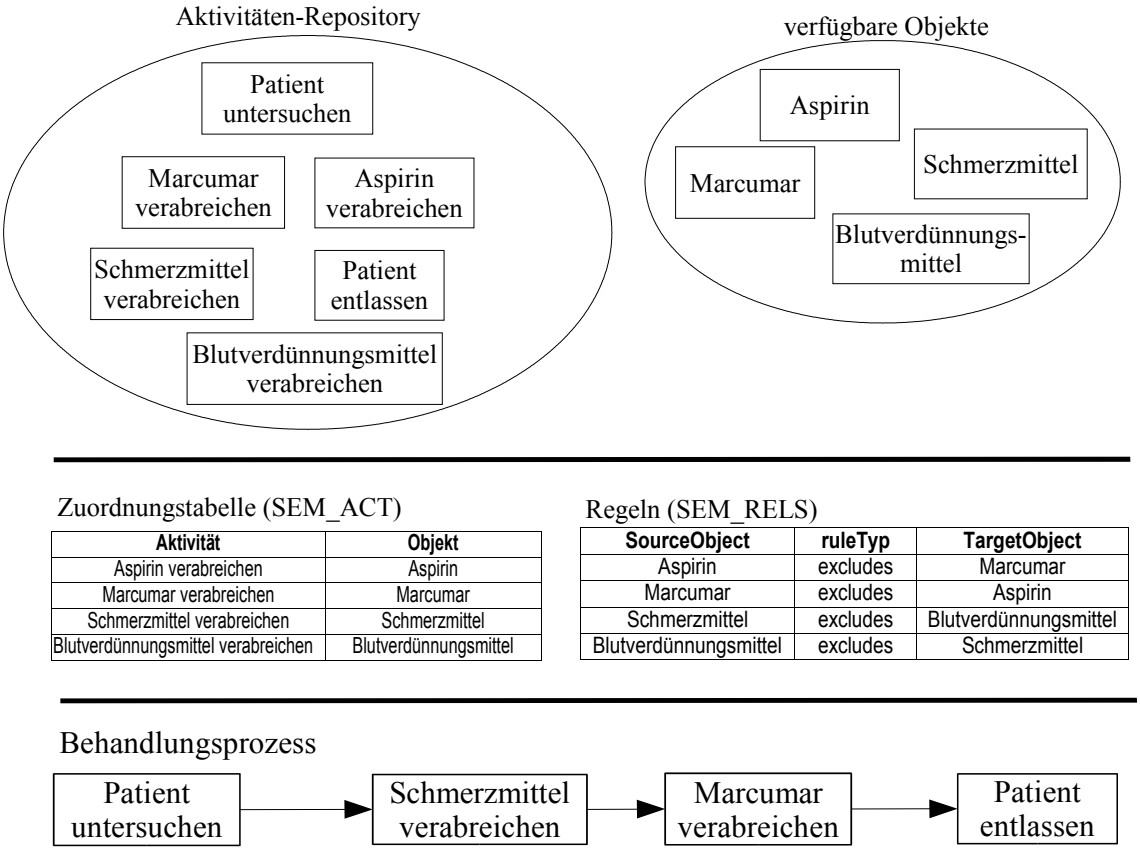
4

Organisation semantischer Informationen

Die Sicherstellung der semantischen Korrektheit basiert auf der semantische Beschreibung von Aktivitäten über die Zuordnung von Objekten, sowie der Festlegung von semantischen Beziehungen zwischen den Objekten. Bisher wurden diese Informationen als lose Mengen betrachtet. In diesem Teil der Arbeit wird herausgearbeitet, wie die semantischen Informationen organisiert werden können, um die Übersichtlichkeit zu wahren und maximalen Nutzen aus den semantischen Informationen zu ziehen.

4.1. Objekte als Typen

Semantische Beziehungen zwischen Objekten etablieren binäre Relationen zwischen zwei Objekten. Bei der Auswertung einer Regel wird im Prozess nach einem Vorkommen des angegebenen Zielobjekts der Regel gesucht. Wird dieses Zielobjekt nicht gefunden, so ist die Regel nicht erfüllt. Abbildung 4.1 zeigt diesen Zusammenhang. Gegenüber dem Beispiel in Abbildung 3.2 wurde der dargestellte Behandlungsprozess abgeändert, indem die Aktivität `Aspirin verabreichen` durch die Aktivität `Schmerzmittel verabreichen` ersetzt wurde. Im so abgeänderten Prozess wird nun kein semantischer Konflikt mehr signalisiert, da keine Regel festgelegt wurde, die eine Ausschlussbeziehung zwischen den Objekten `Schmerzmittel` und `Marcumar` definiert. Unter der Annahme, dass Marcumar nie zusammen mit einem Schmerzmittel verabreicht werden darf, ist der in Abbildung 4.1 dargestellte Prozess semantisch nicht korrekt, da dem Patienten sowohl eine Medikation Schmerzmittel als auch eine Medikation Marcumar verabreicht wird. In Abbildung 4.2 wird dies korrigiert, indem die fehlende semantische Information in Form der Regel `excludes(Marcumar,Schmerzmittel)` ergänzt wird. Nun tritt



Zuordnungstabelle (SEM_ACT)

Aktivität	Objekt
Aspirin verabreichen	Aspirin
Marcumar verabreichen	Marcumar
Schmerzmittel verabreichen	Schmerzmittel
Blutverdünnungsmittel verabreichen	Blutverdünnungsmittel

Regeln (SEM_RELS)

SourceObject	ruleTyp	TargetObject
Aspirin	excludes	Marcumar
Marcumar	excludes	Aspirin
Schmerzmittel	excludes	Blutverdünnungsmittel
Blutverdünnungsmittel	excludes	Schmerzmittel

Abbildung 4.1.: Semantik ohne Typbeziehungen

ein semantischer Konflikt zwischen den Aktivitäten `Schmerzmittel verabreichen` und `Marcumar verabreichen` auf, was der oben genannten Annahme entspricht.

Die semantische Korrektheit eines Prozesses kann nur dann garantiert werden, wenn sämtliche existierende semantische Beziehungen in Form von Regeln erfasst werden. In Anwendungsdomänen mit vielen definierten Objekten kann die Anzahl der zu definierenden semantischen Beziehungen große Ausmaße annehmen, so dass es schwierig ist, die Übersicht zu behalten. Leicht kann eine semantische Beziehung bei der Definition der Regeln übersehen werden, womit die semantische Konsistenz sämtlicher Prozesse in der Domäne gefährdet ist. Um diesen Sachverhalt zu untersuchen, werden folgende Regeln einer genaueren Betrachtung unterzogen:

- `excludes(Marcumar,Schmerzmittel)`
- `excludes(Marcumar,Aspirin)`

Beide genannten Regeln sind im Regelpool von Abbildung 4.2 enthalten und besitzen das selbe Quellobjekt `Marcumar`. Die Regeln unterscheiden sich lediglich im Zielobjekt,

Zuordnungstabelle (SEM_ACT)

Aktivität	Objekt
Aspirin verabreichen	Aspirin
Marcumar verabreichen	Marcumar
Schmerzmittel verabreichen	Schmerzmittel
Blutverdünnungsmittel verabreichen	Blutverdünnungsmittel

Regeln (SEM_RELS)

SourceObject	Rule	TargetObject
Aspirin	excludes	Marcumar
Marcumar	excludes	Aspirin
Schmerzmittel	excludes	Blutverdünnungsmittel
Blutverdünnungsmittel	excludes	Schmerzmittel
Marcumar	excludes	Schmerzmittel

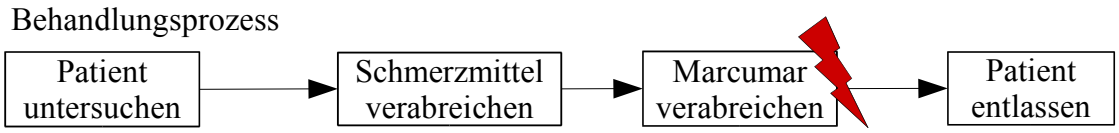


Abbildung 4.2.: Semantik ohne Typbeziehungen - erweiterte Regelmenge

`Schmerzmittel` in der ersten Regel, `Aspirin` in der zweiten Regel. Das Medikament Aspirin wird häufig zur Bekämpfung von Schmerz eingesetzt und stellt damit einen Vertreter der Schmerzmittel dar. Es besteht also ein Zusammenhang zwischen den beiden Objekten `Schmerzmittel` und `Aspirin`. Folglich ist es sinnvoll, dass Regeln mit dem Zielobjekt `Schmerzmittel` implizit auch `Aspirin` als Zielobjekt umfassen. Die Regel `excludes(Marcumar,Aspirin)` ist überflüssig.

Der durch das Beispiel deutlich gewordene Zusammenhang zwischen Objekten erfordert eine Organisation der Objekte einer Domäne, die solche Zusammenhänge auswertbar macht. Diese bisher abstrakt gehaltenen Zusammenhänge konkretisieren sich in dem Begriff der Typisierung. Dies bedeutet, dass zwischen verschiedenen Objekten einer Domäne Typbeziehungen bestehen. Dieser Sachverhalt manifestiert sich beispielsweise darin, dass das Objekt `Aspirin` ein ganz bestimmtes `Schmerzmittel` mit spezifischen Eigenschaften darstellt. Ähnliche Zusammenhänge finden sich in der objektorientierten Programmierung (OOP).
Die Organisationsform eines hierarchisch aufgebauten Typsystems ist durch folgende charakteristische Merkmale gekennzeichnet:

- Definition immer wiederkehrender Eigenschaften an einer Stelle.
- Intuitive Definition von Untertypen mittels Zuordnung zu einem Obertyp und anschließende Festlegung von speziellen Eigenschaften.
- Änderung zentraler Eigenschaften nur an einer Stelle.
  $\Longrightarrow$ Automatische Propagation auf die Untertypen.

Dieser Ansatz eines hierarchischen Typsystems muss im folgenden Schritt auf den Kontext von Prozessen und PMS übertragen werden, womit sich das nächste Kapitel befasst.

4.2. Ontologieansatz

Beim Abbilden einer Anwendungsdomäne in ein Typsystem läuft ein Transfer von Objekten und Bezugsgrößen der realen Welt in ein abstraktes System ab, das zur Abbildung der Objekte und Bezugsgrößen genau festgelegte Beziehungsformen bietet. Das Problem dieses Transfers liegt in der Bestimmung der für die Anwendung benötigten Objekte, ihrer Eigenschaften und ihrer Beziehungen zu anderen Objekten. Hier kann man eine Parallele zur KI ziehen, deren Arbeitsweise sich im Bereich der Wissensmodellierung und Wissenrepräsentation oftmals mit folgenden Schritten beschreiben lässt [Caw03]:

1. Aufbau einer Wissensbasis.
2. Erstellung von Inferenzalgorithmen, die auf der geschaffenen Wissensbasis arbeiten und diese auswerten (Inferenzmaschine).

Übertragen auf den Kontext Prozess-Management ist der Aufbau der Wissensbasis gleichbedeutend mit der Erstellung hierarchisch organisierter Objektsysteme, in denen sämtliche für die semantische Beschreibung von Aktivitäten zur Verfügung stehenden Objekte eingeordnet sind. Genauso findet sich mit den Auswertungsalgorithmen zur Ermittlung der semantischen Korrektheit eine Entsprechung der Inferenzalgorithmen. Konzeptuell kann der Aufbau einer Wissensbasis in Anlehnung an [DHH01] unter Berücksichtigung des Kontextes PMS wie in Abbildung 4.3 in vier Phasen eingeteilt werden.
Die Identifizierungsphase dient dazu, die Objekte und Anwendungssituationen der zu

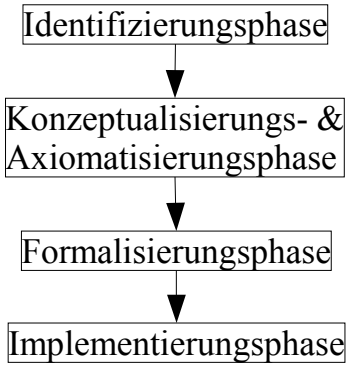


Abbildung 4.3.: Aufbau einer Wissensbasis - Phasen

erfassenden Domäne zu ermitteln. Um dies zu erreichen, sind die folgenden Fragen zu beantworten:

- Welche Anwendungssituationen sollen abgedeckt werden?
- Welche Aktivitäten werden benötigt?
- Welche Objekte werden benötigt, um die Aktivitäten zu beschreiben?

Die darauffolgende Phase ist die Konzeptualisierungsphase. In [GN87] wird der Begriff Konzeptualisierung wie folgt eingeführt:

"A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them."

[Gru93] greift diese Definition auf und erweitert sie:

"A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly."

Eine Konzeptualisierung ist nach dieser Aussage eine Abstraktion eines Ausschnitts der realen Welt, wobei diese Abstraktion durch Objekte und die Beziehungen zwischen ihnen beschrieben wird. Aus den Ergebnissen der Identifizierungsphase werden in der Konzeptualisierungsphase die identifizierten Konzepte, d.h. die Objekte, typisiert und in einem entsprechenden Typsystem organisiert. Dabei wird untersucht, wie feingranular aufgelöst die benötigte Objektmenge tatsächlich sein muss und welche Typbeziehungen zwischen den Objekten existieren. Darauf aufbauend wird ermittelt, welche Objekte für jede der identifizierten Aktivitäten benötigt werden, um sie semantisch vollständig beschreiben zu können.
Anknüpfend an die Konzeptualisierungsphase folgt die Formalisierungs- und Axiomatisierungsphase. Das konzeptuell erstellte Typsystem wird hier in ein formales Modell überführt. Schließlich werden die semantischen Beziehungen zwischen den Objekten ermittelt und in einer formalen Ausdrucksweise festgelegt, was der Axiomatisierung entspricht.
Die letzte Phase ist die Implementierungsphase, in der das erstellte formale Modell in eine maschinenlesbare Repräsentation überführt wird.

Die Identifizierungsphase, die die grundlegend festzulegenden Aktivitäten und Objekte liefert, ist nicht Gegenstand dieser Ausarbeitung und es wird im Folgenden davon ausgegangen, dass eine Domäne bereits entsprechend erfasst wurde. Schwerpunkte setzen vielmehr die Mechanismen der Konzeptualisierungsphase, sowie der Formalisierungs- und Axiomatisierungsphase.

Die Objekte einer Domäne werden nicht mehr in einer losen Menge organisiert, sondern in einem Typsystem, einer Konzeptualisierung. Der aus dem griechischen stammende Begriff Ontologie, der ursprünglich die philosophische Disziplin *"[...] vom Sein, dem Seienden als solchem und den fundamentalen Typen von Entitäten"* bezeichnete [Wik05], baut auf der Definiton der Konzeptualisierung auf. Gruber umschreibt den Begriff der Ontologie in [Gru93] als explizite Spezifikation einer Konzeptualisierung. Eine Ontologie legt nach dieser Definition eine Domäne aus der Sicht einer bestimmten Anwendung fest. Dabei werden die Objekte der Domäne in Form einer Taxonomie organisiert und die Beziehungen zwischen ihnen über Relationen und Axiome abgebildet. Oftmals stellt sich dieses Typsystem als Klassifizierungshierarchie heraus, was auch auf die Typisierung der Objekte von Prozessen zutrifft. Die Relationen zwischen Objekten beschränken sich

dabei auf Obertyp-Untertyp-Beziehungen. Die Regeln, die die semantischen Beziehungen zwischen den Objekten beschreiben, können als die Axiome der Ontologie betrachtet werden. Definition 4.1 formalisiert den Begriff Ontologie für den Kontext PMS.

Definition 4.1 (Ontologie - PMS)

Eine Ontologie ONT in einer Domäne $\mathcal{D}$ = ($\mathcal{N}$,$\mathcal{O}$, $\mathcal{R}$, SEM_ACT, SEM_RELS) wird durch ein Tupel ($\mathcal{O}$,R,$\mathcal{R}$) repräsentiert, wobei $\mathcal{O}$ der Menge der Objekte und $\mathcal{R}$ der Menge der semantischen Beziehungen der Domäne $\mathcal{D}$ entsprechen. R := {(s,t)|s,t ∈ $\mathcal{O}$} gibt die Menge der Typbeziehungen in ONT an.

Für o ∈ $\mathcal{O}$ bezeichne RD(o) := {SR(o,t)∈ $\mathcal{R}$} die für o festgelegten semantischen Beziehungen.

Definition 4.2 legt fest, wann ein Objekt einer Ontologie ein Untertyp eines zweiten Objekts ist.

Definition 4.2 (Untertypbeziehung)

Gegeben sei eine Ontologie ONT = ($\mathcal{O}$, R, $\mathcal{R}$). Ein Objekt o ∈ $\mathcal{O}$ ist Untertyp eines Objekts p ∈ $\mathcal{O}$, repräsentiert als subType(o,p) $\Longleftrightarrow$ $\exists$ $s_1, ..., s_k \in \mathcal{O}$, *so dass gilt:*
$s_1 = p \wedge s_k = o$ *und* $\forall i$ *mit* $1 \leq i < k$ $\exists$ $(s_i, s_{i+1}) \in R$

In Worten ausgedrückt bedeutet Definition 4.2, dass o Untertyp von p ist, wenn es eine Reihe zusammenhängender Typbeziehungen gibt, die von Objekt p ausgehen und in Objekt o münden.

Abbildung 4.4 stellt die bereits bekannten Objekte `Schmerzmittel`, `Blutverdünnungsmittel, Aspirin und Marcumar` in einer Klassifizierungshierarchie dar, deren Wurzel das neu eingeführte Objekt `Medikament` bildet. Die Klassifizierung der Objekte folgt dabei dem Prinzip "vom Allgemeinen hin zum Speziellen" und deckt sich mit den bereits weiter oben festgestellten Sachverhalten, dass Aspirin ein bestimmtes Schmerzmittel und Marcumar ein bestimmtes Medikament zur Blutverdünnung darstellen. Darauf aufbauend ändert sich die Regelmenge, die die semantischen Beziehungen der Objekte umfasst, wie in Abbildung 4.5 dargestellt. Es sind noch immer vier Regeln vorhanden. Die neu hinzugekommenen Regeln `excludes(Aspirin,Blutverdünnungsmittel)` und `excludes(Marcumar,Schmerzmittel)` subsumieren die wegfallenden Regeln `excludes(Aspirin,Marcumar)` und `excludes(Marcumar,Aspirin)`. Der Vorteil wird erst deutlich, wenn in der Ontologie beispielsweise ein weiterer Typ `Spalt` als Untertyp von `Schmerzmittel` festgelegt wird, für den der Ausschluss `excludes(Marcumar,Spalt)` gilt. Diese Regel muss dann nicht angegeben werden, da sie durch die Regel `excludes(Marcumar,Schmerzmittel)` abgedeckt ist. Mit der Regelmenge aus Abbildung 4.5 wird der semantische Konflikt zwischen den Aktivitäten `Schmerzmittel verabreichen` und `Marcumar verabreichen`, wie Abbildung 4.6 zeigt, ebenso erkannt wie in Abbildung 4.2. Die Typbeziehungen bewirken, dass eine definierte Regel alle Regeln in sich vereinigt, deren Zielobjekte Untertypen des in

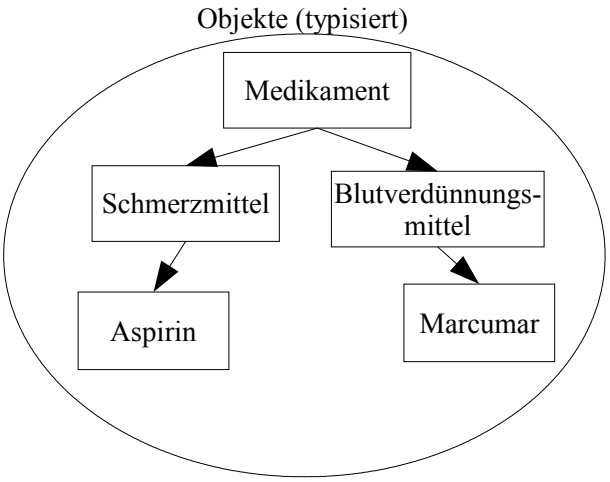


Abbildung 4.4.: Einfache Medikamenten-Ontologie

Regeln (SEM_RELS)

SourceObject	ruleType	TargetObject
Aspirin	excludes	Marcumar
Marcumar	excludes	Aspirin
Schmerzmittel	excludes	Blutverdünnungsmittel
Blutverdünnungsmittel	excludes	Schmerzmittel

Typbeziehungen

SourceObject	ruleType	TargetObject
Schmerzmittel	excludes	Blutverdünnungsmittel
Blutverdünnungsmittel	excludes	Schmerzmittel
Aspirin	excludes	Blutverdünnungsmittel
Marcumar	excludes	Schmerzmittel

Abbildung 4.5.: Regelmenge unter Anwendung von Typbeziehungen

der Definition der Regel angegebenen Zielobjekts sind. Dieser Sachverhalt wird in Definition 4.3 formal ausgedrückt.

Definition 4.3 (Substitution von Regeln)

Gegeben sei eine Ontolgie ONT = $(\mathcal{O}, R, \mathcal{R})$. $r \in \mathcal{R}$ mit r = SR(source,target) sei eine Regel mit dem Quellobjekt source $\in \mathcal{O}$ und dem Zielobjekt target $\in \mathcal{O}$. r substituiert dann alle Regeln der Menge Γ, wobei für Γ folgendes gilt:
$\Gamma := \{SR(source, t) | (t \in \mathcal{O}) \wedge (subType(t, target))\}$

Zuordnungstabelle (SEM_ACT)

Aktivität	Objekt
Aspirin verabreichen	Aspirin
Marcumar verabreichen	Marcumar
Schmerzmittel verabreichen	Schmerzmittel
Blutverdünnungsmittel verabreichen	Blutverdünnungsmittel

Regeln (SEM_RELS)

SourceObject	ruleType	TargetObject
Schmerzmittel	excludes	Blutverdünnungsmittel
Blutverdünnungsmittel	excludes	Schmerzmittel
Aspirin	excludes	Blutverdünnungsmittel
Marcumar	excludes	Schmerzmittel

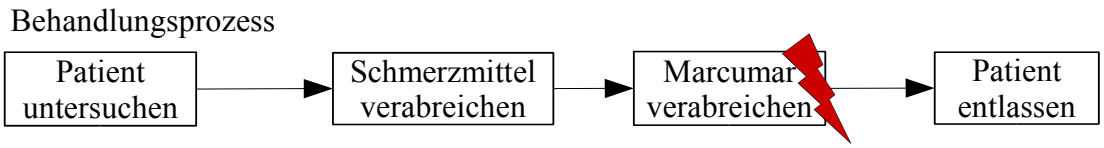


Abbildung 4.6.: Semantischer Konflikt durch Typbeziehungen

Ein Zahlenbeispiel kann eindrucksvoll verdeutlichen, wie sich die Berücksichtigung von Typbeziehungen auf die Anzahl an Regeln auswirkt. Angenommen es existieren zwei Objekte `object1` und `object2` und es seien jeweils vier weitere Objekte festgelegt, die Spezialisierungen der Typen von `object1` und `object2` entsprechen. Dabei handelt es sich um die Objekte `object1_1`,...,`object1_4`, sowie um die Objekte `object2_1`,...,`object2_4`. Alle Objekte, die Ableitungen oder Spezialisierungen von `object1` darstellen, sollen in Ausschlussbeziehungen zu allen von `object2` abgeleiteten Objekten stehen.

1. Anzahl der Regeln ohne Typbeziehungen zwischen den Objekten:
   Für jedes der vier von `object1` abgeleiteten Objekte und für `object1` selbst müssen jeweils fünf `excludes`-Regeln definiert werden, deren Zielobjekte den Objekten des `object2`-Strangs entsprechen, also insgesamt 5*5=25 Regeln.

2. Anzahl der Regeln mit Typbeziehungen zwischen den Objekten:
   Es müssen fünf Regeln definiert werden, um die selbe Vollständigkeit wie im ersten Fall zu erlangen. Dabei handelt es sich um die Regeln `excludes(object1,object2)`, `excludes(object1_1,object2)`, `excludes(object1_2,object2)`, `excludes(object1_3,object2)` und `excludes(object1_4,object2)`.

Im Beispiel ergibt sich eine kombinatorische Explosion der Regeln, sofern keine Typbeziehungen berücksichtigt werden. Um die Übersicht über die Anzahl der Regeln für die semantischen Beziehungen zu behalten, scheint es unumgänglich, Objekte zu klassifizieren und Zielobjekte wie in Definition 4.3 zu substituieren. Dies ist insbesondere wichtig, wenn sich eine semantische Beziehung ändert oder wenn eine neue semantische Beziehung etabliert wird, die für alle Objekte eines bestimmten Typs gilt.

In diesem Kapitel wurde die Verwendung einer Ontologie zur Organisation der Objekte einer Domäne eingeführt, in der Typbeziehungen zwischen den Objekten etabliert

werden. Im Folgenden muss näher auf die Rolle des Vererbungsprinzips für die Objekte dieser Ontologie eingegangen werden.

4.2.1. Ontologien und Vererbung

Wenn von Ontologien und Typhierarchien gesprochen wird, stellt sich die Frage, ob diese Ontologien lediglich dem Zweck der Klassifizierung dienen oder auch das Konzept der Vererbung mit diesem Ansatz verfolgt wird. Dieses Prinzip der Vererbung wird häufig von objektorientierten Programmiersprachen benutzt, um Eigenschaften eines Objekts auf ein anderes zu Übertragen und um Code wiederzuverwenden. Diese vererbten Eigenschaften finden ihre Entsprechung in den semantischen Beziehungen. Durch die Vererbung von Regeln müssen diese Regeln nicht für jedes Objekt einer Ontologie neu definiert werden, sondern es wird ausgenutzt, dass die Objekte, die in einer Typhierarchie stehen, die selben semantischen Beziehungen aufweisen. Der Vorteil liegt dabei in einer geringeren Speicheranforderung sowie der Möglichkeit, eine Regel an einer Stelle zu ändern und diese geänderte Regel implizit auf alle Untertypen propagieren zu können. Existiert keine Regelvererbung, so wird für jedes Objekt eine eigene Regelmenge festgelegt. Dies scheint unter dem Aspekt der Flexibilität ein Vorteil zu sein. Es muss untersucht werden, ob die Flexibilität durch vollständig eigenständige Regelmengen der Objekte im Zusammenhang mit der Semantik eines Objekts eine wünschenswerte oder notwendige Eigenschaft ist.

Klassifizierungshierarchien sind meist Spezialisierungshierarchien, d.h. je tiefer ein Objekt in der Hierarchie angeordnet ist, desto speziellere Eigenschaften besitzt es. Dies drückt sich normalerweise darin aus, dass die semantischen Beziehungen der Obertypen geerbt und durch weitere Regeln ergänzt werden. Ein Objekt kann jedoch ein Untertyp eines bestimmten Obertyps sein und trotzdem vollständig andere semantische Beziehungen fordern als der Obertyp vorgibt. Dies wird deutlich, wenn man sich eine Medikamentenontologie vorstellt. Jedes Medikament, unabhängig von dessen Typzuordnung, kann eigene Nebenwirkungen zu anderen Medikamenten besitzen, unabhängig davon, welche Nebenwirkungen der Obertyp besitzt. Um dieser Situation gerecht zu werden, und trotzdem die Vorteile der Vererbung beibehalten zu können, müssen folgende Mechanismen bereitgestellt werden:

- Möglichkeit der Aufhebung geerbter Regeln
- Redefinition geerbter Regeln

Der erste Punkt geht davon aus, dass in den geerbten semantischen Beziehungen Regeln existieren, mit deren Zielobjekten das erbende Objekt in keinerlei Beziehung steht. Solche Regeln dürfen bei Anwendung des Objekts nicht zur Auswertung kommen, da sonst unnötigerweise semantische Konflikte verursacht werden können und die Benutzbarkeit des Objekts eingeschränkt wird. Es ist vorstellbar, dass in der Definition eines Objekts eine Art "Negativ-Liste" mit potentiellen Zielobjekten geerbter Regeln festgelegt wird. Jede geerbte Regel, deren Zielobjekt in dieser Liste enthalten ist, wird dann aus der

Regelmenge des Objekts entfernt.
Die zweite Forderung lehnt sich an das Konzept der Redefinition bzw. des Überschreibens von Methoden bei objektorientierten Programmiersprachen an. Die Redefinition einer semantischen Beziehung erfolgt, indem eine Regel innerhalb des Objekts festgelegt wird, deren Zielobjekt bereits Ziel einer geerbten Regel ist. In diesem Fall wird die geerbte Regel aus der Regelmenge entfernt.
Die Einführung dieser Möglichkeiten stellt einen Kompromiss zwischen dem Regelkorsett, das die reine Vererbung von semantischen Regeln bedeutet, sowie dem Overhead durch die Neudefinition sämtlicher Regeln innerhalb eines Objekts dar und ermöglicht die Beibehaltung der Vorteile beider Konzepte, die zentrale Definition allgemeiner Regeln bei gleichzeitiger Erhaltung der Flexibilität für einzelne Objekte.
Die Priorität durch die Einführung dieser Konzepte liegt klar auf der Regelvererbung, da davon ausgegangen werden kann, dass ein Abweichen der semantischen Beziehungen von den geerbten Regeln eher einen Ausnahmecharakter aufweist. Dies schließt natürlich nicht aus, dass dies in einigen Anwendungsgebieten trotzdem häufiger vorkommen kann.

Geht man davon aus, dass einige Objekte gegenüber ihren Obertypen stark in ihren semantischen Beziehungen abweichen, so ergibt sich die Überlegung, ob das Prinzip der Substitution von Zielobjekten durch Vorkommen von Untertypen ohne Ausnahme als allgemeingültig betrachtet werden kann. Zur Verdeutlichung kann ein Beispiel dienen:

Ein bestimmtes Medikament `MedA` legt die Ausschlussbeziehung `excludes(MedA, Schmerzmittel)` fest. Diese Ausschlussbeziehung trifft tatsächlich für fast alle Schmerzmittel zu. Es existiert dennoch eine Ausnahme.

Eine Möglichkeit, dieser Situation zu begegnen, ist die Festlegung einer Regel für jedes konkrete Schmerzmittel, für das die Ausschlussbeziehung gilt. Der Vorteil der Substituierbarkeit von Untertypen wird dabei aufgegeben. Eine weitere Möglichkeit, Ausnahmen gerecht zu werden, ist die Definition einer Liste mit Objekten innerhalb der Regel, die zwar Untertypen des Zielobjekts darstellen, aber die nicht zur Erfüllung der Regel führen sollen. Wiederum wird damit ein Kompromiss erreicht, der den Vorteil der Definition einer Regel mit möglichst allgemeinem Zielobjekt mit der Flexibilität verbindet, einzelne konkrete Untertypen von der Regel auszuschließen.

In der weiteren Ausarbeitung wird die Möglichkeit der Aufhebung geerbter Regeln und der Ausschluss bestimmter Untertypen von der Erfüllung einer Regel nicht weiter verfolgt, um nicht unnötig durch diese Details von den weiteren Untersuchungen abzulenken. Im Gegensatz dazu wird das Konzept der Redefinition geerbter Regeln in den weiteren Ausführungen vorausgesetzt.

Nach diesen Überlegungen ist eine weitere Frage im Zusammenhang mit dem Ontologieansatz offen, nämlich die Frage, wie eine solche Ontologie organisiert werden kann. Betrachtungen zu dieser Frage finden sich im folgenden Kapitel.

4.3. Organisation der Objekt-Ontologie

Der Aufbau einer globalen Ontologie, die alle für eine Domäne ermittelten Objekte enthält, wurde im vorigen Kapitel eingeführt. Dieser domänenorientierte Standpunkt ist jedoch keineswegs die einzige Möglichkeit, wie Ontologien und Objekte organisiert werden können. So ist es denkbar, Objekte nicht für eine komplette Domäne zu ermitteln und festzulegen, sondern sie jeweils für eine konkrete Anwendungssituation, also einen bestimmten Prozess, zu definieren und somit eine prozessorientierte Sichtweise zu verfolgen. Jedem Prozess kann dann eine Ontologie zugewiesen werden, in der die Objekte, die für diesen Prozess benötigt werden, hierarchisch organisiert sind. Ein Ansatz zur Bewertung dieser Organisationsform zeigt sich in folgenden Vor- und Nachteilen.
Vorteile:

- Die Anzahl verschiedener Objekte, die für einen bestimmten Prozess benötigt werden, ist normalerweise wesentlich geringer, als die Anzahl an Objekten in der kompletten Domäne.
  $\Longrightarrow$ Die Typbeziehungen zwischen den Objekten sind übersichtlicher und schneller auswertbar.
- Die Regeln, die die semantischen Beziehungen zwischen den Objekten festlegen, können genau auf die Erfordernisse des jeweiligen Prozesses zugeschnitten sein.
  $\Longrightarrow$ Die Regelmenge bleibt kleiner, was sich positiv auf die Laufzeit der Korrektheitstests auswirkt.

Nachteile durch die Dezentralisierung der Objekt- und Regeldefinitionen:

- Der Ansatz orientiert sich sehr stark an der Struktur des jeweiligen Prozesses. Dadurch besteht die Gefahr, dass der Blickwinkel bei der Definition der semantischen Beziehungen vom Objekt weggerückt wird, hin zu einer pragmatischen Festlegung anhand der Prozessstruktur, ohne jedoch der tatsächlich vorhandenen Beziehungen gerecht zu werden. Fehlende semantische Beziehungen können insbesondere bei Änderungsoperationen zu unerwünschten Ergebnissen führen.
- Die semantische Information muss für jeden Prozess erneut erstellt werden. Eine Konsequenz davon kann sein, dass die Qualität der semantischen Information leidet. Durch mehrfache Definition der selben semantischen Information in verschiedenen Prozessen wächst beispielsweise die Gefahr, Fehler zu machen und so unkorrekte Regeln oder Typbeziehungen zu formulieren.
- Die Gesamtheit der semantischen Information der Domäne kann nicht auf einfache Weise ermittelt werden. Zum Beispiel können verschiedene Namen für eigentlich identische Objekte in unterschiedlichen Ontologien verwendet werden, die Typeinordnung von Objekten in den Ontologien kann abweichen oder in verschiedenen Prozessen können Regeln festgelegt sein, die einander widersprechen.

- Änderungen an Beziehungen müssen in vielen Prozessen nachgezogen werden. In verschiedenen Prozessen können jedoch verschiedene Namen für die gleichen Objekte verwendet werden, so dass es erschwert wird, die zu ändernden Regeln auszumachen. Weiter bleibt unklar, welche Regeln angepasst werden müssen, wenn die Änderung eines globalen Sachverhaltes Objekte betrifft, die für den aktuellen Prozess nicht definiert sind, es jedoch Regeln mit Quell- und Zielobjekt gibt, die dennoch als Untertypen betrachtet werden können.

Diese vielen Nachteile zeigen, dass der zentrale, domänenorientierte Ansatz wesentlich weniger Probleme erzeugt als der dezentrale, prozessorientierte Ansatz und deswegen bevorzugt umgesetzt werden sollte. Insbesondere die objektzentrierte Sichtweise bei der Definition der semantischen Beziehungen in einer Domäne, zu der der zentrale Ansatz angesichts keiner konkret zugrundeliegenden Prozessstrukturen nahezu zwingt, sorgt dafür, dass qualitativ hochwertige semantische Informationen in der Domäne existieren. In der Ausarbeitung wird in den weiteren Untersuchungen deshalb vom domänenorientierten Ansatz ausgegangen.

Abbildung 4.7 zeigt die Darstellung einer Ontologie, wie sie im weiteren verwendet wird. Die Regeln eines Objekts werden beim Objekt dargestellt, das als Quellobjekt der Regel fungiert. Alle Untertypen eines Objekts besitzen dann in ihrer Regelmenge als Folge der Regelvererbung implizit ebenfalls die Regeln der Obertypen, unter Beachtung der Redefinition von Regeln.

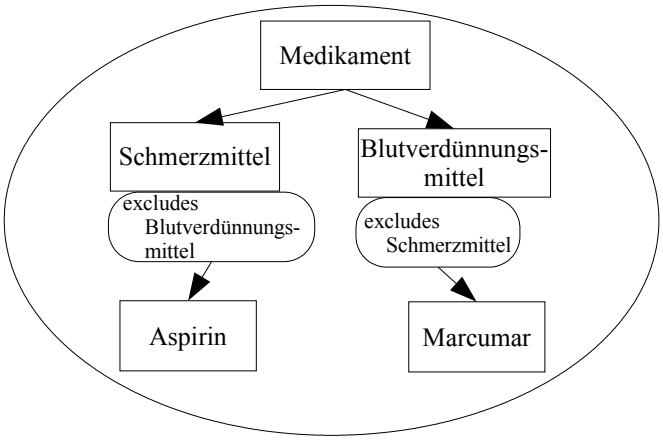


Abbildung 4.7.: Darstellungsweise einer Ontologie

Ob die dargestellte Organisation der Objekte und der semantischen Beziehungen in Form einer Ontologie ausreichend für alle Anwendungssituationen ist, wird im nächsten Kapitel einer näheren Untersuchung unterzogen.

4.4. Kontextabhängige semantische Beziehungen

Die Regeln eines Ontologie-Elementes kommen zur Auswertung, wenn beispielsweise im Zuge einer Änderungsoperation ein Korrektheitstest erforderlich ist. Im bisher verfolgten Ansatz werden immer alle Regeln ausgewertet, die einem Ontologie-Element zugeordnet sind. Die Unterscheidung verschiedener Anwendungssituationen in der Definition der Regelmengen wird nicht berücksichtigt. Abbildung 4.8 verdeutlicht den Zusammenhang zwischen semantischen Beziehungen und unterschiedlichen Anwendungssituationen. Im Beispiel wird auf die bereits verwendete Medikamentenontologie zurückgegriffen. Im Aktivitäten-Repository sind neue Aktivitäten hinzugekommen, die vom neuen Prozess, der eine Medikamentenbestellung modelliert, verwendet werden. Der Behandlungsprozess zeigt wie erwartet einen semantischen Konflikt an, da einem Patienten sowohl `Aspirin` als auch `Marcumar` verabreicht werden soll. Im Bestellprozess tritt jedoch ebenso ein semantischer Konflikt auf, der zum nicht erwünschten Ergebnis führt, dass `Aspirin` und `Marcumar` nicht zusammen bestellt werden können. Bei den zwei in Konflikt stehenden Aktivitäten handelt es sich um die Aktivität `Aspirin bestellen`, die durch ein Objekt des Typs `Aspirin` semantisch beschrieben wird, sowie um die Aktivität `Marcumar bestellen`, der ein Objekt vom Typ `Marcumar` zugeordnet ist.
Am Beispiel wird deutlich, dass verschiedene Anwendungssituationen unterschiedliche semantische Beziehungen erfordern. So ist der Ausschluss zwischen `Schmerzmittel` und `Blutverdünnungsmittel` gerechtfertigt, solange die zwei Medikationen im Kontext des Verabreichens verwendet werden. Werden die Objekte jedoch in einem anderen Kontext verwendet, wie es im Beispiel mit den Bestellaktivitäten der Fall ist, so können die definierten semantischen Beziehungen zu unerwünschten Resultaten führen.

Semantische Beziehungen, die in einem spezifischen Anwendungsfall bestehen, können also für einen anderen zu starke Restriktionen in Bezug auf Ausschlussbeziehungen aufweisen. Ebenso können geforderte Abhängigkeiten in einer Situation notwendig sein, andere Anwendungen werden jedoch dadurch eingeschränkt. Es muss geklärt werden, welche Möglichkeiten zur Lösung dieses Problems existieren. Dazu dienen folgende Strategien als Ausgangspunkt der Überlegungen:

- Existieren unterschiedliche Restriktionen in verschiedenen Anwendungsfällen, so wird die Vereinigung der Restriktionen aller Anwendungssituationen in semantische Beziehungen umgesetzt.
- Statt die vereinigte Menge der Restriktionen zu benutzen, wird die Schnittmenge der semantischen Beziehungen aller Anwendungsfälle ermittelt.

Die erstgenannte Strategie sorgt dafür, dass nach einem erfolgreich absolvierten Korrektheitstest die Prozesse auch tatsächlich semantisch korrekt sind. Beim Korrektheitstest werden insbesondere auch alle Regeln ausgewertet, die für den aktuellen Prozess relevant sind. Die Auswertung sämtlicher Regeln für alle abzudeckenden Situationen kann

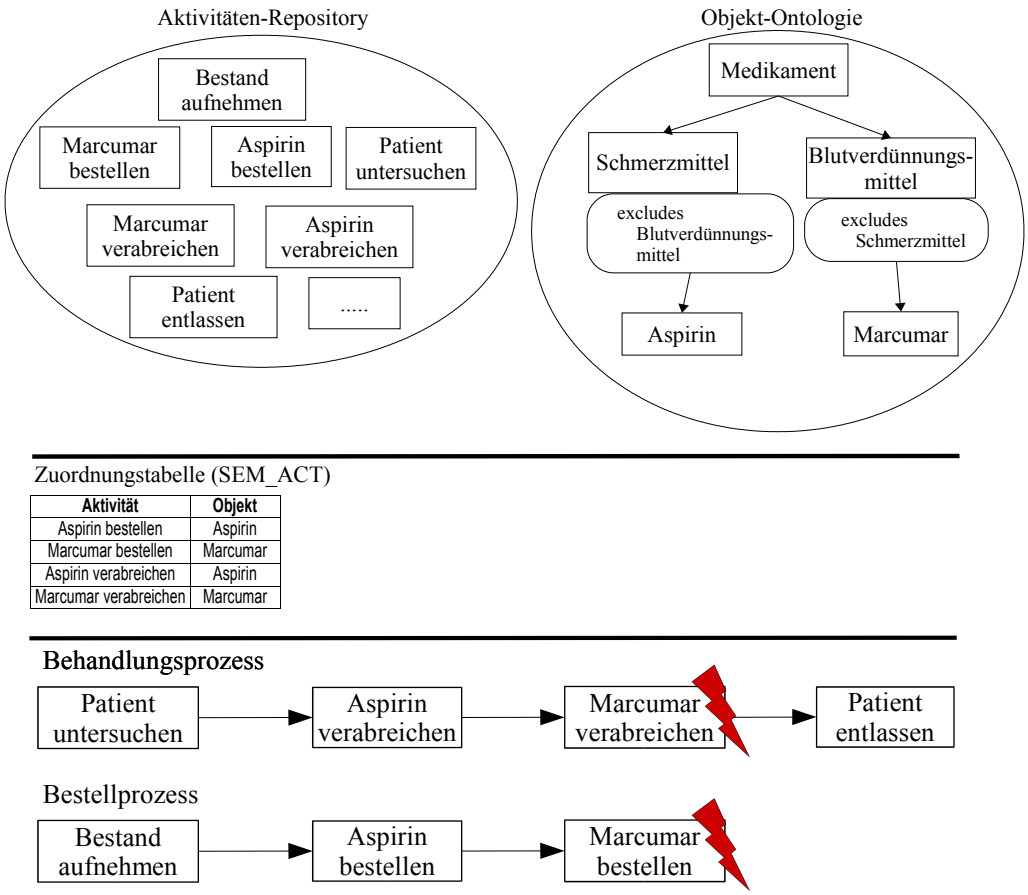


Abbildung 4.8.: Semantische Beziehungen ohne abgegrenzte Anwendungssituationen

jedoch den Effekt haben, dass ein Objekt in beinahe jeder Anwendung einen semantischen Konflikt verursacht. Dies kann man sich beispielsweise so vorstellen, dass eine sehr spezielle Anwendung für ein Objekt eine Abhängigkeit fordert, die nur in einem sehr eng begrenzten Anwendungskontext überhaupt Sinn macht. Damit wäre die Verwendung dieses Objekts in anderen Prozessen blockiert, die andere Anwendungssituationen repräsentieren, in denen das Objekt, das zur Erfüllung der geforderten Abhängigkeit notwendig ist, nicht verwendet wird.
Da mit der Anzahl verschiedener Anwendungsfälle die Anzahl der semantischen Beziehungen stetig steigt, wird die Anwendbarkeit von Objekten immer mehr eingeschränkt, bis sie faktisch kaum noch nutzbar sind. Zusammengefasst bedeutet diese Strategie eine maximale Garantie der semantischen Korrektheit von Prozessen, schränkt jedoch die Objekte bis zur Unbenutzbarkeit ein, so dass in einer realen Umsetzung diese Möglichkeit nicht in Betracht gezogen werden kann.

Die zweite Möglichkeit schränkt die Verwendbarkeit der Objekte in den verschiedenen Anwendungskontexten hingegen nicht ein. Dadurch, dass aus jedem Kontext nur die se-

mantischen Beziehungen in die Ontologie übernommen werden, die auch für alle anderen Anwendungsfälle zutreffen, werden potentiell Restriktionen einzelner Anwendungssituationen verworfen. Der mögliche Einsatzbereich der definierten Ontologie-Elemente kann als Folge davon sogar eine Erweiterung erfahren. Dies wird damit erkauft, dass einzelne Prozesse beim Korrektheitstest zwar keine semantischen Konflikte aufweisen und somit die Anwendung der Aktivitäten in der vorgegebenen Prozessstruktur möglich scheint, in Wirklichkeit unter Berücksichtigung der eigentlich real existierenden Restriktionen die Prozesse jedoch nicht korrekt sind. Dies kann fatale Folgen haben. Betrachtet man das Beispiel aus Abbildung 4.8 abermals unter dem Gesichtspunkt, dass lediglich die Beziehungen unter der Vorgabe des kleinsten gemeinsamen Nenners der beiden Anwendungssituationen `Behandlungsprozess` und `Bestellprozess` berücksichtigt werden, so würde sich dies folgendermaßen darstellen:

- Die Regelmengen der Objekte in der dargestellten Ontologie müssen unter diesen Voraussetzungen als leere Mengen betrachtet werden, da es für den Kontext der Medikamentenbestellung keinerlei Restriktionen gibt und somit dieser Anwendungsfall die Regelmengen bestimmt.
- Beide Prozesse sind nun semantisch korrekt. Betrachtet man diesen Sachverhalt für die verschiedenen Kontexte, so ergibt sich folgendes Bild:
  - Bestellprozess: der semantische Konflikt aus Abbildung 4.8 besteht nicht mehr, was zum erwarteten und gewünschten Ergebnis führt, dass beide Medikamente gleichzeitig bestellt werden können.
  - Behandlungsprozess: auch hier tritt der semantische Konflikt nicht mehr auf. Der Prozess kann also auf die in Abbildung 4.8 dargestellte Weise ausgeführt werden. Die Medikamentenunverträglichkeit besteht aber in der Realität weiterhin. Somit kann das Aufweichen der Restriktionen gesundheitliche Konsequenzen für den Patienten haben.

Wie am Beispiel deutlich wird, bestimmt derjenige Anwendungskontext die Restriktivität der semantischen Beziehungen, der die wenigsten semantischen Regeln besitzt. Im Extremfall ist die Regelmenge also die leere Menge, wenn es einen Prozess gibt, der für ein Objekt keinerlei Restriktionen oder nur solche Restriktionen fordert, die in keinem anderen Kontext bestehen. Dies ist selbst dann der Fall, wenn alle anderen Prozesse sehr starke Restriktionen benötigen, um die semantische Korrektheit zu garantieren. Damit ist auch die zweite Strategie für den realen Einsatz nicht geeignet.

Abgesehen von den aufgezeigten Nachteilen haben beide Ansätze das Problem, dass die Anwendungssituationen, die eintreten können, samt ihrer notwendigen Restriktionen, bereits zur Designzeit der Ontologie bekannt sein müssen. Sollten bei Verwendung der ersten Strategie im Lauf der Zeit weitere Restriktionen neuer Anwendungssituationen hinzukommen, so könnte dies dazu führen, dass bisher als semantisch korrekt eingestufte Prozesse bei einer Neuauswertung nicht mehr semantisch korrekt sind, wenn die neuen Restriktionen mit berücksichtigt werden. Tatsächlich könnte das Phänomen auftreten, dass mit der Zeit immer weniger Prozesse korrekt sind und immer mehr Inkonsistenzen

in der Bewertung der Korrektheit der Prozesse auftreten.
Der umgekehrte Fall spielt sich bei der zweiten Strategie ab. Entstehen in der Domäne neue Anwendungen, die weniger Restriktionen fordern, als der bisher kleinste gemeinsame Nenner aller Prozesse vorgibt, so werden bestehende semantische Beziehungen entsprechend des neuen Minimums abgebaut. Über die Zeit gesehen werden also immer weniger Restriktionen für die Anwendung von Objekten bestehen und es werden immer mehr Prozesse als semantisch korrekt eingestuft.
Dieses Verhalten einer nicht stabilen Regelmenge kann zu großer Konfusion führen, und lässt der Bewertung der semantischen Korrektheit der Prozesse nahezu eine Art von Beliebigkeit zukommen.

Da die beiden Strategien nicht den Anforderungen genügen, die eine sinnvolle Umsetzung der Semantik für Prozesse mit sich bringt, muss ein Konzept entwickelt werden, das eben diese Anforderungen erfüllt. Die Anforderungen können folgendermaßen formuliert werden:

- Berücksichtigung des Anwendungskontexts für die Bestimmung der jeweils gültigen Regelmenge.
- Stabilität der Regelmengen in Bezug auf verschiedene Anwendungskontexte. Das bedeutet nicht, dass sich die Regelmenge für eine Anwendungssituation nicht ändern darf, sondern dass sich Änderungen an den Regeln eines Anwendungsfalls nicht auf die Regeln der anderen Anwendungsfälle auswirken dürfen.

Um ein Konzept entwickeln zu können, das die genannten Anforderungen erfüllt, muss zunächst untersucht werden, wie verschiedene Anwendungssituationen voneinander unterschieden werden können und was den Anwendungskontext für ein Objekt ausmacht.

4.4.1. Operationen

Die natürliche Sprache ist die Basis, anhand derer die Objekte und Aktivitäten einer Domäne beschrieben werden. Deshalb ist es naheliegend, die Strukturen der natürlichen Sprache in Hinblick auf die Beschreibung von Anwendungssituationen eines Ontologie-Elements zu untersuchen. Als Startpunkt wird erneut das Beispiel aus Abbildung 4.8 bemüht. Von Interesse sind insbesondere die Aktivitäten `Aspirin verabreichen`, `Marcumar verabreichen`, `Aspirin bestellen` und `Marcumar bestellen`. Wie bereits besprochen, macht die Ausschlussbeziehung zwischen `Schmerzmittel` und `Blutverdünnungsmittel` Sinn in den Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen`, jedoch nicht in den zwei Bestellaktivitäten. Die textuelle Beschreibung der Aktivitäten gibt dabei erste Hinweise auf den Unterschied zwischen den verschiedenen Aktivitäten. So unterscheiden sich die zwei Aktivitäten der Medikamentengabe und die Bestellaktivitäten in ihrer textuellen Beschreibung im Verb. Einmal ist dieses Verb ”verabreichen”, bei den anderen Aktivitäten ”bestellen”. Offenbar ist dieser Unterschied im Beispiel bereits der Entscheidende für die Beurteilung, ob

die definierte Ausschlussbeziehung sinnvoll ist oder nicht. Bettet man die textuelle Beschreibung in einen vollständigen Satz der natürlichen Sprache ein, so zeigt sich, dass es sich bei diesem Verb um das Prädikat des jeweiligen Satzes handelt. Dem Prädikat kommt innerhalb eines Satzes eine zentrale Bedeutung zu, bestimmt das Prädikat doch die Grundaussage des Satzes. Zur Verdeutlichung sollen folgende Beispiele dienen:

- Der Lagerist bestellt Aspirin.
  $\Longrightarrow$ Das Prädikat legt fest:
  - die Tätigkeit, die der Lagerist durchführt: bestellen
  - den Gegenstand der Tätigkeit des Lagerists: Aspirin
- Die Krankenschwester verabreicht Aspirin.
  $\Longrightarrow$ Das Prädikat legt fest:
  - die Tätigkeit, die die Krankenschwester durchführt: verabreichen
  - den Gegenstand der Tätigkeit der Krankenschwester: Aspirin

Wie an den Beispielen ersichtlich ist, legt das Prädikat fest, welche Tätigkeit in einem Satz beschrieben wird, und übertragen auf Prozesse, was sich hinter der tatsächlich auszuführenden Tätigkeit einer Aktivität verbirgt. Im Folgenden wird der Begriff Operation synonym für den Begriff Tätigkeit verwendet, d.h. das Prädikat legt die Operation fest, die auf dem Objekt ausgeführt wird. Verschiedene Operationen bedeuten einen unterschiedlichen Kontext für den Gegenstand der Tätigkeit. Übertragen auf die Aktivitäten von Prozessen bedeutet dies einen unterschiedlichen Kontext für die Objekte, die die semantische Beschreibung der Aktivitäten darstellen.

Für die Bestimmung der auszuwertenden semantischen Beziehungen eines Eintrages der semantischen Aktivitätenbeschreibung muss die Operation auf dem zugeordneten Objekt miteinbezogen werden, d.h. die semantischen Beziehungen und damit die Regeln beziehen sich nicht nur auf das Objekt allein, sondern auf die Verwendung des Objektes zusammen mit einer bestimmten Operation. Das Beispiel aus Abbildung 4.8 kann nun wie in Abbildung 4.9 erweitert werden. Die semantische Beschreibung der Aktivitäten benötigt außer der Zuordnung von Objekten jeweils eine Assoziation mit einer Operation, welche den Kontext für die Auswahl der gültigen Regeln des Objektes festlegt. Für den Kontext `verabreichen` werden wie bisher die Ausschlussbeziehungen `excludes(Schmerzmittel,Blutverdünnungsmittel)` und die Umkehrregel definiert, für den Operation-Kontext `bestellen` werden hingegen keinerlei semantische Beziehungen festgelegt. Diese Differenzierung der semantischen Beziehungen in Hinblick auf den Kontext führt dazu, dass beide Prozesse semantisch korrekt modelliert werden können und zudem zum erwarteten Ergebnis führen. Im Behandlungsprozess tritt dabei wie gewünscht der semantische Konflikt auf, wenn sowohl Aspirin als auch Marcumar verabreicht werden. Der Bestellprozess enthält bei Berücksichtigung des Operations-Kontexts hingegen keinen semantischen Konflikt.

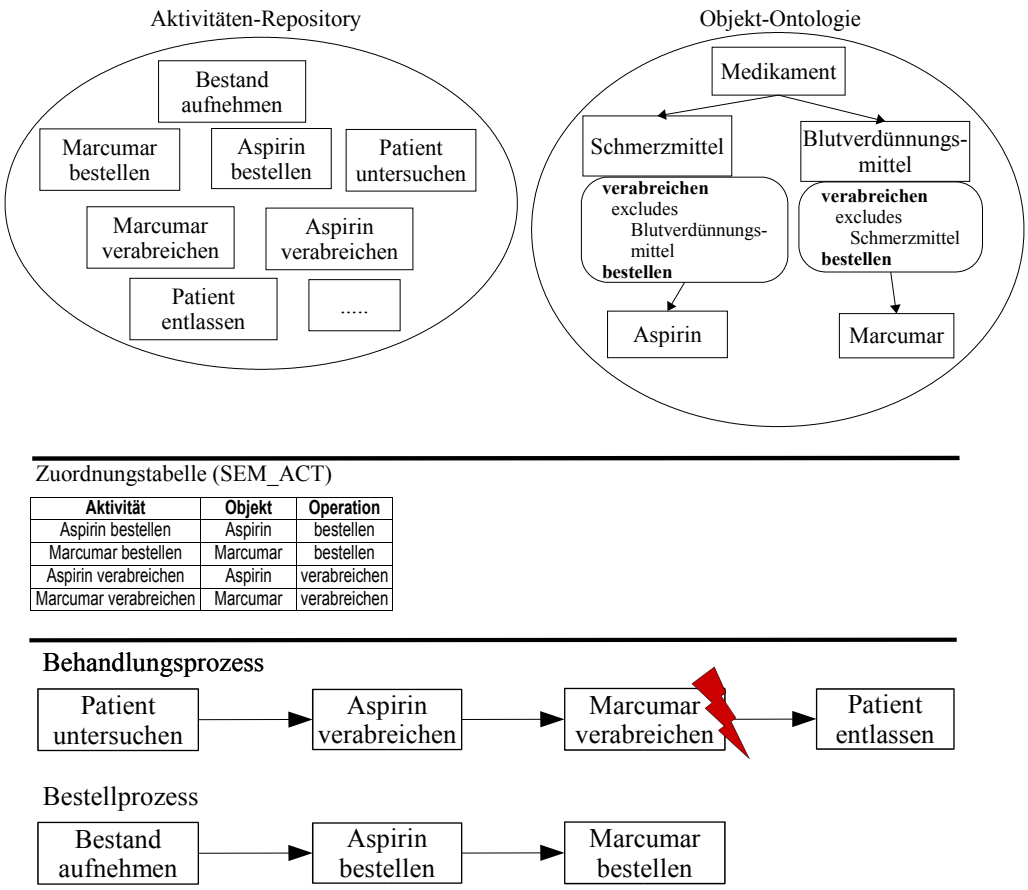


Abbildung 4.9.: Semantische Beziehungen: Berücksichtigung der Operation

Im Beispiel aus Abbildung 4.9 werden die Operationen als zusätzliche Attribute der Objektdefinition verwendet. Es ist sehr wahrscheinlich, dass eine Operation in Kombination mit verschiedenen Objekten Verwendung findet, wobei die Operation für verschiedene Objekte unterschiedliche semantische Beziehungen festlegen kann, aber auch allgemeine Beziehungen, die für viele verschiedene Objekte im Zusammenhang mit dieser Operation gelten. Die Operation `bestellen` legt z.B. vermutlich für die meisten Objekttypen keine Restriktionen fest. Eine Operation kann demnach isoliert vom Objekt betrachtet werden. Aus dieser Perspektive erscheint es sinnvoll, Überlegungen hinsichtlich der Organisation der Operationen und ihrer festgelegten semantischen Beziehungen anzustellen. Grundlegend ergeben sich dabei folgende Möglichkeiten:

- Die Operationen werden wie in Abbildung 4.9 lediglich als Attribute der Objekte verwendet, um unterschiedliche Anwendungssituationen des Objekts mit verschiedenen semantischen Beziehungen in Form von Regeln zu ermöglichen.

- Neben der Objekt-Ontologie existiert parallel eine Operation-Ontologie. Die Operationen können unabhängig von den Objekten beschrieben werden, beispielsweise um zentral semantische Beziehungen zu modellieren, die primär von der verwendeten Operation abhängen. Trotzdem können für bestimmte Objekt-Operation-Konstellationen spezifische Regeln definiert werden. Dazu existiert innerhalb eines Elements der Operation-Ontologie die Möglichkeit, mittels eines domain-Attributs bestimmte Regeln festzusetzen. Wird die Operation in der semantischen Aktivitätenbeschreibung zusammen mit dem im domain-Attribut festgelegten Objekt oder einem Untertyp davon verwendet, so besitzen diese Regeln Gültigkeit. Der Wertebereich dieses Attributs entspricht also den Objekten der Objekt-Ontologie.

Der erste Ansatz stellt das Objekt in den Mittelpunkt der Definition der semantischen Beziehungen. Das Problem bei diesem Ansatz besteht darin, dass semantische Beziehungen, die hauptsächlich durch die Operation beeinflusst werden, in vielen verschiedenen Objekten erneut definiert werden müssen. Somit entstehen Redundanzen im Regelwerk, mit allen damit verbundenen Nachteilen.

Die Verwendung zweier Ontologien, einer Objekt- und einer Operation-Ontologie, misst den Operationen den selben Stellenwert zu wie den Objekten. Semantische Beziehungen können aus der Sicht der Objekte auf der Ebene der Objekt-Ontologie festgelegt werden, oder aus der Sicht der Operationen in der Operation-Ontologie. Der Grundgedanke dabei ist, dass allgemeine Regeln zwischen Objekten in der Objekt-Ontologie erfasst werden. Für Anwendungssituationen, die unter dem Blickwinkel einer bestimmten Operation von diesen allgemeinen Regeln abweichen, können in der Operation-Ontologie semantische Beziehungen festgelegt werden. Ebenso können dort Standardregeln modelliert werden, die die gültige Regelmenge für die Operation darstellen, wenn innerhalb der Definition der Operation kein domain-Attribut existiert, das als Wert das in der semantischen Aktivitätenbeschreibung aktuell mit der Operation verknüpfte Objekt besitzt. Dies ist vorteilhaft, da ein Satz an Regeln für einen durch eine Operation bestimmten Anwendungskontext nur einmal zentral formuliert wird. Ein weiterer Vorteil der Strategie mit getrennten Ontologien ist, dass genauso wie für die Objekte eine Klassifizierungshierarchie der Operationen aufgebaut werden kann. Diese besitzt den bereits behandelten Vorteil der Vererbung von semantischen Beziehungen und der daraus resultierenden zentralen Definition von Regeln.

Der zweite Ansatz bietet eine höhere Flexibilität, als es die Berücksichtigung der Operationen in Form von Attributen der Objekte erlaubt, und wird dem Stellenwert gerecht, den die Operationen für die Semantik einer Aktivität darstellen. Nicht zuletzt der Vorteil der Klassifizierungsmöglichkeit der Operationen durch die separate Ontologie zeigt, dass diese Strategie die zu bevorzugende Alternative darstellt. In der weiteren Ausarbeitung wird demzufolge diese Möglichkeit verfolgt. Abbildung 4.10 zeigt den Aufbau der beiden Ontologien an der aus Abbildung 4.9 bereits bekannten Anwendung.

Neben der Zuordnung eines Objekts kann der semantischen Aktivitätenbeschreibung also zusätzlich eine Operation zugewiesen werden. Bei der Bestimmung der Regelmenge für einen Eintrag der semantischen Aktivitätenbeschreibung gilt dann folgendes Prinzip:

1. Für ein zugewiesenes Objekt ist keine Operation festgelegt:
   Die Objekt-Ontologie wird ausgewertet, d.h. die semantischen Beziehungen des Objekts aus der Objekt-Ontologie gelten für diesen Anwendungsfall.
2. Mit dem zugewiesenen Objekt ist eine Operation assoziiert:
   In diesem Fall wird die Objekt-Ontologie nicht ausgewertet, sondern es kommen die Regeln zum Zug, die in der Operation-Ontologie für die jeweilige Operation hinterlegt sind. Gibt es dort ein domain-Attribut, dessen Wert dem in der Zuordnung genannten Objekt oder einem Obertyp entspricht, so werden die dieser domain zugeordneten Regeln ausgewertet. Andernfalls gilt die Standardregelmenge für die Operation.

Die Überlegungen zur Aufteilung in verschiedene Ontologien und zum zugrundeliegenden Auswertungs-Prinzip dieser Ontologien zeigen, dass den semantischen Beziehungen der Operation-Ontologie mehr die Bedeutung von Ausnahmesituationen zukommen. Es existiert also eine Hierarchie zwischen den Ontologien.
Diese hierarchische Aufteilung entspricht eher dem Charakter einer Empfehlung als einer festen Vorgabe. Letztlich bleibt dem Designer beim Entwurf überlassen, wie die unterschiedlichen Ontologien genutzt werden sollen.

4.4.2. Subjekte und Targets

Es stellt sich die Frage, ob sämtliche Anwendungssituationen bereits in genügender Weise unterstützt werden, so dass keine unerwarteten oder unerwünschten semantischen Konflikte mehr entstehen. Die Antwort liefert das Beispiel, das in Abbildung 4.11 visualisiert ist.
Dargestellt ist der schon öfter bemühte Behandlungsprozess. Die Aktivitäten `Aspirin verabreichen` und `Marcumar verabreichen` wurden durch die Aktivitäten `H. Meier Aspirin verabreichen` und `H. Müller Marcumar verabreichen` ersetzt, also der Prozess dahingehend abgeändert, dass zwei verschiedene Personen in dem Prozess mit Medikationen versorgt werden. Der Prozess enthält noch immer einen semantischen Konflikt, obwohl die unverträglichen Medikamente verschiedenen Patienten gegeben werden. Das Ergebnis ist, dass der modellierte Prozess nicht wie gewünscht umgesetzt werden kann. Es muss also weiter untersucht werden, wie die semantische Information in den Aktivitäten erweitert werden muss, um auch solche Anwendungssituationen sinnvoll umsetzen zu können.

Ein weiterer Blick auf die Grammatik eines Satzes ist als Grundlage der weiteren Diskussion lohnenswert. Folgender Satz soll dazu näher betrachtet werden:

"Herrn Meier wird Aspirin verabreicht."

Damit Sätze auf die semantische Aktivitätenbeschreibung übertragbar sind, müssen sie bestimmten Anforderungen genügen. Der als Objekt der semantischen Aktivitätenbeschreibung identifizierte Satzbestandteil muss das Akkusativ-Objekt des Satzes darstel-

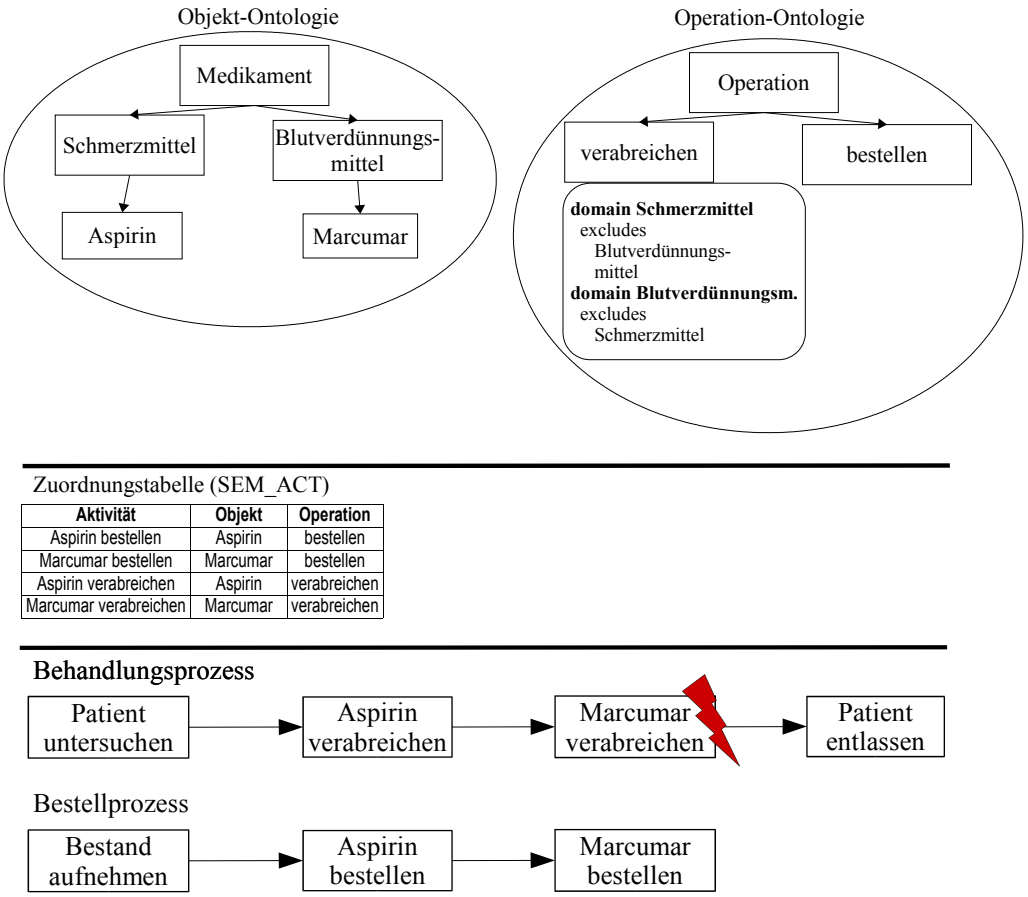


Abbildung 4.10.: Semantische Beziehungen: Objekt- und Operation-Ontologie

len. Dies ist in obigem Satz nicht der Fall. Aspirin repräsentiert dort das passive Subjekt des Satzes. Existiert in einem Satz kein aktives Subjekt, so kann dieser Satz mittels eines neutralen aktiven Subjekts, beispielsweise dem Subjekt ”jemand”, umstrukturiert werden. Damit kann obiger Satz ”Herrn Meier wird Aspirin verabreicht” in den Satz ”Jemand verabreicht Herrn Meier Aspirin” transformiert werden. Aspirin ist nun das Akkusativ-Objekt des Satzes, was obiger Forderung entspricht.
Folgende Satzglieder werden bisher für die Semantikbeschreibung von Aktivitäten berücksichtigt:

- Das Akkusativ-Objekt des Satzes (”Aspirin”): wird im Zuge der semantischen Aktivitätenbeschreibung als Objekt mit Aktivitäten assoziiert
- Das Prädikat des Satzes (”verabreichen”): die Operation die das Objekt in einen spezifischen Kontext setzt

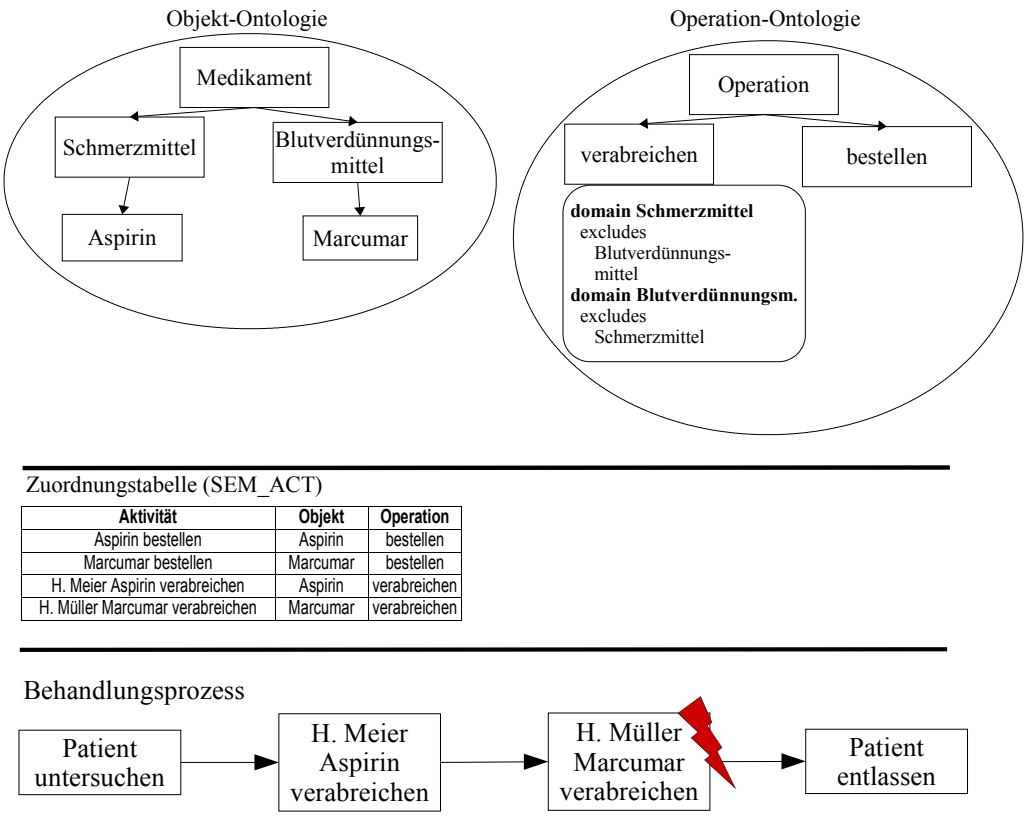


Abbildung 4.11.: Semantischer Konflikt: fehlende Berücksichtigung des Ziels der Aktivität

Nicht berücksichtigt für die Umschreibung des Kontexts wird das Dativ-Objekt (”Herrn Meier”), sowie das Subjekt des Satzes. Wie aus dem Beispiel aus Abbildung 4.11 deutlich hervorgeht, ist eine Entsprechung des Dativ-Objekts in der semantischen Beschreibung der Aktivitäten notwendig, um die Anwendungssituationen eines Objekts weiter differenzieren zu können.

Es bleibt zu untersuchen, ob auch die Berücksichtigung des Subjekts eines natürlichsprachlichen Satzes zur Unterscheidung unterschiedlicher Anwendungskontexte des Objekts notwendig ist. Angenommen der Chefarzt einer Station unterliegt keinerlei Restriktionen bezüglich seiner Entscheidungen, d.h. er kann auch die Entscheidung treffen, dass einem Patient gleichzeitig zwei eigentlich unverträgliche Medikamente verabreicht werden. Kein anderer Arzt der Station im Krankenhaus hat die Befugnis, eine solche Entscheidung zu treffen. Das bedeutet, dass bei der Anwendung beispielsweise des Objekts `Marcumar` in einer Aktivität unterschieden werden muss, ob `Marcumar` vom Chefarzt oder einem gewöhnlichen Arzt verabreicht wird. Je nachdem gelten verschiedene semantische Beziehungen.

Für die Berücksichtigung des Dativ-Objekts und des Subjekts gelten die selben Überlegungen, wie diese bereits für das Prädikat eines Satzes angestellt wurden. Sowohl das Subjekt als auch das Dativ-Objekt sind jedoch nicht, wie dies beim Prädikat der Fall ist, primär an das Objekt gebunden, sondern es ist sinnvoller sie mit dem Prädikat, d.h. mit der Operation, in Verbindung zu setzen. Die Berücksichtigung des Subjekts und Dativ-Objekts ermöglicht eine gegenüber der Operation weitere Spezialisierung des Anwendungskontexts. Durch die Bindung des Subjekts und Dativ-Objekts an die Operation entsteht eine klar strukturierte Spezialisierungshierarchie, die die Flexibilität einfacher Satzkonstruktionen mit Subjekt, Prädikat, Akkusativ- und Dativ-Objekt abzubilden vermag.

Das Dativ-Objekt eines Satzes stellt das eigentliche Ziel der Operation dar, d.h., übertragen auf den Kontext PMS, dass die der Aktivität zugeordnete Operation auf dem zugeordneten Objekt dieses Dativ-Objekt zum Ziel hat. Damit findet das Dativ-Objekt eines Satzes eine Entsprechung im Begriff `Target` in der semantischen Aktivitätenbeschreibung. Analog zur Objekt- und Operation-Ontologie existiert eine Target-Ontologie, die die möglichen Targets der Domäne erfasst.
Analog dazu wird das Subjekt eines Satzes in den `Subjekt`-Eintrag der semantischen Aktivitätenbeschreibung abgebildet. Die Subjekte einer Domäne werden in einer Subjekt-Ontologie erfasst, die eine Klassifizierungshierarchie für die Subjekte repräsentiert.

Die Elemente beider Ontologien nehmen Bezug auf Operationen. Dazu können die Regeln jeweils für verschiedene Operationen getrennt formuliert werden, was in äquivalenter Weise zu den Elementen der Operation-Ontologie über ein domain-Element ermöglicht wird. Die Regeln, die nicht einer speziellen domain zugeordnet werden, gelten dann bei der Verwendung des Ontologien-Elements mit allen Operationen, die nicht in spezifischen domain-Elementen erfasst sind.

Zusätzlich kann ein Element der Subjekt- und Target-Ontologie in Kombination mit einem domain-Attribut ein objContext-Attribut festlegen, dessen Wertebereich die Objekte der Objekt-Ontologie umfasst. Die Regeln des Elements werden dadurch in ihrer Gültigkeit auf Anwendungsfälle beschränkt, in denen das Subjekt bzw. Target zusammen mit der im domain-Attribut angegebenen Operation und dem durch das objContext-Attribut angegebenen Objekt verwendet wird.

Um Missverständnissen vorzubeugen, muss nochmals der Zusammenhang zwischen den Ontologien und den semantischen Beziehungen klar dargestellt werden. Gegenstand der semantischen Beziehungen und damit der Regeln sind die Objekte, d.h. auf Objektebene spielen sich die Ausschlussbeziehungen und Abhängigkeiten ab. Eine Regel stellt auch weiterhin eine binäre Relation dar. Das Quellobjekt bezieht sich dabei immer auf das Objekt, das dem selben Eintrag der semantischen Aktivitätenbeschreibung zugeordnet ist, wie das Element, über das die Regelmenge bestimmt wird. Ist dieses Element nicht das Objekt selbst, so handelt es sich um ein Element der Operation-, der Subjekt- oder der Target-Ontologie.
Das domain-Attribut in der Operation-Ontologie schränkt das Quellobjekt der definierten semantischen Beziehungen auf das angegebene Objekt oder Untertypen davon ein.

Semantische Beziehungen in der Operation-Ontologie, die unter keinem domain-Attribut definiert sind, haben potentiell alle Objekte aus $\mathcal{O}$ als Quellobjekt, außer denen, die dem Wert eines domain-Attributs entsprechen. Eine so definierte semantische Beziehung umfasst also eine Menge an semantischen Beziehungen. Das objContext-Attribut in der Subjekt- und Target-Ontologie hat eine ähnliche Bedeutung. Semantische Beziehungen, die im Zusammenhang mit einem objContext-Attribut festgelegt werden, umfassen alle semantischen Beziehungen, deren Quellobjekte dem im objContext-Attribut festgelegten Objekt und dessen Untertypen entsprechen. Die Quellobjekte von Regeln dieser Ontologien, die ohne objContext-Attribut festgelegt werden, umfassen alle Objekte aus $\mathcal{O}$ außer diejenigen, für die ein objContext-Attribut genannt ist. Die unterschiedlichen Ontologien dienen also der Realisierung verschiedener semantischer Beziehungen zwischen Objekten für unterschiedliche Anwendungskontexte. Dabei dienen die Einträge der Semantikbeschreibung der Selektion der korrekten Regelmenge. Aufgrund der Eigenschaft der Operation-, Subjekt- und Target-Ontologien, semantische Beziehungen für spezifische Anwendungssituationen festzulegen, wird im Folgenden von diesen drei Ontologien auch als den Kontext-Ontologien gesprochen.

Abbildung 4.12 zeigt eine Darstellung der Hierarchie zwischen den Ontologien. Im einzelnen ergeben sich für die unterschiedlichen Ontologien folgende Anwendungsszenarien:

- Die Objekt-Ontologie definiert die allgemeinsten Regeln.
- Die Operation-Ontologie definiert Regeln für das Auftreten eines Objektes in Verbindung mit einer bestimmten Operation.
- Die Subjekt-Ontologie legt Regeln für die Verwendung eines Objektes in Verbindung mit einer bestimmten Operation und einem bestimmten Subjekt fest.
- Die Target-Ontologie legt Regeln für die Verwendung eines Objektes in Verbindung mit einer bestimmten Operation und einem bestimmten Target fest.

Bei der Bestimmung der Regelmenge für ein mit einer Aktivität assoziiertes Objekt wird nur die Ontologie verwendet, deren entsprechender ”nicht-leerer” Eintrag in der semantischen Beschreibung der Aktivität dem in der Ontologien-Hierarchie höchsten Spezialisierungsgrad entspricht.

Beispiel: In der Semantikbeschreibung ist sowohl ein Objekt, als auch eine Operation angegeben.
$\Longrightarrow$ Die Regeln aus der Operation-Ontologie werden verwendet, da diese einen höheren Spezialisierungsgrad aufweist, als die Objekt-Ontologie.

Bei der Betrachtung der hierarchisch organisierten Ontologien fällt auf, dass sich die Subjekt- und Target-Ontologie auf der selben hierarchischen Ebene befinden und ihnen damit ein vergleichbarer Spezialisierungsgrad zugeschrieben wird. Dies resultiert daraus, dass sowohl das Subjekt als auch das Target Einfluss auf die tatsächliche Bedeutung der Operation auf dem gegebenen Objekt nehmen, jedoch aus unterschiedlichen Blickwinkeln. Einmal vom Standpunkt des handelnden Subjekts aus gesehen, das andere Mal vom Blickwinkel des Ziels der auf dem assoziierten Objekt durchgeführten Operation. Durch

diese nicht existierende Hierarchie zwischen Subjekt- und Target-Ontologie stellt sich die Frage, welche Ontologie für die Ermittlung der semantischen Beziehungen zuständig ist, wenn folgeder Fall eintritt:

Sowohl ein Subjekt, als auch ein Target sind in der semantischen Beschreibung einer Aktivität angegeben.
Da sich beide Ontologien auf der selben Hierarchieebene befinden, werden die für die aktuelle Anwendungssituation geltenden Regelmengen aus beiden Ontologien zu einer Regelmenge zusammengefasst.

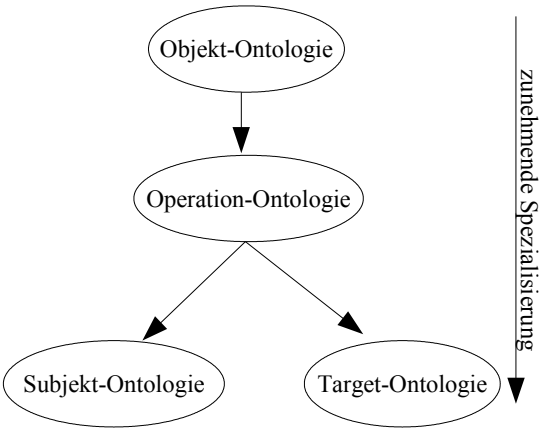


Abbildung 4.12.: Hierarchische Einordnung der Ontologien

Die Konventionen zur Auswertung der Ontologien bei der Bestimmung der Regelmenge sind in nachstehender Tabelle 4.1 zusammengefasst. Die Spaltenbeschriftungen entsprechen den möglichen Einträgen der semantischen Beschreibung einer Aktivität, die Zeilenbeschriftungen den verschiedenen Ontologien. Die Tabelle ist so zu lesen, dass eine Ontologie dann zur Auswertung kommt, wenn in einer Semantikbeschreibung die Einträge existieren, die in der Tabelle angekreuzt sind. Das Symbol ∘ in den Zeilen der Subjekt- und Target-Ontologie bedeutet, dass auch in der Konstellation, bei der Einträge sowohl für das Subjekt als auch für das Target in der Aktivitätenbeschreibung existieren, die Regeln der Subjekt- bzw. Target-Ontologie gelten.

Ontologie/Eintrag semant. Beschreibung	Objekt	Operation	Subjekt	Target
Objekt	X			
Operation	X	X		
Subjekt	X	X	X	∘
Target	X	X	∘	X

Tabelle 4.1.: Auswertungsrichtlinien für die Ontologien

Die eingeführten Kontext-Ontologien setzen das Quellobjekt einer semantischen Beziehung in unterschiedliche Kontexte und ermöglichen die Definition unterschiedlicher semantischer Beziehungen für ein Quellobjekt. Die Kontext-Ontologien beschreiben wie die Objekt-Ontologie die Domäne. Deshalb müssen die Operationen, Subjekte und Targets in der Definition einer Domäne berücksichtigt werden. Definition 4.4 erweitert Definition 3.1 entsprechend.

Definition 4.4 (Domäne - Erweiterte Beschreibung)

Eine Anwendungsdomäne $\mathcal{D}$ wird durch ein Tupel $(\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS)$ semantisch repräsentiert:

- *$\mathcal{O}$ ist die Menge der Objekte, die $\mathcal{D}$ semantisch beschreiben*
- *$\mathcal{OP}$ ist die Menge der Operationen, die für $\mathcal{D}$ festgelegt sind*
- *$\mathcal{S}$ ist die Menge der Subjekte, die für $\mathcal{D}$ festgelegt sind*
- *$\mathcal{T}$ ist die Menge der Targets, die für $\mathcal{D}$ festgelegt sind*
- *$\mathcal{R} := \{SR(s,t) \mid s,t \in \mathcal{O}\}$ ist die Menge der semantischen Beziehungen in $\mathcal{D}$, mit $SR(s,t) := ruleType(s,t\ [, directionTarget])$ mit*
  - *$ruleType \in \{excludes, dependsOn\}$*
  - *s ist Quellobjekt der Semantikregel*
  - *t ist Zielobjekt der Semantikregel*
  - *$directionTarget \in \{pre, post\}$ gibt die relative Position des Zielobjekts an*
- *$SEM\_ACT: \mathcal{N} \rightarrow \mathcal{P}(K)$, $K := \mathcal{O} \cup (\mathcal{O} \times \mathcal{OP}) \cup (\mathcal{O} \times \mathcal{OP} \times \mathcal{S}) \cup (\mathcal{O} \times \mathcal{OP} \times \mathcal{T}) \cup (\mathcal{O} \times \mathcal{OP} \times \mathcal{S} \times \mathcal{T})$ stellt die semantische Aktivitätenbeschreibung dar*
- *$SEM\_RELS: E \rightarrow \mathcal{P}(\mathcal{R})$, $E := \{\mathcal{O} \cup \mathcal{OP} \cup \mathcal{S} \cup \mathcal{T}\}$, ordnet den Objekten, Operationen, Subjekten und Targets die semantischen Beziehungen zu*

Damit sind die Mittel gegeben, um einfache Sätze der natürlichen Sprache mit Subjekt, Prädikat, Akkusativ- und Dativ-Objekt in die semantische Aktivitätenbeschreibung abzubilden und damit den Kontext für das Quellobjekt einer Regel festzulegen.

4.4.3. Kontext von Zielobjekten

Mit den dargestellten Mechanismen kann das Quellobjekt einer semantischen Beziehung auf einen bestimmten Kontext festgelegt werden, der Kontext des Zielobjekts wird jedoch weiterhin nicht berücksichtigt. Das bedeutet, wenn in einer Regel ein Zielobjekt beispielsweise ausgeschlossen wird, so darf dieses nicht im Prozess auftreten, unabhängig vom Kontext, in dem dieses Zielobjekt im Prozess verwendet wird. Zur Verdeutlichung dieses Problems dient das Beispiel aus Abbildung 4.13. Die Abbildung zeigt einen un-

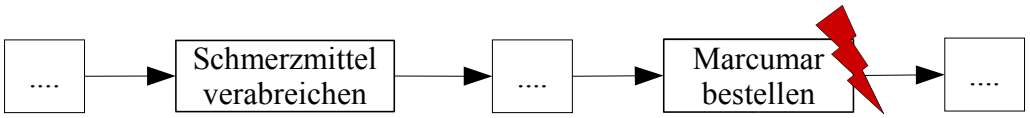


Abbildung 4.13.: Semantischer Konflikt: Fehlender Kontext des Zielobjekts

erwünschten semantischen Konflikt. Der Konflikt tritt auf, obwohl offensichtlich zwischen den zwei beteiligten Aktivitäten `Schmerzmittel verabreichen` und `Marcumar bestellen` in Wirklichkeit keine Wechselwirkung besteht. Die Ursache des Konflikts ist die semantische Ausschlussbeziehung zwischen Objekten des Typs `Schmerzmittel` und Objekten des Typs `Blutverdünnungsmittel`, die durch die Aktivität `Schmerzmittel verabreichen` im Prozess vorhanden ist. Da der Aktivität `Marcumar bestellen` in der semantischen Beschreibung das Objekt `Marcumar` zugeordnet wird, wird die genannte Ausschlussbeziehung verletzt.

Es müssen zusätzliche Maßnahmen ergriffen werden, um solche unerwünschten Konflikte zu vermeiden. Um dies zu erreichen muss es möglich sein, den Anwendungskontext des Zielobjekts einer semantischen Beziehung modellieren und auswerten zu können. D.h. die semantischen Beziehungen, die für den Anwendungskontext des Quellobjekts bestimmt wurden, werden beim Auftreten des Zielobjekts im Prozess nochmals anhand des Anwendungskontexts des Zielobjekts auf das Zutreffen für die jeweilige Situation überprüft. Dabei spielen die selben Kriterien eine Rolle wie für die Berücksichtigung des Kontexts beim Quellobjekt:

- Operation
- Subjekt
- Target

Die Begrenzung der Gültigkeit semantischer Beziehung durch die Formulierung von Anwendungssituationen des Zielobjekts erfordert eine Modellierungsmöglichkeit in den Ontologie-Elementen. Diese findet sich in Form folgender Attribute für die Kriterien Operation, Subjekt und Target:

- relatedOp: Legt fest, dass die Regel nur gilt, wenn das Zielobjekt der Regel zusammen mit der durch relatedOp festgelegten Operation verwendet wird.
- relatedSubj: Dadurch wird die Regel auf die Anwendungsfälle beschränkt, in denen das Zielobjekt zusammen mit dem in relatedSubj angegebenen Subjekt verwendet wird.
- relatedTar: Der Kontext des Zielobjekts, für den die Regel Gültigkeit besitzt, wird auf die Situationen reduziert, in denen das Zielobjekt zusammen mit dem genannten Target auftritt.

Somit kann das Beispiel aus Abbildung 4.13 korrigiert werden, in dem die zugrundeliegenden Ontologien entsprechend angepasst werden, wie Abbildung 4.14 zeigt.
Die Aktivität `Marcumar bestellen` besitzt als Objekt ihrer semantischen Beschrei-

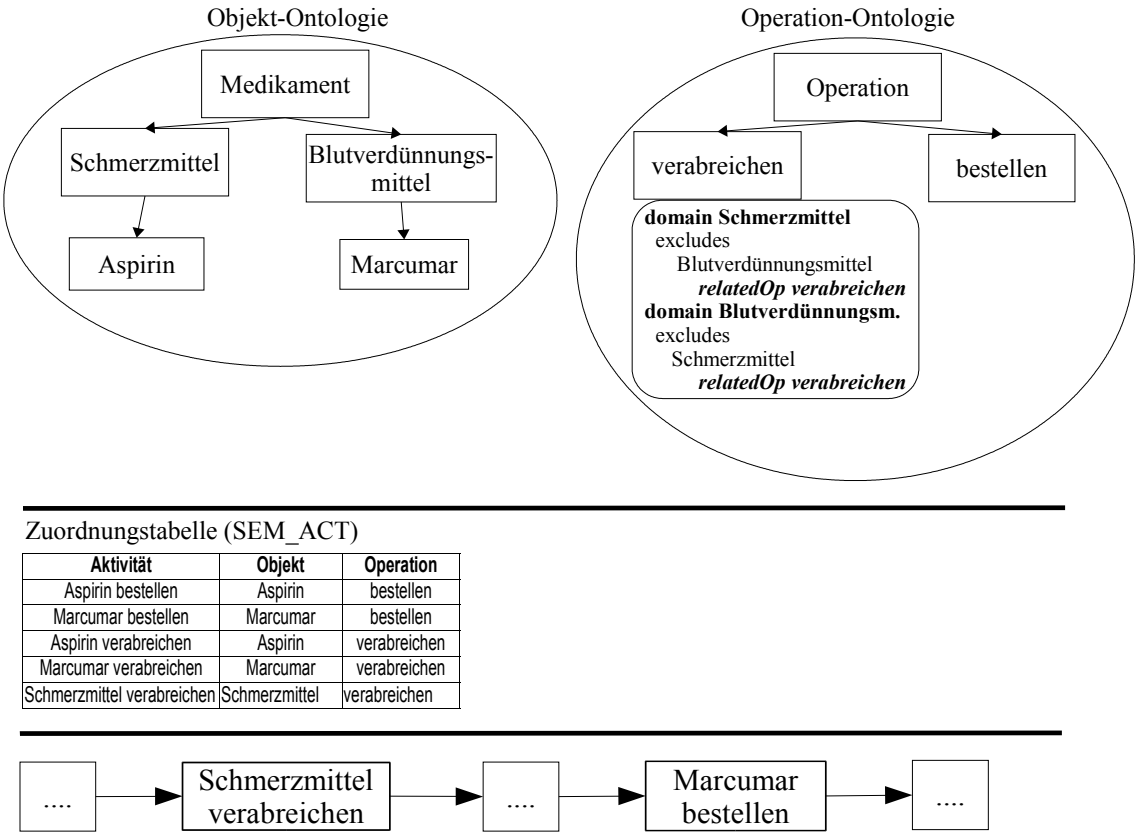


Abbildung 4.14.: Kontext des Zielobjekts

bung nach wie vor das Objekt `Marcumar`. Es kommt jedoch nicht mehr zum semantischen Konflikt, da die semantische Ausschlussbeziehung, die die Aktivität `Schmerzmittel verabreichen` durch ihre semantische Definition fordert, nicht mehr verletzt wird. Dies ist der Fall, da das Objekt `Marcumar` in der Aktivität `Marcumar bestellen` nicht zusammen mit der Operation `verabreichen` vorkommt, sondern mit der Operation `bestellen` verwendet wird.

4.4.4. Ausprägungen von semantischen Aktivitätenbeschreibungen

Bis jetzt wurde einer Aktivität genau eine semantische Beschreibung zugewiesen. Dies verhinderte in den bisherigen Beispielen, dass es eine allgemeine Aktivität `Medikation verabreichen` geben konnte. Eine generische Aktivität wie `Medikation verabreichen` muss verschiedene semantische Beschreibungen besitzen können. Um dies

zu ermöglichen, wird zwischen verschiedenen Ausprägungen der Semantikbeschreibung einer Aktivität unterschieden. Bei Verwendung einer Aktivität in einem Prozess muss jeweils angegeben werden, welche Ausprägung benutzt werden soll. Dabei ist es möglich, eine default-Ausprägung zu definieren, die benutzt wird, wenn keine konkrete Ausprägung benannt wird.

Definition 4.5 (Domäne - verschiedene Aktivitätenausprägungen)

Eine Domäne Dom sei ein Tupel
$(\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS, ACT\_OCCURRENCES)$

- $ACT\_OCCURRENCES \subseteq (\mathcal{N} \times \mathbb{N})$ *repräsentiert die festgelegten Ausprägungen der Aktivitäten in der Domäne*
- SEM_ACT: $ACT\_OCCURRENCES \rightarrow \mathcal{P}(K)$, $K := \mathcal{O} \cup (\mathcal{O} \times \mathcal{OP}) \cup (\mathcal{O} \times \mathcal{OP} \times \mathcal{S}) \cup (\mathcal{O} \times \mathcal{OP} \times \mathcal{T}) \cup (\mathcal{O} \times \mathcal{OP} \times \mathcal{S} \times \mathcal{T})$ *stellt die semantische Aktivitätenbeschreibung für die verschiedenen Ausprägungen der Aktivitäten dar*

Definition 4.6 (Prozessschema - verschiedene Aktivitätenausprägungen)

S = (N,E,D,Dom) sei ein Prozessschema in der Domäne
$Dom = (\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS, ACT\_OCCURRENCES)$ *mit*

- $N \subseteq ACT\_OCCURRENCES$ *stellt die dem Schema zugeordneten Aktivitäten in der Form der verwendeten Ausprägungen dar*

4.4.5. Zusammenspiel der Ontologien

Einige Beispiele sollen die beschriebenen Zusammenhänge zwischen den verschiedenen Ontologien veranschaulichen. Die Ontologien, die die Basis für die weiteren Beispiele bilden, sind in Abbildung 4.15 illustriert.

Die Beispiele stützen sich jeweils auf die Ermittlung der semantischen Korrektheit im Zuge einer Einfügeoperation. Dabei werden unterschiedliche Szenarien zum Einsatz kommen, die die Mechanismen und verbliebenen Problemstellen dieser Mechanismen verdeutlichen sollen. Beim Auftreten einer solchen Problemstelle wird diese diskutiert und mögliche Lösungsansätze dargelegt. Die Beispiele sind so einfach wie möglich gehalten, um nicht vom Schwerpunkt, d.h. der Verdeutlichung des Zusammenwirkens der verschiedenen Ontologien, abzulenken. Deswegen wird nicht der vollständige Algorithmus für den Korrektheitstest zugrundegelegt. Die Beispiele sind vielmehr so konstruiert, dass semantische Konflikte durch die Verletzung der Semantikregeln der einzufügenden Aktivität entstehen.

Für die einzufügende Aktivität aus Abbildung 4.16 werden die semantischen Beziehungen aus der Objekt-Ontologie für das Objekt `Marcumar` ausgewertet, da in der Semantikbeschreibung der Aktivität kein anderer "Ontologien-Eintrag" festgelegt ist. Eine dieser Regeln definiert eine Ausschlussbeziehung zum Objekt `Aspirin`. Da die Aktivität

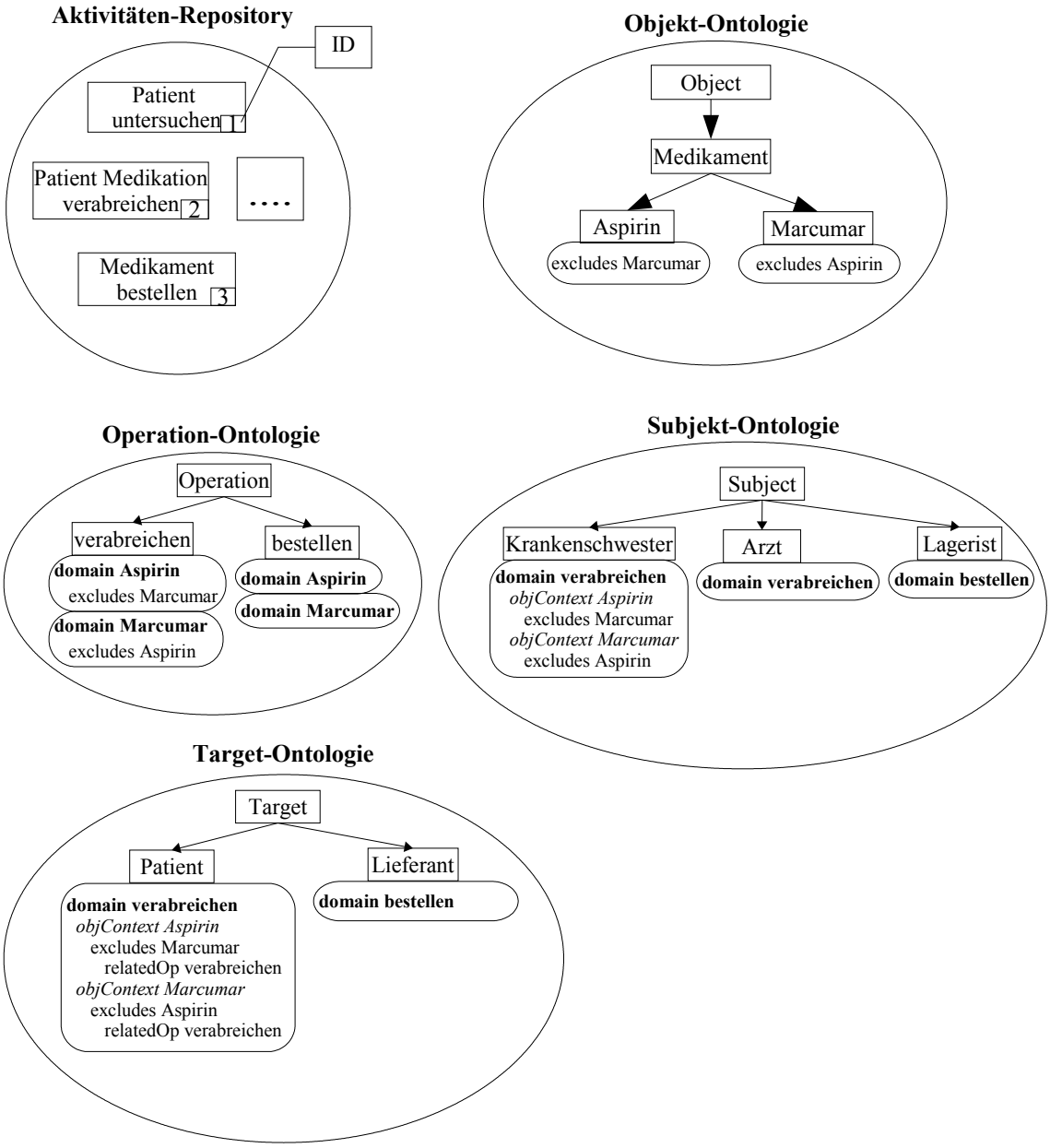


Abbildung 4.15.: Beispiel-Ontologien

`Patient Medikation verabreichen` in der Ausprägung 1 ein Objekt `Aspirin` definiert, kommt es beim Einfügen zu einem Konflikt durch die verletzte Ausschlussbeziehung.

Abbildung 4.17 stellt eine weitere Einfügesituation dar. Für die einzufügende Aktivität werden diesmal die Regeln aus der Operation-Ontologie für die Operation `bestellen`

1. Fall:

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	2	Aspirin	-	-	-
2	2	Marcumar	-	-	-

Prozess P:

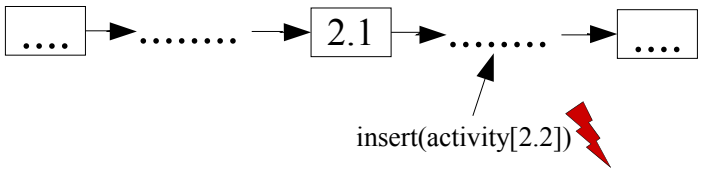


Abbildung 4.16.: Objekt-Ontologie als Ziel der Auswertung

2. Fall:

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	3	Aspirin	bestellen	-	-
2	3	Marcumar	bestellen	-	-

Prozess P:

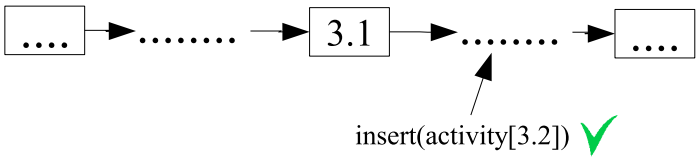


Abbildung 4.17.: Operation-Ontologie als Ziel der Auswertung

ausgewertet, da in der Aktivitätenbeschreibung sowohl ein Objekt, als auch eine Operation angegeben sind. Da für diese Objekt-Operation-Kombination keinerlei Regeln in der Ontologie vorhanden sind, kommt es zu keinem Konflikt beim Einfügen.

Der dritte Anwendungsfall ist in Abbildung 4.18 abgebildet. Die einzufügende Aktivität 2.1 hat lediglich für den Objekteintrag einen Wert. Somit wird für das Objekt `Marcumar` die Objekt-Ontologie konsultiert, wo für das Objekt eine Ausschlussbeziehung zum Objekt `Aspirin` hinterlegt ist. Die Aktivität `Medikament bestellen` enthält in der im Prozess verwendeten Ausprägung das Objekt `Aspirin`, womit der Ausschluss greift und es zu einem semantischen Konflikt kommt.
Dieser semantische Konflikt tritt auf, da für das Zielobjekt der Regel kein spezifischer Kontext festgesetzt wurde. Um einen spezifischen Kontext zu setzen, der den unerwünschten Konflikt in Abbildung 4.18 vermeidet, muss die Objektontologie aus Ab-

3. Fall:

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	3	Aspirin	bestellen	-	-
1	2	Marcumar	-	-	-

Prozess P:

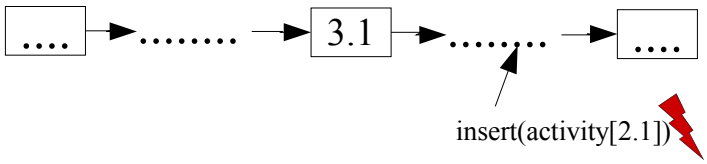


Abbildung 4.18.: Objekt- und Operation-Ontologie

bildung 4.15 wie in Abbildung 4.19 dargestellt mit der Zusatzinformation `relatedOp verabreichen` angereichert werden.

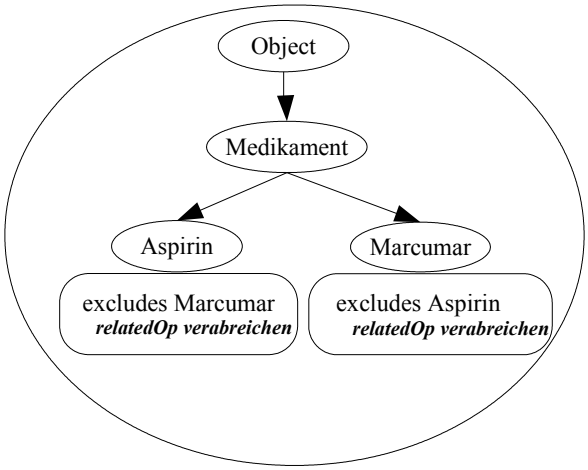


Abbildung 4.19.: Objekt-Ontologie - Erweiterung

Beim Einfügen in Abbildung 4.20 werden wiederum die Regeln aus der Objekt-Ontologie für das Objekt `Marcumar` ausgewertet. Es existiert eine Regel, die festlegt, dass es zu einem semantischen Konflikt kommt, falls im Prozess bereits das Objekt `Aspirin` existiert und dies zusammen mit der Operation `verabreichen` verwendet wird. `Aspirin` tritt jedoch im Prozess mit der Operation `bestellen` auf, so dass Aktivität 2.1 ohne Konflikt eingefügt werden kann.

Der vierte Fall aus Abbildung 4.21 stellt eine Situation dar, in der der Arzt die Medikationen Aspirin und Marcumar verabreicht. Beim Einfügen der Aktivität 2.2 werden somit die Regeln aus der Subjekt-Ontologie ausgewertet. Für das Subjekt `Arzt` ist für die

3. Fall (basierend auf verbesserter Objekt-Ontologie):

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	3	Aspirin	bestellen	-	-
1	2	Marcumar	-	-	-

Prozess P:

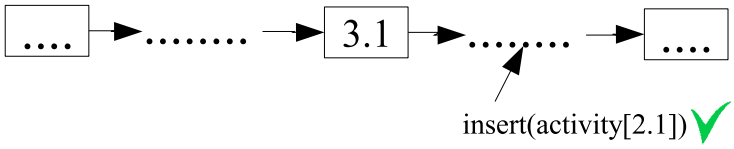


Abbildung 4.20.: Erweiterte Objekt-Ontologie - Auswertung

4. Fall: „Der Arzt verabreicht Aspirin.“
„Der Arzt verabreicht Marcumar.“

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	2	Aspirin	verabreichen	Arzt	-
2	2	Marcumar	verabreichen	Arzt	-

Prozess P:

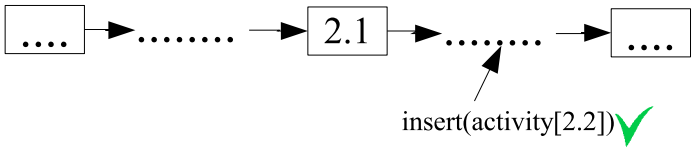


Abbildung 4.21.: Subjekt-Ontologie als Ziel der Auswertung

Domäne `verabreichen` keine Regel definiert, was bedeutet, dass der Arzt bei seinen Entscheidungen keinerlei Einschränkungen unterliegt. Die Restriktion, dass `Aspirin` und `Marcumar` nicht zusammen verabreicht werden dürfen, wird durch das ausführende Subjekt `Arzt` aufgehoben.

Der fünfte Anwendungsfall aus Abbildung 4.22 greift schließlich auch den Target-Eintrag der Semantikbeschreibung auf. Beim Einfügen der Aktivität 2.2 werden entsprechend die Regeln aus der Target-Ontologie für das Target `Patient` mit dem domain-Attribut `verabreichen` ausgewertet. Für den Objektkontext `Marcumar` wird dort definiert, dass sich das Objekt `Aspirin` in Verbindung mit der Operation `verabreichen` nicht bereits im Prozess befinden darf. Dies ist jedoch der Fall, so dass das Einfügen von Aktivität 2.2 einen semantischen Konflikt verursacht.

5. Fall: „Das Medikament Aspirin wird dem Patient verabreicht.“
„Das Medikament Marcumar wird dem Patient verabreicht.“

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	2	Aspirin	verabreichen		Patient
2	2	Marcumar	verabreichen		Patient

Prozess P:

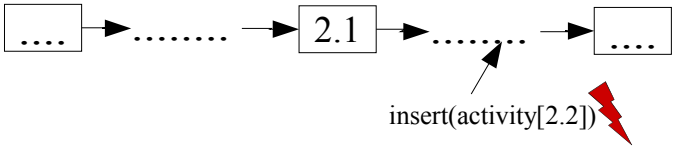


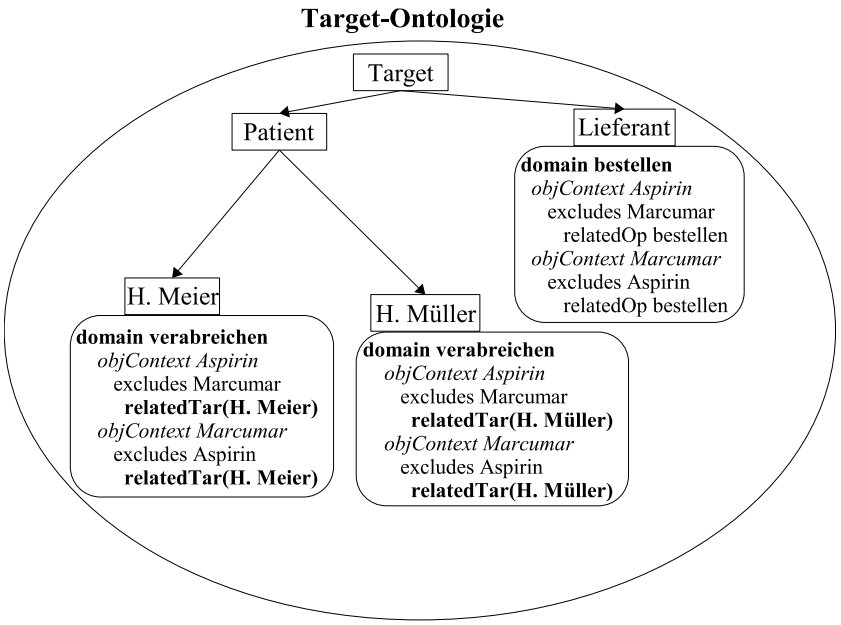
Abbildung 4.22.: Target-Ontologie als Ziel der Auswertung

Der zuletzt dargestellte fünfte Fall weist beim zweiten Blick ein neues Problem auf. Der semantische Konflikt durch die verletzte Ausschlussbeziehung macht Sinn, wenn es sich beim Target `Patient` in beiden Aktivitäten auch tatsächlich um den selben Patient handelt. Handelt es sich jedoch in Wirklichkeit um verschiedene Patienten, so ist nicht einzusehen, warum die Verabreichung von Aspirin und Marcumar einen semantischen Konflikt verursacht. Es ist ein Mechanismus wünschenswert, mit dem man zwischen verschiedenen Targets des gleichen Typs unterscheiden kann. Ein möglicher Lösungsansatz dafür ist Gegenstand des nächsten Kapitels.

4.4.6. Unterscheidung verschiedener Individuals

Das Beispiel aus Abbildung 4.22 hat deutlich gemacht, dass es je nach Anwendung möglich sein muss, verschiedene Ausprägungen eines Objekttyps unterscheiden zu können. Mit den gleichen Argumenten ist es sinnvoll, ebenso zwischen verschiedenen Ausprägungen der Elemente der Kontext-Ontologien unterscheiden zu können.
Ein Individual stellt eine konkrete Instanz eines Typs einer Ontologie dar und muss das Kriterium der eindeutigen Identifizierbarkeit erfüllen. Das bedeutet, dass verschiedene Individuals eines Typs unterscheidbar sind. Abbildung 4.23 zeigt, wie das Beispiel aus Abbildung 4.22 abgeändert werden kann, um zu berücksichtigen, dass die zwei Medikationen verschiedenen Patienten, ”Herr Meier” und ”Herr Müller” verabreicht werden. Wie zu sehen ist, werden die Aktivitätenbeschreibungen angepasst, indem das Target `Patient` jeweils durch konkrete Personen ersetzt wird. Damit diese ausgewertet werden können, muss die Target-Ontologie um diese zwei individuellen Targets erweitert werden. Nur diese zwei Targets zu definieren genügt jedoch noch nicht, da die Aktivität 2.2 immernoch in Konflikt mit der durch Aktivität 2.1 definierten Ausschlussbeziehung `excludes Marcumar` steht. Um dies richtigzustellen, müssen die Zielobjekte der Ausschlussregeln über die Beschränkung `relatedTar` in den korrekten Kontext gesetzt

werden, so dass die Ausschlussbeziehung nur greift, wenn dem selben Target, z.B. dem Target `H. Meier`, beide Medikationen verabreicht werden.



5. Fall: „Das Medikament Aspirin wird Herrn Meier verabreicht."
„Das Medikament Marcumar wird Herrn Müller verabreicht."

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	2	Aspirin	verabreichen		H. Meier
2	2	Marcumar	verabreichen		H. Müller

Prozess P:

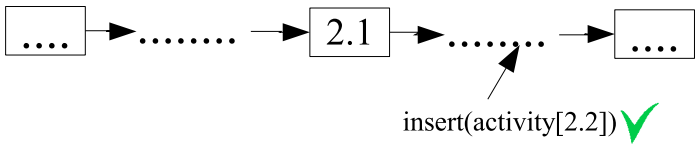


Abbildung 4.23.: Unterscheidung von Targets - konkrete Benennung

Die vorgestellte Lösung der unterschiedlichen Benennung verschiedener Targets stellt keine befriedigende Antwort auf die Frage der allgemeinen Unterscheidbarkeit von Targets dar. So müsste im Beispiel für jeden Patient ein neues Target, also ein neuer Typ innerhalb der Target-Ontologie, angelegt und im entsprechenden Prozess die verwendeten Ausprägung der Aktivität `Medikation verabreichen` ständig angepasst werden. Dies ist aus naheliegenden Gründen nicht praktikabel. Somit wird ein Mechanismus benötigt, der es ermöglicht, verschiedene Ausprägungen eines Typs, und nicht unterschiedliche Typen wie im obigen Lösungsansatz zu unterscheiden und den Kontext des Zielob-

jekts einer semantischen Beziehung entsprechend festzulegen. In der objektorientierten Programmierung kann das Laufzeitsystem verschiedene Objekte eines Typs anhand unterschiedlicher Speicherstellen identifizieren, da Objekte als Zeiger auf Speicherstellen realisiert werden können [WM97]. Dieser Ansatz ist jedoch nicht auf die semantische Beschreibung von Aktivitäten und ihre Auswertung übertragbar, da es sich bei den Einträgen der Aktivitätenbeschreibung nicht um unterscheidbare Speicherobjekte handelt, sondern die Einträge über den Typnamen in die entsprechende Ontologie verweisen. Deshalb muss ein anderer Mechanismus zur Unterscheidung verschiedener Individuals entwickelt werden, der folgende Kriterien erfüllen muss:

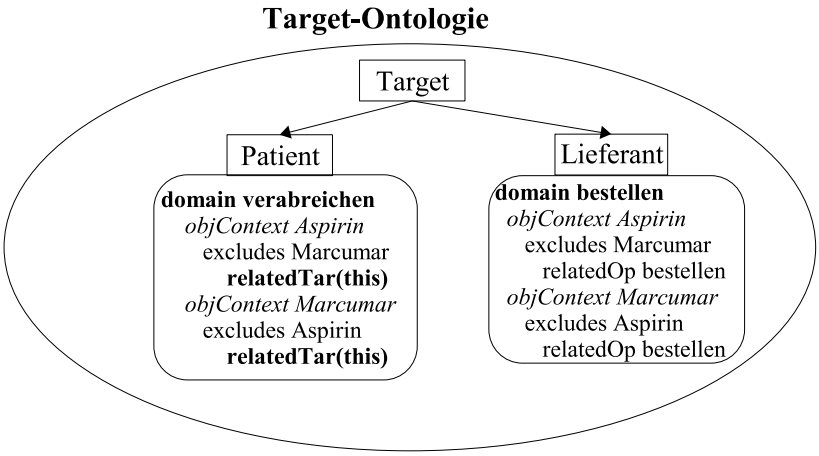
- Der Typzusammenhang muss bestehen bleiben und leicht identifizierbar sein.
- Eine spezifische Ausprägung oder Individual muss eindeutig zu identifizieren sein.
- Der Mechanismus soll sich nahtlos in die bestehenden Konzepte für die semantische Aktivitätenbeschreibung und die Festlegung des Kontexts für das Zielobjekt einer semantischen Regel einfügen.

Folgender Lösungsansatz erfüllt alle diese Kriterien. Ein Individual, das einer konkreten Ausprägung eines bestimmten Typs entspricht, wird innerhalb der semantischen Aktivitätenbeschreibung instantiiert. Die Instantiierung wird vollzogen, indem dem Typnamen eine eindeutige ID angehängt wird. Ein Individual besitzt also folgende Form:
T#ID , mit T entspricht dem Typnamen, # trennt den Typnamen von der ID und ID $\in \mathbb{N}$.

Um den Kontext für das Zielobjekt einer Regel auf die Situationen beschränken zu können, in denen die selbe Ausprägung des Targets verwendet wird, die verantwortlich für die auszuwertenden semantischen Regeln zeichnet, muss über `relatedTar` ein Bezug auf die jeweils aktuelle Ausprägung bzw. das aktuelle Individual hergestellt werden. Der Kontext für das Zielobjekt muss sich dynamisch an das aktuell ausgewertete Individual anpassen. Etwas ähnliches findet sich in einigen OOP-Sprachen, wie z.B. Java und ist dort unter dem Konzept der ”Selbstreferenz” bekannt. Wie in [Ste99] dargelegt, dient diese Referenz bei Java beispielsweise dem Zugriff auf durch lokale Methoden-Parameter überdeckte Objekt-Felder, wobei über die Selbstreferenz immer das aktuelle Objekt referenziert wird, für das beispielsweise ein Methodenaufruf durchgeführt wurde. Bei Java wird dafür das Schlüsselwort `this` benutzt. Auch für den dynamisch anzupassenden Kontext des Zielobjekts einer Regel soll das Schlüsselwort `this` als Wert des `relatedTar`-Elements benutzt werden.

Damit kann das Beispiel aus Abbildung 4.22 erneut angepasst werden. Mit den Aktivitäten sind nicht länger konkrete Personennamen als Targets assoziiert, sondern der abstrakte Typ `Patient`, einmal als Individual `Patient#1`, einmal als Individual `Patient#2`. In der Target-Ontologie werden die Zielobjekte der semantischen Beziehungen des Targets Patient für die domain `verabreichen` in einen Kontext der Selbstreferenz gesetzt. Das Objekt `Marcumar` in Abbildung 4.24 darf also nicht zusammen mit dem Target `Patient#1` in der einzufügenden Aktivität verwendet werden, da sonst ein semantischer Konflikt durch das Einfügen verursacht wird. Dies ist im Beispiel nicht der

Fall. Dort wird das Objekt `Marcumar` mit dem Target `Patient#2` verwendet, so dass keine semantische Beziehung verletzt wird und somit kein semantischer Konflikt auftritt. Es kann nun also im Beispiel zwischen zwei beliebigen Patienten unterschieden werden,



5. Fall: „Das Medikament Aspirin wird Herrn Meier verabreicht.“
„Das Medikament Marcumar wird Herrn Müller verabreicht.“

Ausprägung	Aktivität-ID	Objekt	Operation	Subjekt	Target
1	2	Aspirin	verabreichen		Patient#1
2	2	Marcumar	verabreichen		Patient#2

Prozess P:

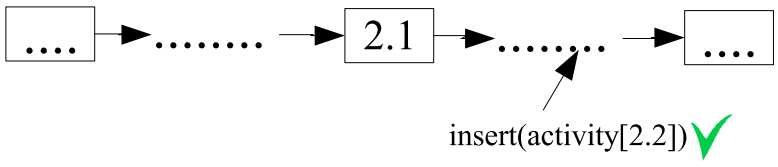


Abbildung 4.24.: Unterscheidung von Targets - ID-Ansatz

was den realen Erfordernissen entspricht.

Der selbe Mechanismus kann verwendet werden, um verschiedene Individuals von Elementen der anderen Ontologien zu unterscheiden. Wie bei den Targets werden die Typnamen in der Aktivitätenbeschreibung um das Separationszeichen und eine eindeutige ID erweitert. Über den für die Target-Individuals beschriebenen Mechanismus der Selbstreferenzierung können auch hier die einzelnen Ausprägungen unterschieden werden.

Die Einführung der Individuals bzw. Ausprägungen von Typen bringt erweiterte Möglichkeiten in die semantische Beschreibung von Aktivitäten. Bisher wurden lediglich

Typnamen verwendet, um die entsprechenden Einträge der semantischen Aktivitätenbeschreibung mit Werten zu versehen. Nun kommen als zweite Möglichkeit konkrete, unterscheidbare Individuals dieser Typen hinzu. Individuals von Typen werden jedoch nur benötigt, wenn verschiedene Einträge voneinander unterschieden werden müssen, ansonsten genügt die Angabe des Typnamens.

4.5. Mehrfachklassifizierung

Der Ontologie-Ansatz ist charakterisiert durch die Etablierung von Relationen zwischen Typen auf der Basis von Obertyp-Untertyp-Beziehungen. Diese Organisationsform birgt viele Vorteile in sich, unter anderem durch das Prinzip der Substituierbarkeit von Zielobjekten. Im bisher beschriebenen Mechanismus wird für einen Typ nur ein einziger Obertyp berücksichtigt. Auf den ersten Blick scheint dies ausreichend zu sein. Auch viele Programmiersprachen, wie beispielsweise Java, lassen nur einen einzigen Obertyp zu, jedoch schränkt diese Limitation die Praxistauglichkeit deutlich ein. In der Aktivität `Aspirin verabreichen` aus den früheren Beispielen wird `Aspirin` als Untertyp des Typs `Schmerzmittel` betrachtet. Aspirin kann jedoch auch für andere Einsatzzwecke benutzt werden. So tragen beispielsweise die Wirkstoffe zur Verdünnung des Blutes bei. Ist die Zuordnung nur zu einem Obertyp möglich, kann dieser Sachverhalt nicht modelliert werden. `Aspirin` ist entweder ein Untertyp von `Schmerzmittel` oder ein Untertyp von `Blutverdünnungsmittel`. Dies stellt nicht das erwünschte Ergebnis dar. Um die Realität genauer abbilden zu können, muss es möglich sein, dass ein Typ Untertyp mehrerer verschiedener Obertypen sein kann. Dazu sind folgende Fragen zu beantworten:

1. Kann der bestehende Ansatz zur Einordnung der Typen dahingehend erweitert werden, dass Mehrfachklassifizierung unterstützt wird?
2. Treten neue Probleme durch Mehrfachklassifizierung auf?

Definition 4.7 zeigt, wie die Regelmenge eines mehrfach-klassifizierten Objekts gebildet wird.

Definition 4.7 (Regelmenge eines mehrfach-klassifizierten Objekts)

Gegeben sei eine Ontologie $ONT = (\mathcal{O}, R, \mathcal{R})$. $o \in \mathcal{O}$ *sei ein mehrfach-klassifiziertes Objekt mit den direkten Obertypen* $p_1, ..., p_k \in \mathcal{O}$. $RS(p_1), ..., RS(p_k)$ *seien die Regelmengen der Obertypen. Die Regelmenge* $RS(o)$ *des Objekts o stellt sich wie folgt dar:*
$RS(o) := RD(o) \cup RS(p_1) \cup, ..., \cup RS(p_k)$

Abbildung 4.25 visualisiert die Idee der Mehrfachklassifizierung am Beispiel des Objekts `Aspirin`. Für Regeln, die als Zielobjekt das Objekt `Schmerzmittel` besitzen, gilt wie bisher auch das Auftreten eines Objekts `Aspirin` als erfüllte Bedingung. Neu hinzu kommt nun, dass ebenso Regeln mit dem Zielobjekt `Blutverdünnungsmittel` durch Vorkommen des Objekts `Aspirin` erfüllt werden können.

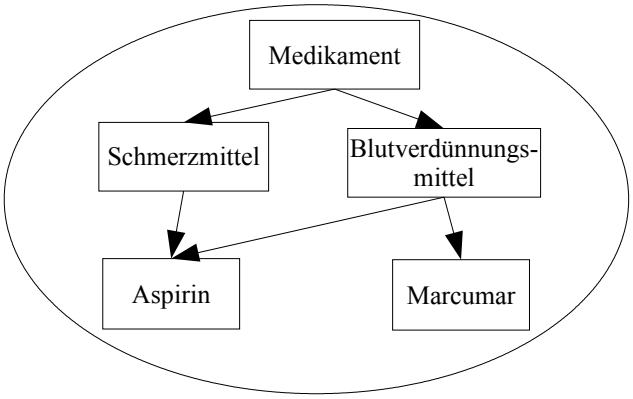


Abbildung 4.25.: Ontologie mit Mehrfachklassifizierung

In Abbildung 4.25 sind absichtlich keine Regeln in der Ontologie dargestellt. Abbildung 4.26 macht die Gründe dafür deutlich. Das Objekt `Schmerzmittel` definiert eine Ausschlussbeziehung zu Objekten des Typs `Blutverdünnungsmittel`, und durch die Typbeziehungen auch zu allen Untertypen von `Blutverdünnungsmittel`. Analog dazu legt das Objekt `Blutverdünnungsmittel` eine Ausschlussbeziehung zu Objekten des Typs `Schmerzmittel` oder zu Objekten von Untertypen davon fest. Dem Objekt `Aspirin` sind über die Mehrfachklassifizierung zwei Obertypen zugeordnet, `Schmerzmittel` und `Blutverdünnungsmittel`. Durch die Vererbung gelten die Regelmengen beider Obertypen für das Objekt `Aspirin`. Damit würde das Objekt `Aspirin` selbst die beiden definierten Ausschlussbedingungen verletzen und so zu einem semantischen Konflikt führen, das Objekt widerspricht sozusagen sich selbst. Dieser Zusammenhang ist in Abbildung 4.26 beispielhaft dargestellt.

Das Auftreten dieses paradoxen Verhaltens muss näher untersucht werden, d.h. es muss die Frage beantwortet werden, wann das inkonsistente Regelwerk zu Problemen führt. Für `excludes`-Beziehungen zeigt Abbildung 4.26 deutlich, dass die aufgezeigte Inkonsistenz in den Regeln zu dem beschriebenen paradoxen Verhalten des Selbstausschlusses führt. Es bleibt zu untersuchen, ob solche Probleme auch im Zusammenhang mit `dependsOn`-Beziehungen auftreten können. Dazu soll ein Beispiel dienen, das in Abbildung 4.27 visualisiert ist. Der dargestellte Prozess soll unter dem Gesichtspunkt der semantischen Korrektheit betrachtet werden. Dabei werden die zwei Regeln `dependsOn(kombinierte Behandlung,Behandlung1)` und `dependsOn(kombinierte Behandlung,Behandlung2)` des Objekts `kombinierte Behandlung` ausgewertet. Diese beiden Abhängigkeiten werden durch das Objekt `kombinierte Behandlung` der Aktivität `kombinierte Behandlung durchführen` beide erfüllt, da das Objekt `kombinierte Behandlung` sowohl Untertyp von `Behandlung1` als auch Untertyp von `Behandlung2` ist. Durch die Vereinigung der Regelmengen entsteht bei `dependsOn`-Beziehungen also keine semantische Inkonsistenz, da das von den ent-

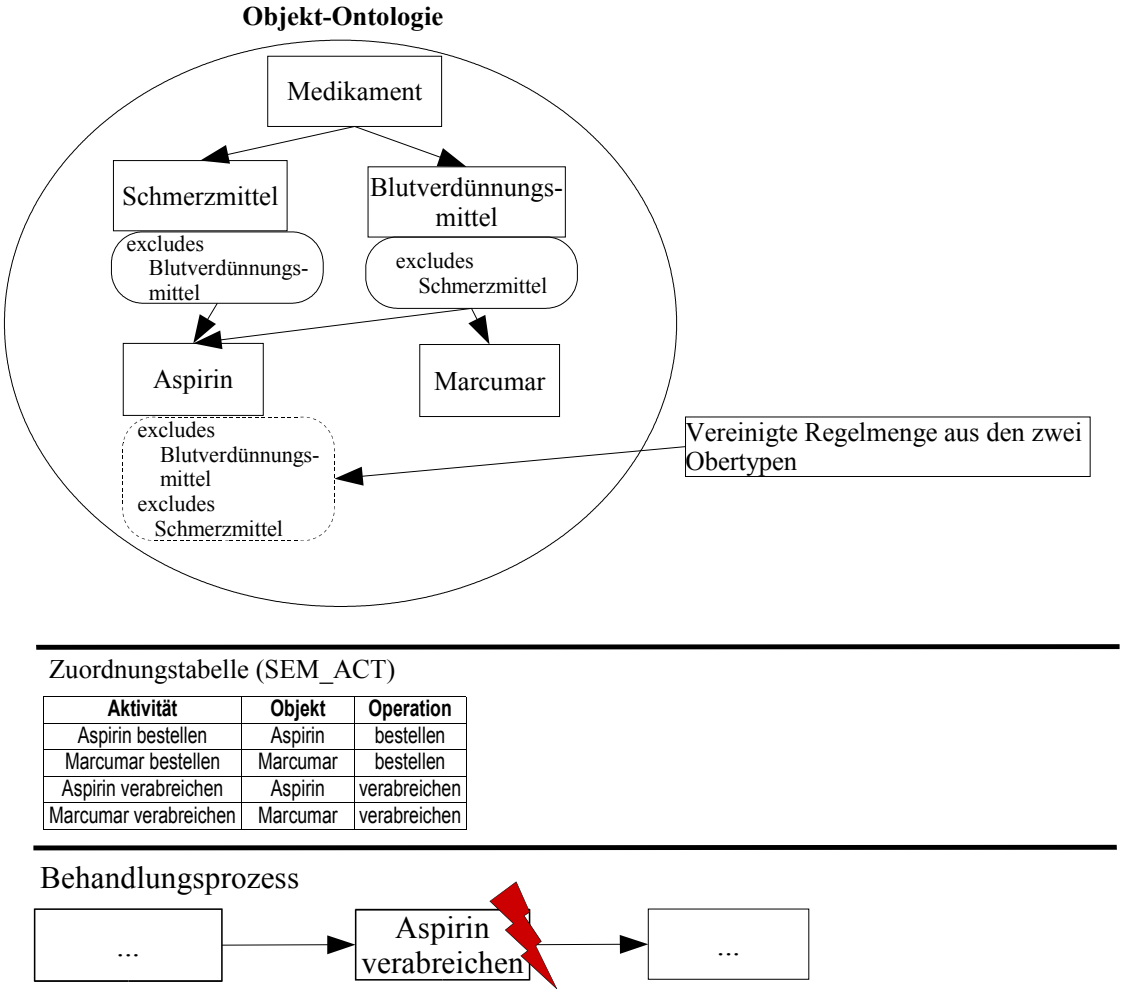


Abbildung 4.26.: Mehrfachklassifizierung - Selbstausschluss

sprechenden Obertypen abgeleitete Objekt selbst die Abhängigkeiten erfüllt.

Es muss untersucht werden, wie mit der Situation des Selbstausschlusses umgegangen werden kann, um inkonsistente Regelmengen zu vermeiden. Folgende Möglichkeiten zur Umgehung des Problems stehen zur Diskussion:

1. Es muss überprüft werden, ob ein Element, das von mehreren Obertypen erbt, in einer Situation des Selbstausschlusses resultiert. In diesem Fall kann das Objekt nicht auf diese Weise definiert werden.
2. Es werden nur diejenigen Regeln aus den Regelmengen auf den abgeleiteten Typ vererbt, die das Objekt nicht in einen Selbstausschluss setzen.

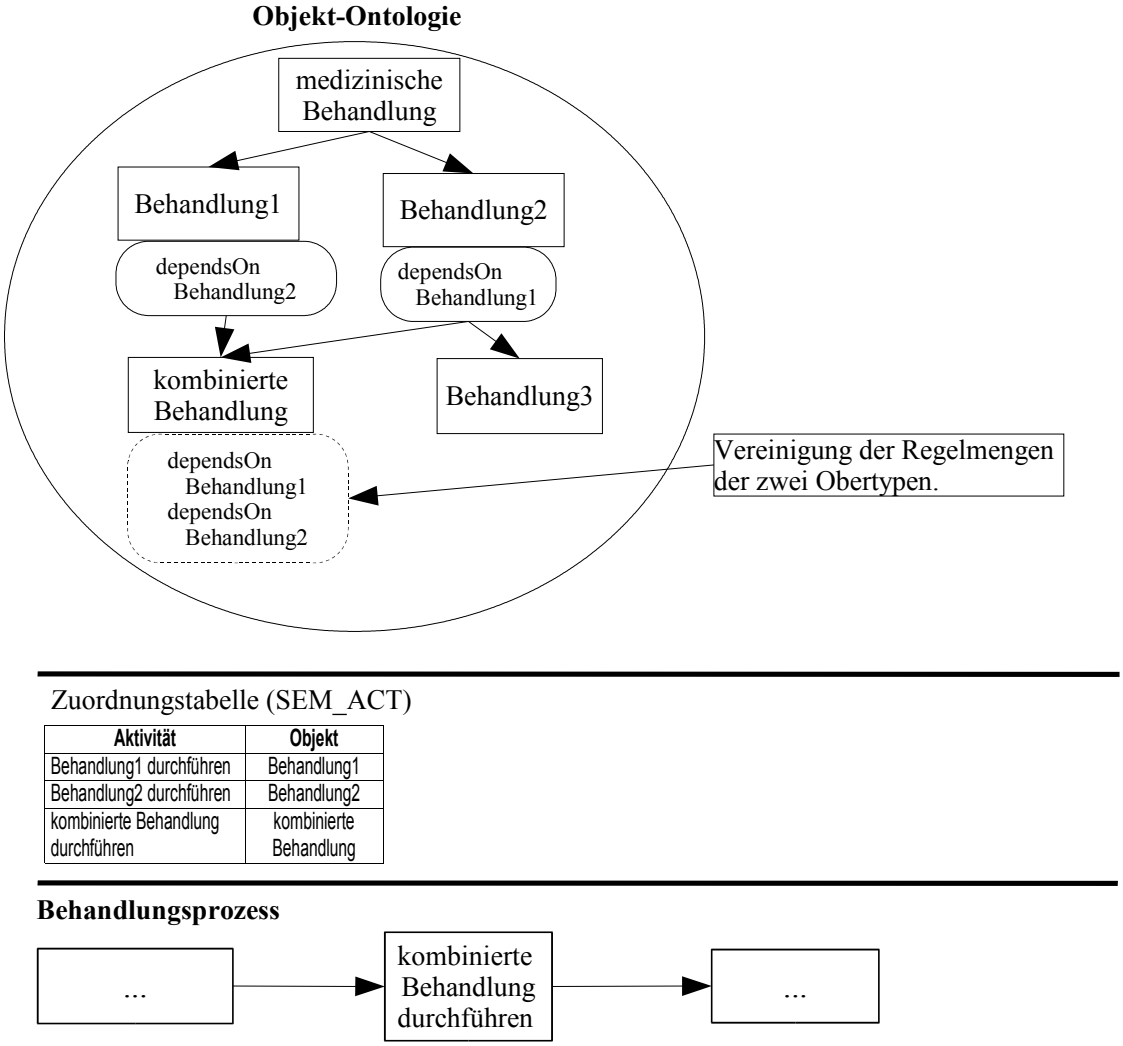


Abbildung 4.27.: Mehrfachklassifizierung - selbsterfüllte Abhängigkeiten

3. Objekte des mehrfach-klassifizierten Typs, sowie Objekte der Untertypen, werden automatisch von der Auswertung von Regeln ausgenommen, die das Objekt in einen Selbstausschluss setzen.

Die erste und zweite Strategie setzen voraus, dass die vererbten Regelmengen hinsichtlich ihrer Zielobjekte auf gegenseitige Konflikte untersucht werden müssen. Der Unterschied zwischen den beiden Strategien liegt dann letztlich in der Behandlung des Ergebnisses. Wird eine Inkonsistenz in der Regelmenge erkannt, so kann im ersten Fall der abgeleitete Typ nicht in die Ontologie eingefügt werden. Dies stellt sicher, dass nur Typen definiert werden können, die in sich nicht widersprüchlich sind. Der zweite Lösungsansatz fährt eine Strategie der selektiven Auswahl von Regeln, d.h. nur die Regeln aus

den Regelmengen verbleiben in der geerbten Regelmenge, die keinen Selbstausschluss des Objekts verursachen. Dies führt dazu, dass es keine Einschränkung in der Definition der Typen gibt, jedoch die durch Mehrfachklassifizierung vereinigten Regelmengen weniger ausdrucksstark sind, als eventuell beabsichtigt. Zur Verdeutlichung dieses Sachverhaltes dient Abbildung 4.28. Die Ontologie legt eine Ausschlussbeziehung `excludes(Schmerzmittel,Blutverdünnungsmittel)` fest. Die korrespondierende Bedingung `excludes(Blutverdünnungsmittel,Schmerzmittel)` hingegen wurde nicht mit angegeben, da dies aus Gründen der Symmetrie bei Ausschlussbeziehungen nicht notwendig ist. Das Objekt `Aspirin` ist sowohl Untertyp von `Schmerzmittel` als auch von `Blutverdünnungsmittel`. Da die vererbte Regelmenge Inkonsistenzen enthält, wird zur Umgehung die zweite Strategie gewählt, d.h. die Regeln, die die Inkonsistenz verursachen, werden aus der Regelmenge entfernt. Beim Test des ersten dargestellten Prozesses auf Korrektheit tritt nun kein semantischer Konflikt durch den Selbstausschluss des mehrfach-klassifizierten Objekts `Aspirin` auf. Der zweite Prozess zeigt ebenso keinen semantischen Konflikt an. In Wirklichkeit besteht die Medikamentenunverträglichkeit zwischen `Aspirin` und `Marcumar` weiterhin. Die zweite Strategie führt also nicht zum erwünschten Ergebnis, da die semantischen Beziehungen aufgeweicht werden und somit semantisch inkorrekte Prozesse erzeugt werden können. Sie stellt damit keine praxistaugliche Möglichkeit dar.

Es bleibt die dritte Strategie zu untersuchen. Diese benutzt einen anderen Ansatzpunkt als die vorigen drei Möglichkeiten. Es werden keine Änderungen an den geerbten Regeln vorgenommen und alle Regeln in der vereinigten Regelmenge belassen. Die Umgehung des Konsistenzproblems wird in die Verantwortung der Algorithmen zur Korrektheitsprüfung übergeben. Das bedeutet, Regeln, die ein Typ definiert oder geerbt hat, sind nicht auf den Typ selbst oder alle seine Untertpyen anzuwenden. Dazu lohnt ein zweiter Blick auf das Beispiel aus Abbildung 4.26. Dieses Mal soll das Beispiel aus der Perspektive der dritten Strategie betrachtet werden. Der semantische Konflikt existiert nicht länger. Die Regel `excludes(Aspirin,Blutverdünnungsmittel)` kommt zur Auswertung. `Aspirin` wird als `Schmerzmittel` und als `Blutverdünnungsmittel` identifiziert und würde damit die definierte Ausschlussbeziehung verletzen. In der Regel `excludes(Aspirin,Blutverdünnungsmittel)` müsste das Zielobjekt der Regel `Blutverdünnungsmittel` also durch `Aspirin` substituiert werden. Die temporäre Regel lautet `excludes(Aspirin, Aspirin)`. Die Kernaussage der dritten Strategie ist jedoch, dass die Regeln nicht auf das Element selbst angewendet werden dürfen, also das Quellobjekt und das Zielobjekt der temporären Regel nicht das selbe sein dürfen. Die Bedingung der definierten Ausschlussbeziehung wird im Beispiel durch das Objekt `Aspirin` also nicht erfüllt, so dass kein semantischer Konflikt entsteht. Dies stellt das für die Ontologie erwartete und erwünschte Verhalten dar.
Die verbliebene Ausschlussbeziehung greift nur für Objekte, die nicht dem mehrfachklassifizierten Typ oder einem seiner Untertypen entsprechen. Damit stellt sich das Beispiel aus Abbildung 4.28 unter der Berücksichtigung der dritten Strategie dar wie in Abbildung 4.29. Der erste Prozess ist nach wie vor semantisch korrekt, da die Ausschlussbeziehungen nicht für das Objekt `Aspirin` gelten. Der zweite dargestellte Prozess

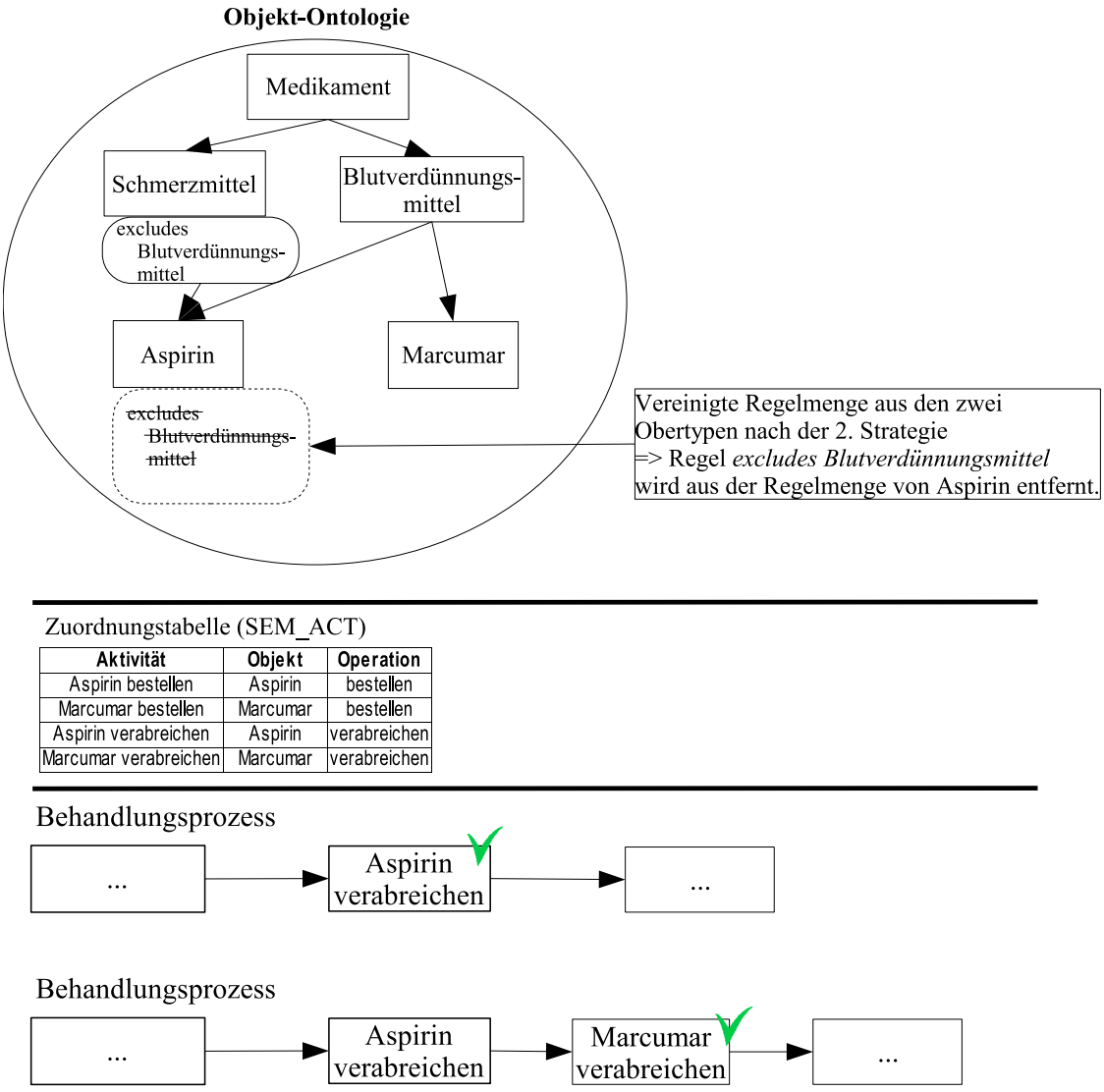


Abbildung 4.28.: Behandlung inkonsistenter Regelmengen - 1. Möglichkeit

signalisiert jedoch korrekterweise einen Konflikt aufgrund der verletzten Ausschlussbeziehung `excludes(Aspirin,Blutverdünnungsmittel)`. Genauer betrachtet ergibt sich dieser Konflikt folgendermaßen: Bei der Substitution von `Blutverdünnungsmittel` durch `Marcumar` entsteht die temporäre Regel `excludes(Aspirin,Marcumar)`. Da `Marcumar` nicht Untertyp von `Aspirin` ist, wird diese Ausschlussbedingung erfüllt und der semantische Konflikt tritt auf.

Wie dargestellt ist die zweite Strategie nicht praxistauglich. Für eine reale Umsetzung muss zwischen der ersten und dritten Strategie gewählt werden. Eine Bewertung, welche

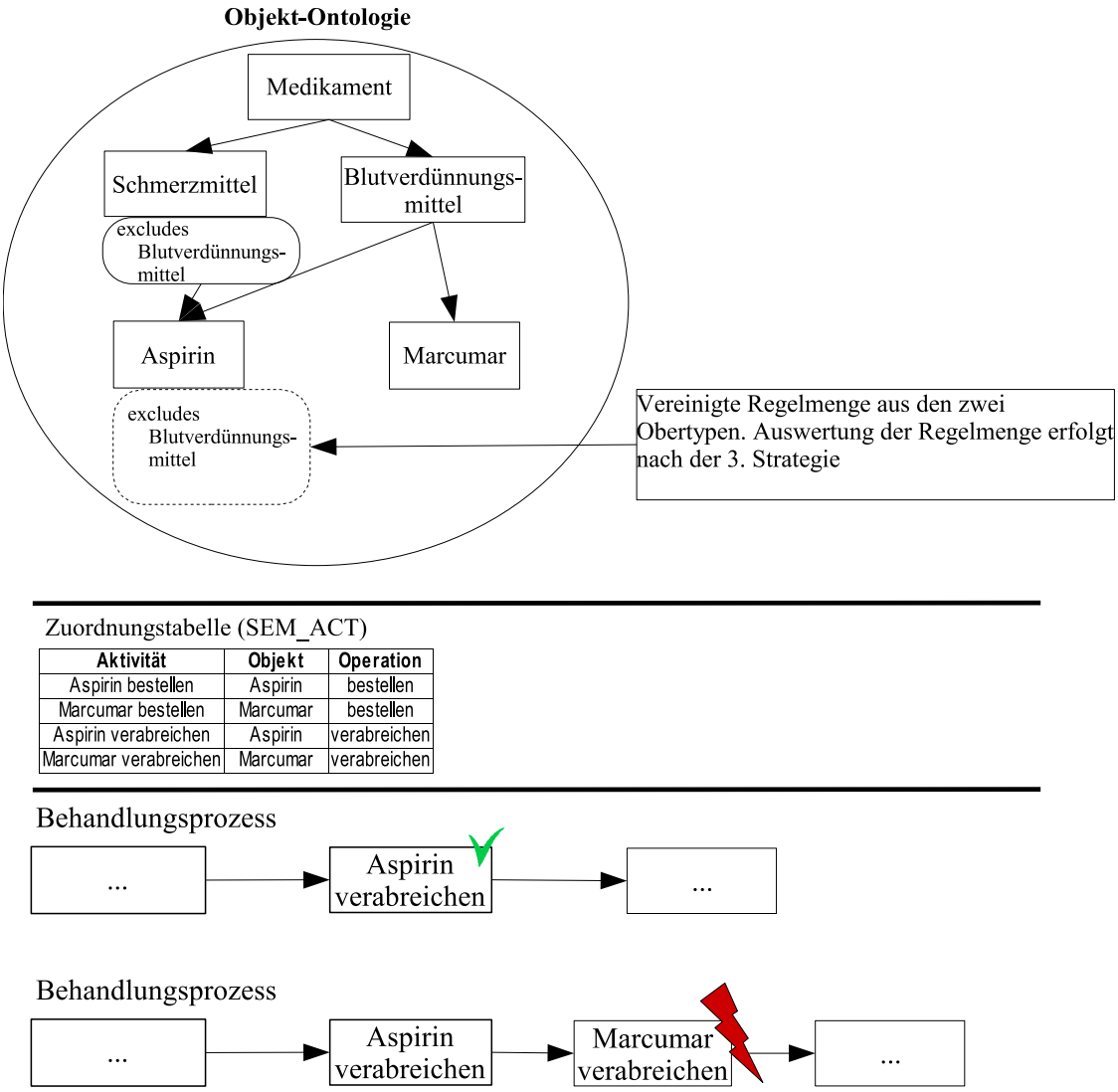


Abbildung 4.29.: Behandlung inkonsistenter Regelmengen - 2. Möglichkeit

der beiden Strategien vorzuziehen ist, fällt schwer. Die erste Strategie ist sehr restriktiv und erlaubt nur mehrfach-klassifizierte Objekte zu definieren, die keine Ausschlussbeziehung mit einem Zielobjekt besitzen, das aus einem der Vererbungsstränge stammt. Diese Restriktivität kann in manchen Fällen eine notwendige Mehrfachklassifizierung unterbinden. Die dritte Strategie hingegen geht einen sinnvollen Kompromiss zwischen Aufweichung der semantischen Beziehungen und restriktiver Anwendung der entsprechenden Regeln ein. Dadurch können auch Mehrfachklassifizierungen vorgenommen werden, die bei der ersten Strategie zurückgewiesen werden, so dass eine höhere Flexibilität entsteht. Aufgrund dieser höheren Flexibilität wird in dieser Ausarbeitung die dritte Strategie fa-

vorisiert. Dazu soll im Folgenden formal definiert werden, unter welchen Bedingungen eine Ausschlussbeziehung eines mehrfach-klassifizierten Objekts erfüllt wird:

Definition 4.8 (Mehrfachklassifizierung - Gültigkeitsbereich semantischer Beziehungen)

Gegeben sei eine Ontologie ONT = ($\mathcal{O}, R, \mathcal{R}$). $o \in \mathcal{O}$ sei ein mehrfach-klassifiziertes Objekt mit direkten Obertypen $p_1, ..., p_k \in \mathcal{O}$.
Eine Regel $r \in \mathcal{R}$ der Form excludes(o,t) mit o,t $\in \mathcal{O}$ gilt als erfüllt $\Longleftrightarrow$ $t \neq o \wedge$ es gilt nicht: subType(o,t)

Damit sind die Grundlagen der Mehrfachklassifizierung für Objekte gelegt. Tatsächlich wurde bisher nur von der Mehrfachklassifizierung von Objekten gesprochen. Durch die Unterscheidung von Objektkontexten über die Kontext-Ontologien muss auch auf die Frage der Mehrfachklassifizierung von Elementen dieser Kontext-Ontologien und möglicher daraus resultierender Konsequenzen eingegangen werden.

Wie bei mehrfach-klassifizierten Objekten steht ein mehrfach-klassifiziertes Element einer der drei Kontext-Ontologien in einer Untertypbeziehung zu einem oder mehreren anderen Elementen. Die Regelmenge eines mehrfach-klassifizierten Elementes einer dieser Ontologien ergibt sich wie beim Objekt aus der Vereinigung der Regelmengen der Obertypen. Bei mehrfach-klassifizierten Objekten besteht das Problem des Selbstausschlusses, da das Objekt selbst Ziel von geerbten Regeln sein kann. Bei Elementen der Kontext-Ontologien stellt sich dies anders dar. Die Elemente sind nicht Ziel von semantischen Beziehungen, sondern dienen lediglich der Abgrenzung verschiedener Anwendungssituationen der Objekte. Deshalb existiert das Problem des Selbstausschlusses für Elemente der Operation-, Subjekt- und Target-Ontologie nicht.

4.5.1. Inkonsistenz durch konträre Regeln

Eine weitere Art von Inkonsistenzen zwischen den Regeln verschiedener Vererbungspfade wird durch die Überlegung deutlich, dass beispielsweise bei mehrfacher Klassifizierung von einem Pfad die Regel `excludes(source,Object1)` und von einem anderen Vererbungspfad die Regel `dependsOn(source,Object1)` in die vereinigte Regelmenge vererbt werden kann. Es ist offensichtlich, dass die allgemeine Abhängigkeit nicht erfüllt werden kann, ohne gleichzeitig den Ausschluss zu verletzen. Es ist zu untersuchen, welche Möglichkeiten zum Umgang mit solchen Inkonsistenzen existieren. Dabei stehen folgende Strategien zur Diskussion:

- Es werden keine gesonderten Maßnahmen getroffen.
- Spätestens bei der Überführung in die Laufzeitstrukturen wird die Regelmenge auf Inkonsistenzen untersucht und Fehler gemeldet.

Die erste Strategie geht davon aus, dass eine Inkonsistenz in der Regelmenge eines Elements keine besondere Behandlung benötigt. Die Auswertung dieser Regelmenge führt automatisch immer zu einem semantischen Konflikt. Das Element und alle seine Untertypen können deshalb nie im Prozess innerhalb einer semantischen Aktivitätenbeschreibung verwendet werden. Dies hat zur Folge, dass auch die inkonsistente Regelmenge nie in den Prozess eingefügt wird. Aus der Sicht der semantischen Korrektheit ist diese Argumentation sinnvoll, da keine semantisch inkorrekten Prozesse durch semantische Inkonsistenzen innerhalb einer Regelmenge entstehen können. Doch stellt die praktische Unbenutzbarkeit der Elemente mit Sicherheit nicht das gewünschte Verhalten dar, sondern die Inkonsistenz ist ein Fehlverhalten, das beim Entwurf der Ontologie übersehen wurde. Dieser Fehler sollte erkannt werden, so dass die Fehlersituation bereinigt werden kann.
Die erstgenannte Möglichkeit ist also nicht für einen Praxiseinsatz tauglich. Daher kommt für eine Umsetzung nur die zweite Strategie in Frage, bei der die Regelmengen der Elemente auf Inkonsistenzen untersucht werden. Bei der Untersuchung auf inkonsistente Regeln wird das Zielobjekt jeder Regel aus der vereinigten Regelmenge mit den Zielobjekten der anderen Regeln verglichen, wobei insbesondere auf Obertypbeziehungen zu den Zielobjekten der zu vergleichenden Regeln geachtet werden muss. Wird das Zielobjekt bzw. ein Obertyp davon in einer der anderen Regeln gefunden, müssen die zwei Regeln näher untersucht werden. Dabei kann es zu folgenden Situationen kommen:

- Es handelt sich um unterschiedliche Typen von Regeln, also einerseits um eine `excludes`- und andererseits um eine `dependsOn`-Regel. In diesem Fall müssen die weiteren Attribute der Regeln untersucht werden, um die Inkonsistenz zu bestätigen oder das Ergebnis zu erhalten, dass die potentielle Inkonsistenz zwischen den Regeln keine tatsächliche Inkonsistenz darstellt. Die zweite Situation kann beispielsweise dann eintreten, wenn die Regeln einen unterschiedlichen Wert des `directionTarget`-Attributs besitzen, so dass z.B. die `dependsOn`-Regel beim Auftreten des Zielobjekts in der einen Richtung erfüllt wird, während der Ausschluss nur gilt, wenn das Zielobjekt in der anderen Richtung auftritt. In diesem Fall könnte die Abhängigkeitsbedingung erfüllt werden, ohne die Ausschlussbeziehung zu verletzen.
- Bei beiden Regeln handelt es sich um den gleichen Regeltyp. Zwei Situationen kommen dann in Frage:
  - Die Regeln sind vollständig identisch.
    $\Longrightarrow$ Es verbleibt nur eine der Regeln in der Regelmenge.
  - Die Regeln besitzen entgegengesetzte `directionTarget`-Attribute. In diesem Fall bleiben beide Regeln in der Regelmenge und müssen bei einem Korrektheitstest ausgewertet werden.

Zusammengefasst sind also lediglich Regeln innerhalb der selben Regelmenge problematisch, die das selbe Zielobjekt besitzen und eine Regel eine `excludes`-, die andere

eine `dependsOn`-Regel darstellt. Regeln des gleichen Typs können prinzipiell alle in der Regelmenge verbleiben.

`excludes`- und `dependsOn`-Regeln mit gleichem Zielobjekt innerhalb der Regelmenge eines Elements stellen also potentielle Konfliktherde dar. Deshalb muss geklärt werden, in welchen Fällen solche Regeln tatsächlich zu einer Inkonsistenz führen und in welchen Situationen beide Regeln in der Regelmenge existieren können. Zunächst wird nochmals aufgeführt, welche Bedeutung das `directionTarget`-Attribut einerseits im Zusammenhang mit einer `excludes`- und andererseits innerhalb einer `dependsOn`-Regel hat.

`directionTarget`: gibt die Richtung an, in der das Ziel-Objekt der Regel auftreten muss, um die Regel zu erfüllen:

- bei `excludes`-Regeln: Tritt das Ziel-Objekt in der angegebenen Richtung auf, kommt es zu einer verletzten Ausschlussbeziehung.
- bei `dependsOn`-Regeln: Tritt das Ziel-Objekt nicht in der angegebenen Richtung auf, so ist die Abhängigkeit nicht erfüllt.

Nun kann die Frage beantwortet werden, welche Attributkombinationen der zwei potentiell in Konflikt stehenden Regeln tatsächlich zu einer Inkonsistenz führen.

- `directionTarget`: entgegengesetzter Wert des `directionTarget`-Attributs in den Ausschluss- und Abhängigkeits-Regeln führt zu keinem Widerspruch und zu keiner Inkonsistenz zwischen den Regeln, da die `excludes`-Regel nicht ausschließt, dass die Abhängigkeit erfüllt wird.
- identischer Wert des `directionTarget`-Attributs bei beiden Regeln führt zu einem Widerspruch und damit zu einer Inkonsistenz zwischen den Regeln. Es ist offensichtlich, dass entweder die Ausschlussbeziehung verletzt wird oder die Abhängigkeitsbedingung unerfüllt bleibt. Gleiches gilt, wenn in keiner der beiden Regeln das `directionTarget`-Attribut gesetzt ist.
- `directionTarget`-Attribut nur in einer Regel gesetzt: zwei Fälle müssen unterschieden werden:
  - `directionTarget`-Attribut in der Ausschlussregel gesetzt:
    Die Abhängigkeitsbedingung kann erfüllt werden, ohne die Ausschlussbedingung zu verletzen, wenn das Zielobjekt in der entgegengesetzten Richtung zu der in der `excludes`-Regel vorgegebenen Richtung auftritt.
  - `directionTarget`-Attribut in der Abhängigkeitsregel gesetzt:
    Die Abhängigkeit kann nicht erfüllt werden, da die allgemeine `excludes`-Regel verletzt wird, wenn das Zielobjekt im Prozess auftritt, ohne dass für die Ausschlussbeziehung eine bestimmte Richtung vorgegeben ist.

Die zweite dargestellte Situation stellt also eine allgemeine Inkonsistenz dar, die gemeldet werden muss. Demgegenüber handelt es sich beim ersten Fall um eine situationsgebundene Inkonsistenz, die nur in einer bestimmten Anwendungssituation einen semantischen Konflikt verursacht. In diesem Fall wird keine Fehlermeldung durch eine inkonsistente Regelmenge erzeugt.

Die gewonnenen Erkenntnisse über Inkonsistenzen in den Regelmengen werden im nächsten Schritt formalisiert.

Definition 4.9 (Inkonsistenzen zwischen Regeln)

Gegeben sei eine Ontologie ONT = ($\mathcal{O}, R, \mathcal{R}$), sowie zwei Regeln r_1,$r_2 \in \mathcal{R}$. r_1 sei von der Form $r_1 = excludes(s, t[, directionTarget])$, r_2 von der Form
$r_2 = dependsOn(s, t[, directionTarget])$.
Es existiert eine Inkonsistenz zwischen r_1 und r_2 , wenn eine der folgenden Bedingungen erfüllt ist:

1. *r_1 = excludes(s,t) $\wedge$ r_2 = dependsOn(s,t)*
2. *r_1 = excludes(s,t,pre) $\wedge$ r_2 = dependsOn(s,t,pre)*
3. *r_1 = excludes(s,t,post) $\wedge$ r_2 = dependsOn(s,t,post)*
4. *r_1 = excludes(s,t) $\wedge$ r_2 = dependsOn(s,t,pre)*
5. *r_1 = excludes(s,t) $\wedge$ r_2 = dependsOn(s,t,post)*

Bei der Bestimmung von Konflikten zwischen Regeln wurde bisher davon ausgegangen, dass diesen Regeln der selbe Quellobjekt- und Zielobjekt-Kontext zugrunde liegt bzw. dass diese sich überlappen. Es ist offensichtlich, dass zwei Regeln nur dann auch tatsächlich in Konflikt stehen, wenn diese Bedingung der Fall ist. Sind die potentiell inkonsistenten Regeln für unterschiedliche Kontexte definiert, so hat die Inkonsistenz keinen realen Bestand. Die Regeln verbleiben in der Regelmenge. Dazu ein Beispiel:

Folgende zwei Elemente aus einer Operation-Ontologie seien gegeben. Zwischen Object2 und Object5 sei vorausgesetzt, dass keine Typbeziehung besteht:

Operation1
domain Object2
excludes Object1

Operation1
domain Object5
dependsOn Object1

$\Longrightarrow$ Die Ausschlussbeziehung wird ausgewertet, wenn Operation1 zusammen mit Object2 in einer Aktivität verwendet wird. Die Abhängigkeit hingegen ist dann zu erfüllen, wenn Operation1 zusammen mit Object5 innerhalb des selben Eintrags einer semantischen Aktivitätenbeschreibung genannt wird. Die beiden Regeln kommen also in unterschiedlichen

Anwendungssituationen des Ontologie-Elements zur Anwendung, so dass kein Konflikt besteht.

Nach der Untersuchung für den Quellobjekt-Kontext muss ein zweites Beispiel die angestellten Überlegungen für den Zielobjekt-Kontext nachvollziehen. Es wird davon ausgegangen, dass Operation1 und Operation2 in keiner Typbeziehung stehen:

Object1
excludes Object2
relatedOp Operation1

Object1
dependsOn Object2
relatedOp Operation2

$\Longrightarrow$ Der Ausschluss tritt in Kraft, wenn das Zielobjekt Object2 in einer Aktivität mit Operation Operation1 assoziiert ist. Die geforderte Abhängigkeit hingegen wird erfüllt, wenn eine Ausprägung von Object2 innerhalb einer Aktivität zusammen mit Operation2 verwendet wird. Aufgrund der fehlenden Typbeziehung zwischen Operation1 und Operation2 können die beiden Bedingungen nie durch das selbe Vorkommen von Object2 erfüllt werden. Die Regeln stehen damit nicht in Konflikt.

Aus diesen Überlegungen kann der Schluss gezogen werden, dass die Untersuchung zweier Regeln auf eine Inkonsistenz nur dann durchgeführt werden muss, wenn die in den Regeln angegebenen Kontexte übereinstimmen oder sich überlappen. Deshalb muss der Überprüfung der Regeln eine Untersuchung des Kontexts der Regeln vorangestellt werden, wie in Abbildung 4.30 verdeutlicht.

Neben den Problemen, die typischerweise von der Mehrfachklassifizierung verursacht werden, gibt es noch einige weitere Probleme im Zusammenhang mit der Verwendung verschiedener Ontologien und der Vererbung. Im nächsten Kapitel werden diese Probleme aufgegriffen.

4.6. Weitere Konfliktquellen

Nicht nur die Mehrfachklassifizierung ruft Situationen mit Selbstausschluss hervor. Auch auf einem Vererbungspfad, der lediglich auf einfacher Vererbung beruht, können solche Situationen auftreten. Ausgangspunkt der Überlegung stellt Abbildung 4.31 dar. Das Objekt `Object4` ist als Untertyp von `Object2` definiert und erbt damit die semantischen Beziehungen, die für `Object2` festgelegt sind. Eine dieser semantischen Beziehungen wird durch die Regel `excludes(Object2, Object4)` etabliert. Wird nun das Objekt `Object4` innerhalb einer Aktivität eines Prozesses verwendet, so wird ein durchgeführter Korrektheitstest für den Prozess einen semantischen Konflikt erkennen, da das Objekt `Object4` durch die vererbte Regel ausgeschlossen wird. Dieses Phänomen ähnelt stark dem in Kapitel 4.5 aufgetretenen Problem des Selbstausschlusses. In der Tat ist

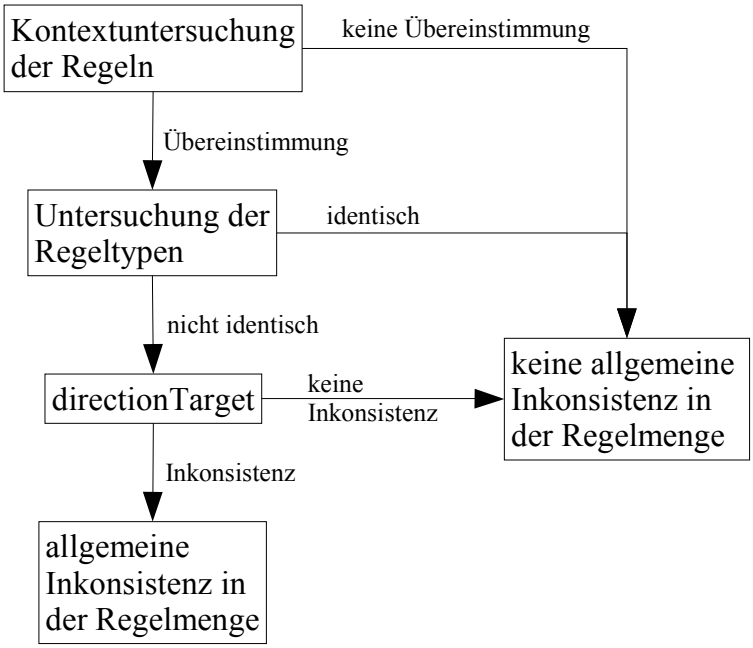


Abbildung 4.30.: Überprüfung von Regelmengen - Berücksichtigung des Kontexts

dieses Phänomen des Selbstausschlusses, wie es bei der Mehrfachklassifizierung auftritt, ein Spezialfall des gerade beschriebenen Sachverhalts. Deswegen gelten auch hier die selben Schlussfolgerungen wie beim Selbstausschluss infolge einer Mehrfachklassifizierung. Die Aussage von Definition 4.8 kann dementsprechend direkt auf diesen Anwendungsfall übertragen werden.

Ein weiteres Phänomen, das zu unerwarteten semantischen Konflikten führen kann, hat seinen Ursprung in der Orthogonalität der verschiedenen Ontologien. In einer semantischen Aktivitätenbeschreibung kann ein Objekt zusammen mit Elementen der Kontext-Ontologien, also Operation-, Subjekt- oder Target-Ontologie, in Verbindung gebracht werden. Zur Bestimmung der Regelmenge wird dann die Ontologie des zuständigen Kontext-Elements herangezogen. Die so erhaltene Regelmenge kann eine Ausschlussbeziehung enthalten, deren Zielobjekt mit dem Objekt übereinstimmt, mit dem das Ontologie-Element in der semantischen Aktivitätenbeschreibung assoziiert ist. Als Folge dieser Ausschlussbeziehung entsteht ein semantischer Konflikt. Ein so entstandener Konflikt deutet entweder auf eine fehlerhafte Assoziation des Objekts mit Elementen der Kontext-Ontologien hin, oder auf einen Fehler innerhalb der zur Bestimmung der Regelmenge ausgewerteten Ontologie. Es handelt sich dabei also nicht um ein dem Selbstausschluss gleichzusetzendes Phänomen. Um solche Fehlersituationen zu vermeiden, kann es sinnvoll sein, die semantischen Aktivitätenbeschreibungen nach deren Definition auf ein Fehlverhalten dieser Art zu überprüfen. Skizzenhaft dargestellt muss ein solcher Algorithmus die Regelmenge für jeden Eintrag der semantischen Aktivitätenbeschreibung

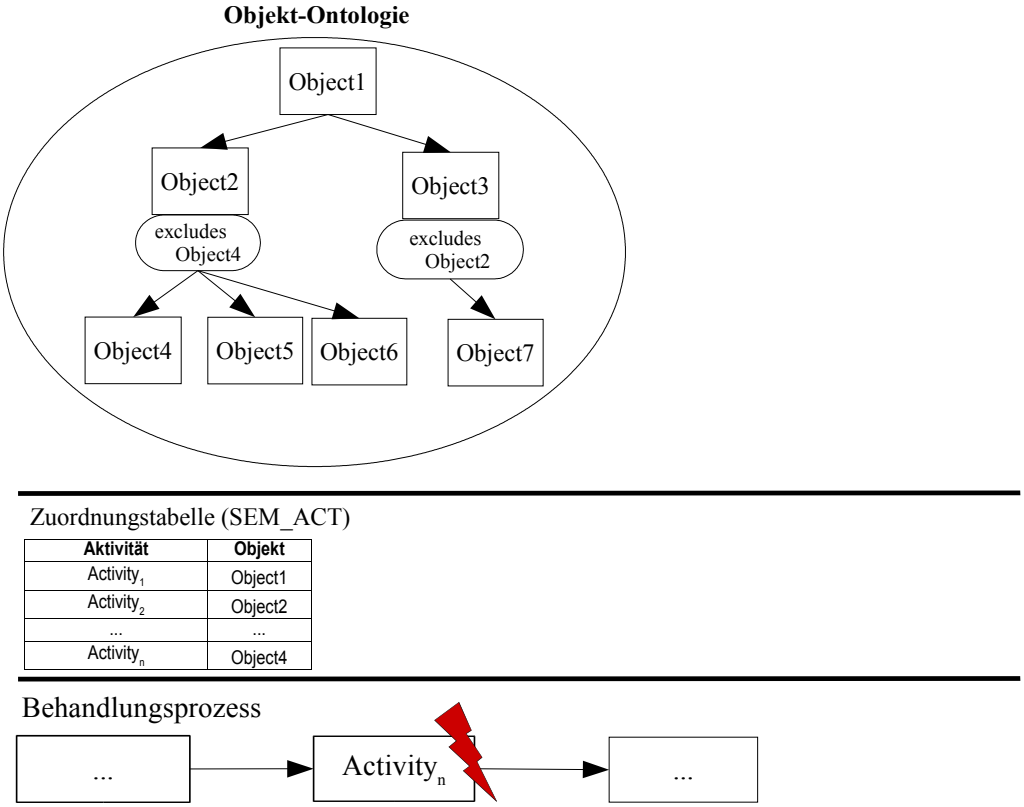


Abbildung 4.31.: Semantischer Konflikt innerhalb eines Vererbungspfades

bestimmen und die Regeln auf Ausschlussbeziehungen untersuchen, die durch das zugeordnete Objekt selbst verletzt werden.

Dieser skizzierte Algorithmus kann erweitert werden, um die identifizierten Ausschlussbeziehungen nicht nur mit dem in der Aktivitätenbeschreibung assoziierten Objekt abzugleichen, sondern auch um die Verletzung der Ausschlussbeziehung durch andere Objekte der Aktivitätenbeschreibung zu überprüfen. Dies ist lohnenswert, da somit Konflikte innerhalb von Aktivitäten, die zum unerwünschten Verhalten der Unbenutzbarkeit einer Aktivität führen, bereits bei der semantischen Beschreibung der Aktivitäten erkannt werden.

5

Änderungen auf Schemaebene und Ad-hoc-Änderungen

Sowohl Instanzen in der Form von Ad-hoc-Änderungen, als auch Prozesstypen, können Ziel von Änderungsoperationen sein. Es stellt sich im Zusammenhang mit der Semantik die Frage, welche Besonderheiten bei Ad-hoc-Änderungen beachtet werden müssen, und welche Auswirkungen die Änderung eines Prozessschemas aus dem Blickwinkel der semantischen Korrektheit auf die existierenden Instanzen hat.

Für alle im Folgenden angestellten Untersuchungen gilt, dass die Migration nur dann durchgeführt wird, wenn sie verträglich zum Kontroll- und Datenfluss des untersuchten Schemas und zur Markierung der jeweiligen Instanz ist, wie in Kapitel 2 beschrieben. Andernfalls ist die Änderung des Schemas nicht möglich. Ist eine Instanz nicht migrierbar, bleibt den Instanzen bis zu deren Beendigung das alte Schema zugeordnet.

5.1. Ad-hoc-Änderungen

Zuerst werden Änderungen auf der Instanz-Ebene näher beleuchtet. Beim Test der semantischen Korrektheit einer Ad-hoc-Änderung müssen folgende Situationen unterschieden werden:

- Die Instanz ist unverzerrt, d.h. sie besitzt keine Abweichungen gegenüber dem Prozessschema.
- Die Instanz ist verzerrt, d.h. sie weicht gegenüber dem Prozessschema ab.

Im ersten Fall dienen die semantischen Informationen, also Objekte, Regeln und Aktivitäten, des Prozessschemas als Grundlage der Auswertung. Im zweiten Fall müssen die spezifischen Abweichungen der Instanz in die Auswertung miteinbezogen werden. Nach der Durchführung des Korrektheitstests ist entweder ein semantischer Konflikt aufgetreten, oder der Korrektheitstest liefert ein positives Ergebnis für die Durchführung der Änderungsoperation. Tritt der Fall des semantischen Konflikts ein, kann im Zusammenhang mit einer Instanz-Änderung von einem semantischen Instanz-Konflikt gesprochen werden. Die Ad-hoc-Änderung kann in diesem Fall nicht durchgeführt werden.

Wurde der Korrektheitstest einer Instanz-Änderung erfolgreich abgeschlossen, so müssen die Änderungen in die Datenstrukturen überführt werden, die die Semantikinformationen einer Instanz beinhalten. Für eine geänderte Instanz gibt es dafür zwei mögliche Strategien:

- Redundante Speicherung der vollständigen semantischen Informationen des zugrundeliegenden Prozessschemas, so dass die durch die Instanzänderung notwendigen Anpassungen vorgenommen werden können.
- Speicherung der Abweichungen gegenüber den semantischen Informationen des Prozessschemas.

Der Vorteil der ersten Strategie liegt darin, dass die Auswertungsalgorithmen nicht abgeändert werden müssen. Anstatt auf die semantischen Informationen des Prozessschemas zuzugreifen, greifen sie nun auf die der Instanz zu. Ein Nachteil dieser Lösung ist jedoch, dass Speicher durch redundante Informationen verschwendet wird. Deshalb muss die zweitgenannte Möglichkeit genauer betrachtet werden. Die Abweichungen der Instanz ergeben, dass die Instanz einerseits Objekte und Regeln besitzen kann, die im Prozessschema fehlen, andererseits jedoch auch Objekte und Regeln in der Instanz gegenüber dem Prozessschema entfernt sein können. Diese Abweichungen können zu der Instanz beispielsweise in Form einer Liste, im Folgenden Delta-Liste genannt, gespeichert werden, deren Einträge jeweils die Art der Abweichung, also hinzugekommen (inserted), entfernt (removed) oder verschoben (moved), sowie die Informationen zum jeweiligen Objekt und zur zugeordneten Aktivität speichern. Damit wird gegenüber dem ersten Ansatz eine Optimierung auf der logischen Ebene erreicht. Abbildung 5.3 stellt am Beispiel eines Behandlungsprozesses die Ad-hoc-Änderung einer Instanz dar und verdeutlicht die Verwendung der Delta-Liste.

Die Auswertung der Korrektheit hat den in Abbildung 5.1 und Abbildung 5.2 dargestellten Ablauf. Bei der Löschoperation ist nur der 1. Teil durchzuführen.

Dieser Ablauf soll am Beispiel aus Abbildung 5.3 angewendet werden. Die erste Instanz-Änderung, also das Entfernen der Aktivität `Aspirin verabreichen`, läuft dann wie folgt ab:

- Instanz ist bisher unverzerrt. ⇒ Delta-Liste leer
- Auswertung der Änderung auf dem Schema: Das Entfernen hinterlässt keine unerfüllte Abhängigkeit und verursacht damit keinen semantischen Konflikt.

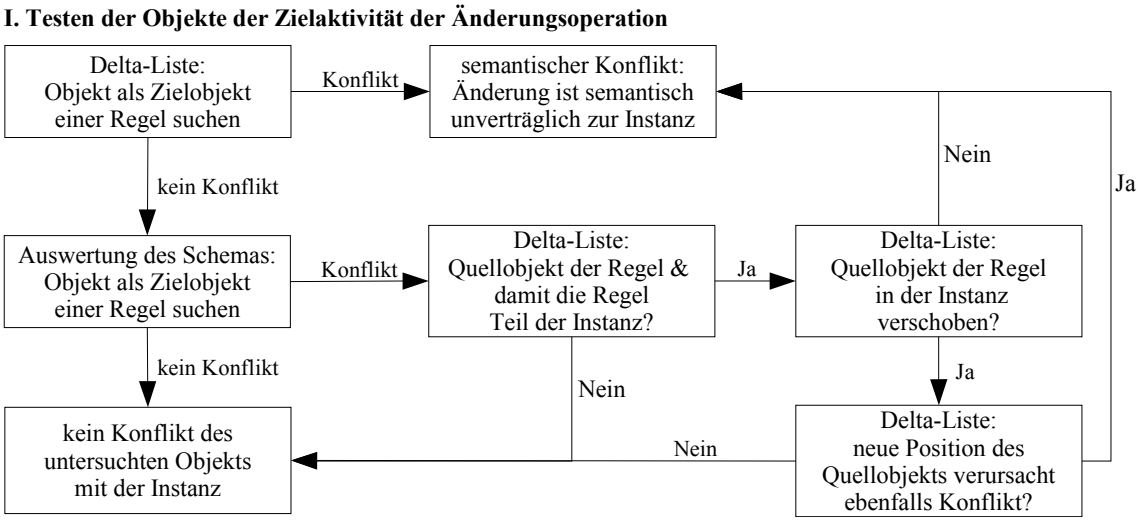


Abbildung 5.1.: Test 1. Teil: Objekte der Zielaktivität

- Änderung der Instanz semantisch korrekt.
- Informationen über die Änderungen an den semantischen Informationen in die Delta-Liste eintragen.

Die zweite Instanz-Änderung bezieht sich auf das Einfügen der Aktivität `Marcumar verabreichen`. Die Auswertung präsentiert sich dabei wie folgt:
1. Teil: Auswertung der Objekte der Aktivität `Marcumar verabreichen`

- Objekt `Marcumar` als Zielobjekt einer Regel in der Delta-Liste suchen. ⇒ Eintrag gefunden, der jedoch mit `removed` gekennzeichnet ist.
- Auswertung der Einfügeoperation auf dem Schema: `Marcumar` steht mit der Regel `excludes(Aspirin,Marcumar)` in einem Konflikt.
- Delta-Liste konsultieren: `Aspirin` findet sich in der Liste als Objekt und hat den Modus `removed`. Die Regel ist also nicht mehr in der Instanz vorhanden ⇒ `Marcumar` verursacht keinen Konflikt in der Instanz.

2. Teil: Auswertung der Regeln der Aktivität `Marcumar verabreichen`

- Regel ist excludes-Regel: excludes(Marcumar,Aspirin)
- Auswertung der Delta-Liste: Wird der Ausschluss durch ein Objekt der Liste verletzt?
  ⇒ Es liegt keine Verletzung der Ausschlussbedingung durch Einträge der Delta-Liste vor.
- Auswertung des Schemas: Ausschlussbedingung ist verletzt, da `Aspirin` im Prozessschema enthalten ist.

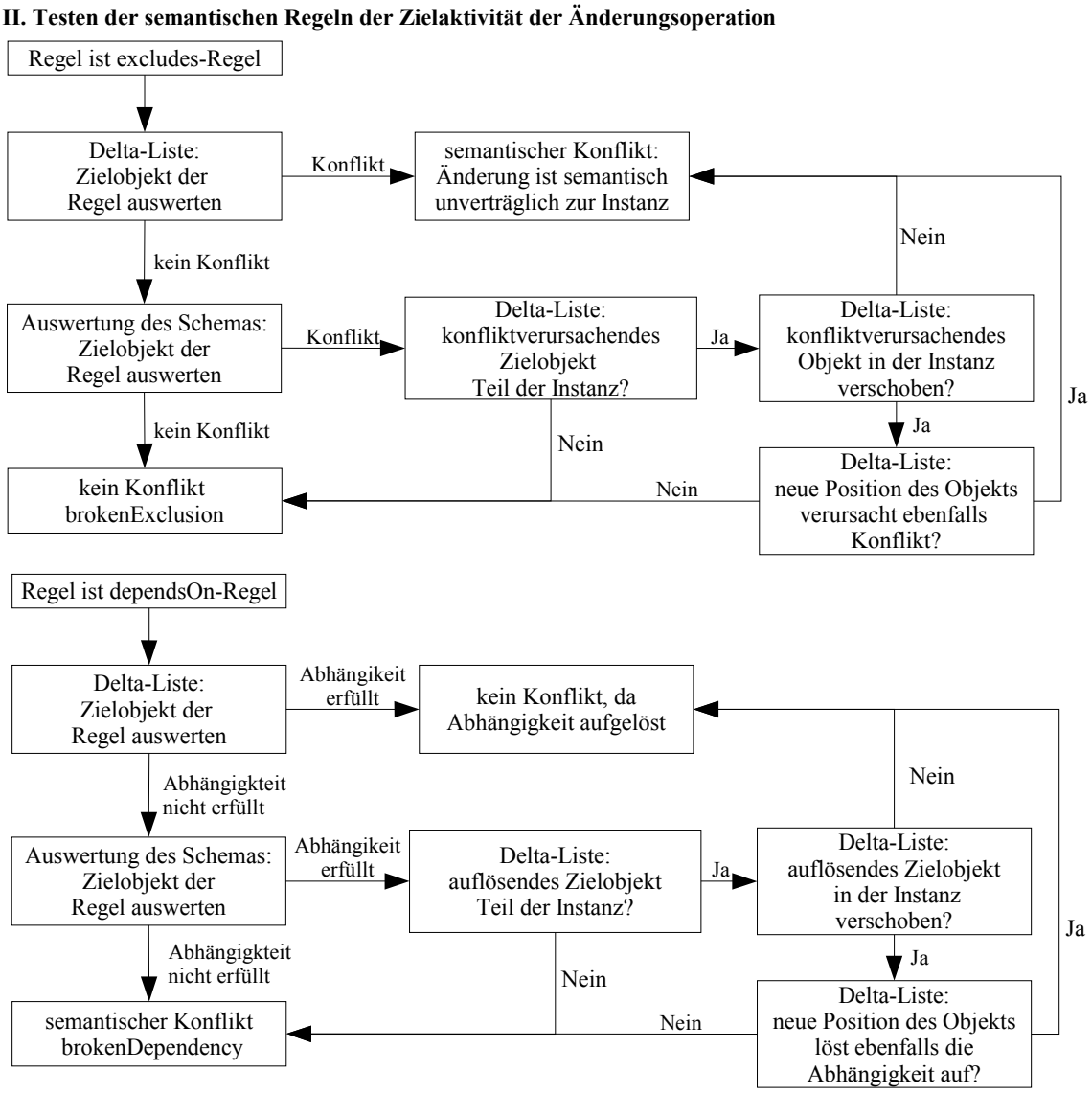


Abbildung 5.2.: Test 2. Teil: Regeln der Zielaktivität

- Delta-Liste konsultieren: `Aspirin` findet sich in der Liste und hat den Modus `removed`, ist also nicht mehr in der Instanz vorhanden.
  ⇒ `Marcumar` verursacht keinen Konflikt in der Instanz.

Nach der erfolgreichen Auswertung werden die für die Abweichung der Instanz notwendigen Einträge in der Delta-Liste vorgenommen.

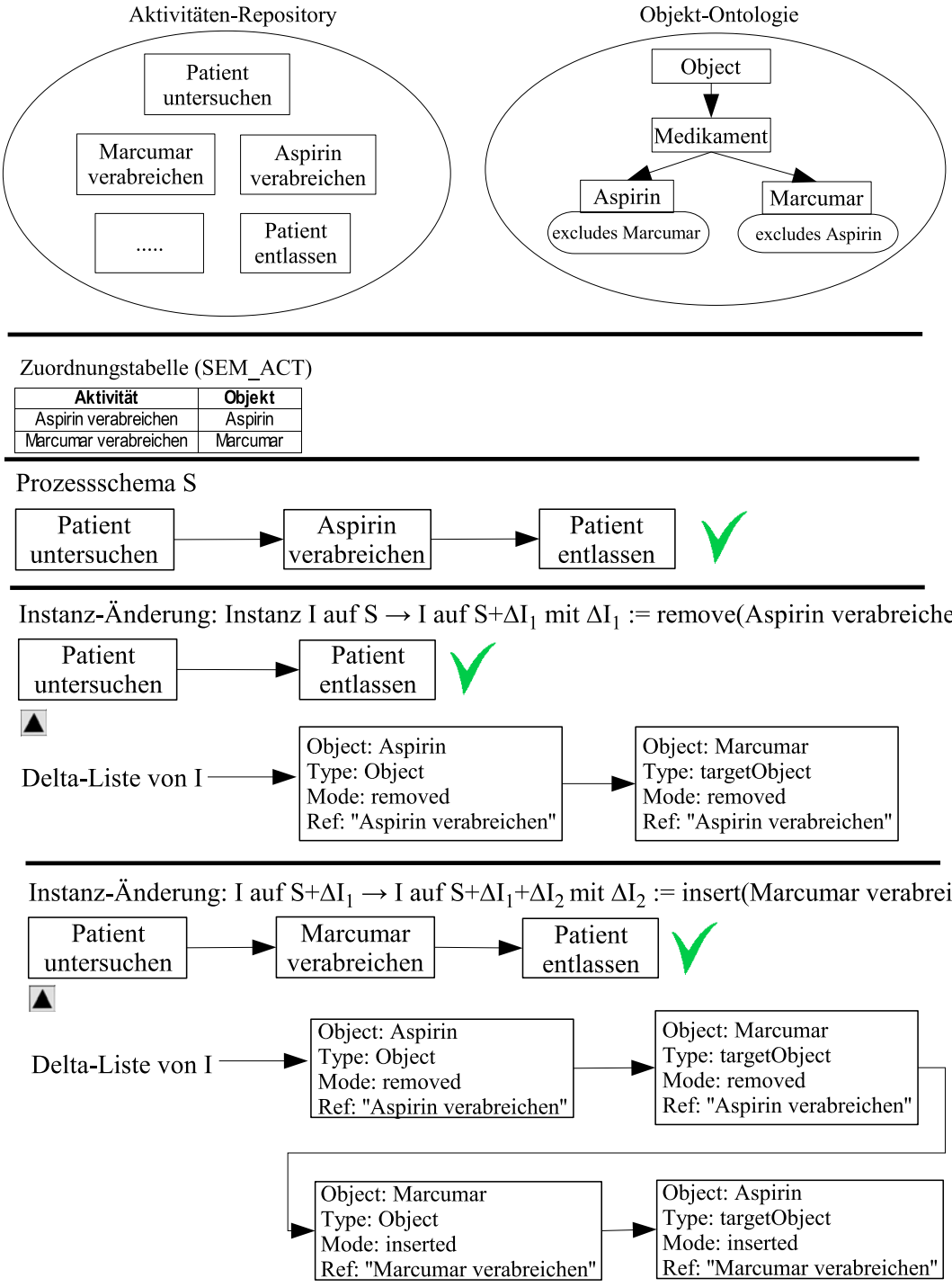


Abbildung 5.3.: Schema vs. Instanz

5.2. Prozesstyp-Änderungen

Das zweite mögliche Ziel von Änderungsoperationen ist das Prozessschema eines Prozesses. Zu einem Prozessschema werden sämtliche relevanten semantischen Informationen gespeichert. Diese Daten dienen wie bisher dem Korrektheitstest als Grundlage für die Entscheidung, ob die Prozessschema-Änderung semantisch korrekt ist oder einen semantischen Konflikt verursacht. Wird die Änderung als semantisch unkorrekt zurückgewiesen, wird von einem semantischen Schema-Konflikt gesprochen. In diesem Fall bleibt das Prozessschema ungeändert. Somit ergeben sich keine Auswirkungen auf die Instanzen des Prozesses. Tritt jedoch der Fall ein, dass die Änderung als semantisch korrekt eingestuft wird, so kann das Prozessschema entsprechend abgeändert werden. In Folge dieser Änderung des Schemas muss die Migrierbarkeit der Instanzen auf das neue Schema geprüft werden. Aus der Sicht der Semantik muss also für jede Instanz geprüft werden, ob sie semantisch verträglich zur Änderung des Prozessschemas ist, so dass die Änderung in der Instanz keinen semantischen Konflikt verursacht. Beim Test auf Migrierbarkeit muss zwischen folgenden Möglichkeiten unterschieden werden.

- Eine untersuchte Instanz ist unverzerrt.
- Eine untersuchte Instanz ist verzerrt.

Im ersten Fall muss kein weiterer Korrektheitstest für die Instanz durchgeführt werden. Sie ist wie das Schema semantisch verträglich zu der Änderung, so dass die Instanz auf das modifizierte Schema migriert werden kann.
Im zweiten Fall ist jedoch ein weiterer Korrektheitstest für die vom ursprünglichen Schema abweichende Instanz notwendig. Dieser Test führt entweder zu dem Ergebnis, dass die Instanz verträglich zu der Änderung ist, oder es wird ein semantischer Konflikt erkannt. War das Ergebnis des Tests positiv, so kann die Instanz aus der Sicht der Semantik auf das neue Schema migriert werden. Wurde andernfalls ein semantischer Konflikt verursacht, so ist die Migration auf das neue Prozessschema nicht möglich. Ein so entstandener Konflikt kann als semantischer Schema-Instanz-Konflikt bezeichnet werden.

5.3. Semantische Verträglichkeit und Markierung

Es fällt auf, dass beim Test der semantischen Korrektheit einer Instanz nicht auf den aktuellen Zustand einer Instanz eingegangen wird. Die Begründung liefert folgendes Beispiel, in dem eine Aktivität eingefügt werden soll, die Object1 in ihrer semantischen Aktivitätenbeschreibung enthält:

Regel:
dependsOn(Object1,Object2,post)

Diese Regel sagt aus, dass Object1 von Object2 abhängt, und zwar unter der Bedingung, dass Object2 im Prozess nach Object1 auftritt. Die Instanz sei bereits über die Aktivität hinaus fortgeschritten, die Object2 in der semantischen Aktivitätenbeschreibung enthält. Aus der statischen Sicht des Prozesses ist die Abhängigkeit erfüllt, wenn die Aktivität vor der Aktivität, der Object2 zugeordnet ist, eingefügt wird. Aus der Sicht des aktuellen Fortschritts der Instanz jedoch nicht, da der Attributwert post die Bedeutung hat, dass Object2 in der Ausführung nach Object1 liegt. Da die Aktivität mit zugeordnetem Objekt Object2 bereits ausgeführt wurde, ist diese Bedingung verletzt. Andererseits kommt die Aktivität mit zugeordnetem Object1 ebenso nicht mehr zur Ausführung, so dass der semantische Konflikt in der Realität nicht eintreten kann. Dies wäre lediglich unter der Berücksichtigung von Schleifen möglich. In diesem Fall würde jedoch, wenn die statische semantische Korrektheit nachgewiesen wurde, zuerst die Aktivität mit zugeordnetem Objekt Object1 in einer weiteren Iteration ausgeführt werden und dann die Aktivität mit assoziiertem Objekt Object2. Die Bedingung der Regel wäre korrekt erfüllt.

6
Effizienzaspekte

Nachdem die theoretischen Grundlagen für die Semantik in Prozessen und PMS gelegt wurden, wird in diesem Teil der Arbeit untersucht, ob diese Konzepte effizient umsetzbar sind. Dazu müssen die performance-kritischen Punkte identifiziert werden. Darauf aufbauend werden Konzepte entwickelt, um diese potentiellen Performance-Engpässe zu umgehen.

Als erstes werden die Korrektheitstests näher beleuchtet. Dabei ist zu untersuchen, an welchen Stellen in deren Ablauf die schwerwiegendsten potentiellen Performance-Engpässe liegen. Diesen Betrachtungen liegen folgende Abläufe zugrunde:

- Einfüge-Operation:
  1. Prüfen der Objekte der einzufügenden Aktivität, ob sie eine Regel des Prozesses erfüllen.
  2. Prüfen aller Regeln der einzufügenden Aktivität, ob sie durch Objekte im Prozess erfüllt werden.
  3. Einfügen der Aktivität mit ihren Objekten und Regeln in den Prozess.
- Lösch-Operation:
  1. Prüfen der Objekte der zu löschenden Aktivität, ob sie Zielobjekte von Abhängigkeitsbeziehungen anderer Aktivitäten des Prozesses sind und diese Abhängigkeiten aufgrund ihrer Positionen erfüllen.
  2. Bleibt keine unerfüllte Abhängigkeit zurück, entferne die Aktivität mit ihren Objekten und deren Regeln aus dem Prozess.

- Die Verschiebe-Operation stellt wie bereits beschrieben eine Komposition aus Lösch- und Einfügeoperation dar.

Beim Vergleich der skizzierten Abläufe ist zu erkennen, dass die Einfüge-Operation aufwendiger ist als die Lösch-Operation. Aufwendig bei der Löschoperation ist die Suche der Objekte der zu löschenden Aktivität als Zielobjekte von Regeln des Prozesses. Alle Regeln des Prozesses müssen dabei durchsucht werden. Dies deckt sich mit dem ersten Schritt bei der Einfüge-Operation. Zusätzlich müssen beim Einfügen die eigenen Regeln der einzufügenden Aktivität auf mögliche semantische Konflikte untersucht werden.
Die Einfügeoperation bietet sich deshalb an, um zu analysieren, an welchen Stellen Optimierungsansätze angebracht sind. Für das Einfügen lassen sich folgende performancekritische Punkte ausmachen:

- Die Objekte der einzufügenden Aktivität können Zielobjekte der Regeln des Prozesses sein. Um dies festzustellen, müssen die Regeln des Prozesses durchsucht werden. Dabei muss darauf geachtet werden, dass ein Objekt der einzufügenden Aktivität potentiell auch Regeln erfüllt, deren Zielobjekte Obertypen darstellen.
- Die auszuwertende Regelmenge für jedes Objekt der einzufügenden Aktivität muss bestimmt werden.
- Die Regeln aller Objekte der einzufügenden Aktivität müssen ausgewertet werden. Dazu müssen die Objekte des Prozesses durchsucht werden, um festzustellen, ob die Zielobjekte der Regeln im Prozess auftreten. Es muss darauf geachtet werden, dass Regeln der einzufügenden Aktivität auch durch Untertypen des festgelegten Zielobjekts erfüllt werden können.

Daraus ergeben sich drei Ansatzpunkte, an denen Optimierungen vorgenommen werden müssen.

- Typbeziehungen müssen möglichst effizient erkannt werden, d.h. bei der Auswertung zweier Objekte muss ohne großen Aufwand feststellbar sein, ob eine Obertyp-Untertyp-Beziehung zwischen ihnen besteht.
- Die in einem Prozess enthaltenen Objekte müssen effizient organisiert werden, damit eine schnelle Suche nach einem bestimmten Objekt möglich ist. Das selbe gilt für die Organisation der Zielobjekte der semantischen Beziehungen.
- Die Umsetzung der Ontologien, so dass die einzelnen Elemente zugriffseffizient organisiert sind.

6.1. Laufzeituntersuchungen und Optimierungsansätze

Die Einführung eines Typsystems zur möglichst allgemeingültigen Formulierung von semantischen Beziehungen bringt, wie in Kapitel 4 diskutiert, viele Vorteile mit sich. Diese

Vorteile werden jedoch dadurch erkauft, dass zur Laufzeit die Typbeziehungen ausgewertet werden müssen. Man kann sich leicht vorstellen, dass bei sehr tiefen Hierarchien der Laufzeitaufwand zur Bestimmung einer möglichen Typbeziehung zwischen zwei Objekten sehr groß sein kann, wenn die Implementierung nicht entsprechend entgegenwirkt. Im Folgenden werden verschiedene Ansätze diskutiert, wie die Bestimmung von Typbeziehungen möglichst effizient umgesetzt werden kann. Dabei müssen die Alternativen unter verschiedenen Kriterien betrachtet werden:

- Stabilität des Laufzeitaufwands unabhängig von der Tiefe der Ontologie.
- Linearität des Speicherbedarfs unabhängig von der Größe der Ontologie.

Das erste Kriterium ist wichtig, da Änderungsoperationen zur Laufzeit eines Prozesses nicht beliebig viel Zeit in Anspruch nehmen dürfen. Es muss eine klare Aussage getroffen werden können, wie lange die Bestimmung auf Typzugehörigkeit dauert.
Das zweite Bewertungskriterium beurteilt die einzelnen Strategien danach, wie sich der Speicherbedarf bei steigender Größe einer Ontologie verhält. So ist es wünschenswert, dass der Speicherbedarf möglichst linear zur Anzahl der Ontologie-Elemente ansteigt, der Speicherbedarf für die Typinformation eines Ontologie-Elements also konstant bleibt. Auch dieses Kriterium ist leicht einzusehen, da vermieden werden muss, dass mit wachsender Größe eines Typsystems die Speicheranforderungen explosionsartig ansteigen.

In erster Näherung wird vom naiven Vorgehen ausgegangen, um Vergleichswerte für spätere Überlegungen zu erhalten. Das naive Vorgehen basiert dabei auf der Annahme, dass Typbeziehungen nur zwischen zwei direkt in einer Obertyp-Untertyp-Beziehung stehenden Elementen erkannt werden können.

6.1.1. Laufzeit beim naiven Verfahren

Die Untersuchung des Laufzeitverhaltens des naiven Verfahrens liefert einen Vergleichswert für die spätere Untersuchung effizienterer Verfahren.

Gegeben sei die Domäne Dom = ($\mathcal{N}$,$\mathcal{O}$, $\mathcal{R}$, SEM_ACT, SEM_RELS). Aktivität Act_{n+1} soll in einen Prozess mit dem Schema S = (N,E,D,Dom) eingefügt werden.
TARGETOBJECT: $\mathcal{R} \rightarrow \mathcal{O}$ liefere das Zielobjekt einer Semantikregel.
Der Ablauf beim Einfügen der Aktivität Act_{n+1} besteht grundlegend aus folgenden Schritten:

- Schritt 0: Instanz(en) des Prozesses sperren.
- Schritt 1: Verträglichkeitstest (Semantik)
  `foreach object in` $SEM\_ACT(Act_{n+1})$ `do{`
  (i) Test: Ist `object` das Ziel einer Regel des Prozesses?
  (ii)

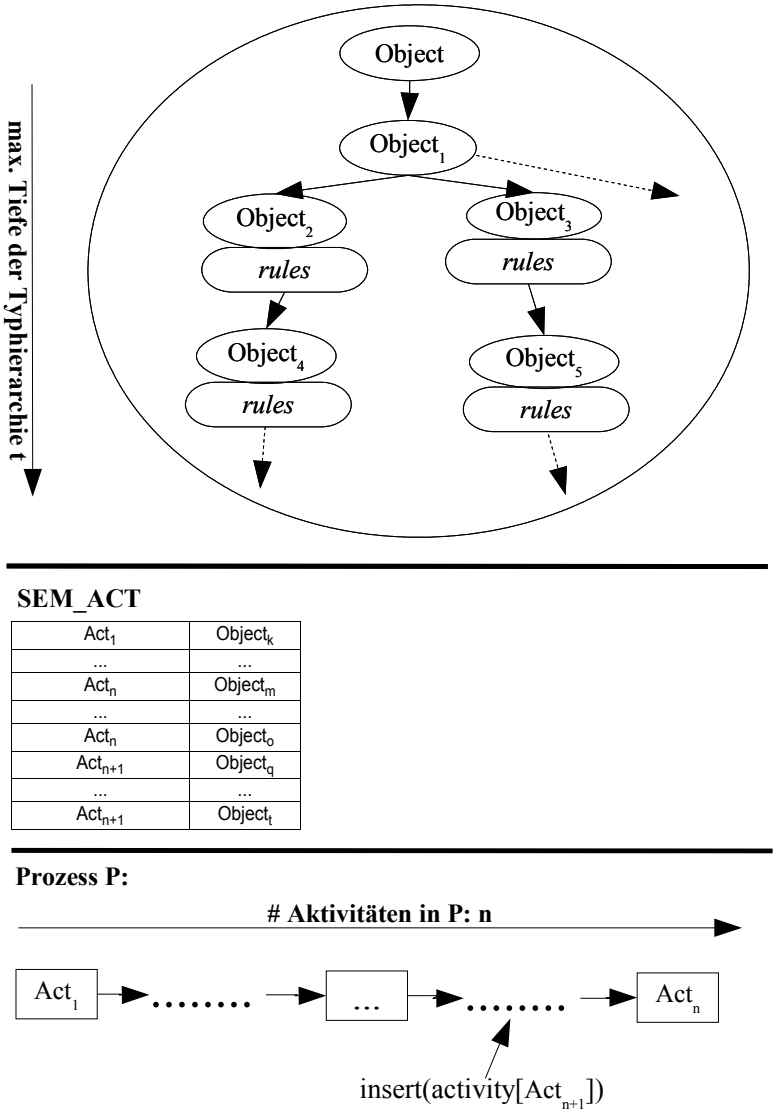


Abbildung 6.1.: Semantische Beziehungen zwischen Objekten (mit Typbeziehungen)

(a) Bestimme die Regelmenge für `object`.
(b) Prüfe jede der Regeln auf Konflikte mit Objekten, die in den Aktivitäten des Prozesses vorhanden sind.

}

- Schritt 2: Einfügen der Aktivität nach erfolgreichem Verträglichkeitstest.
- Schritt 3: Einfügen der Regeln und Objekte der neuen Aktivität.
- Schritt 4: Instanz(en) wieder freigeben.

Besonderes Interesse für das Laufzeitverhalten stellt in diesem Kapitel der Verträglichkeitstest aus Schritt 1 dar. Deshalb sind die Teilschritte von Schritt 1 näher zu untersuchen. Dazu werden sie zunächst in Pseudocode ausgearbeitet, um dann eine Aussage über ihr Laufzeitverhalten anzustellen. Die Funktion `computeSubtypes` stelle alle Subtypen eines übergebenen Typs inklusive des Typs selbst in einer Menge zur Verfügung.

```
Schritt 1.(i):
foreach object in SEM_ACT(Act_{n+1}) do {
   foreach act in N do{
      foreach obj in SEM_ACT(act) do{
            foreach rule in SEM_RELS(obj) do{
                    TargetTypeSet tts=
                      computeSubtypes(TARGETOBJECT(rule));
                    foreach targetType in tts do{
                            compare(object,targetType);
                    }
            }
      }
   }
}

Schritt 1.(ii):
foreach object in SEM_ACT(Act_{n+1}) do {
   foreach rule in SEM_RELS(object) do{
        foreach act in N do{
            foreach obj in SEM_ACT(act) do{
                    TargetTypeSet tts=
                      computeSubtypes(TARGETOBJECT(rule));
                    foreach targetType in tts do{
                            compare(obj,targetType);
                    }
            }
        }
   }
}
```

Kritisch für das Laufzeitverhalten eines Algorithmus sind mehrfach geschachtelte Schleifenkonstruktionen. Alle der einzufügenden Aktivität zugeordneten Objekte und alle Objekte der Aktivitäten des Prozesses müssen durchlaufen werden. Hinzu kommt die Untersuchung der Untertypen. Insgesamt werden in diesem Fall fünf ineinandergeschachtelte Schleifen benötigt.
a sei die Anzahl der Objekte, die die einzufügende Aktivität semantisch beschreiben, m die durchschnittliche Anzahl an zugeordneten Objekten je Aktivität des Prozesses, n die Anzahl der Aktivitäten im Prozess, r die maximale Anzahl an Regeln pro Objekt

und t die Tiefe der Typhierarchie. Damit ergibt sich schlimmstenfalls eine abgeschätzte Laufzeitsituation mit 2*a*n*m*r*t Schleifendurchläufen.

Die Definition der Semantik einer Aktivität über die ihr zugeordneten Objekte und die Einordnung von Objekten in ein Typsystem haben viele Vorteile, die jedoch mit einem schlechten Laufzeitverhalten des Korrektheitstests erkauft werden. Scheinbar besteht ein "Trade-Off" zwischen der Ausdrucksmächtigkeit der Mechanismen zur Festlegung der Semantik und wünschenswerten Laufzeiteigenschaften auf der anderen Seite. Daraus leitet sich die zentrale Fragestellung ab, die es im weiteren zu beantworten gilt:

Können die Auswertungsmechanismen, die auf Objekten und Typbeziehungen zwischen diesen Objekten aufbauen, optimiert werden, um die Nachteile, die sich dadurch im Laufzeitverhalten ergeben, abzuschwächen bzw. zu kompensieren?

6.1.2. Optimierte Ansätze

Zur Beantwortung der genannten Frage steht in einem ersten Schritt die Bestimmung der Objektbeziehungen im Mittelpunkt der Untersuchungen. Es sei angemerkt, dass die Überlegungen auch für die Elemente der Kontext-Ontologien Gültigkeit besitzen, da der grundlegende Aufbau aller Ontologien identisch ist. Deshalb genügt es, um die Ausführungen möglichst einfach und klar zu halten, wenn von der Objekt-Ontologie und den Objekten ausgegangen wird. Bei den folgenden Ansätzen wird zunächst die Problematik der Mehrfachklassifizierung außer acht gelassen und vom Prinzip der einfachen Klassifizierung ausgegangen. Dieses Vorgehen soll verhindern, dass unnötig von der Untersuchung der dargestellten Konzepte abgelenkt wird.

Eine effiziente Bestimmung von Typbeziehungen liegt dann vor, wenn nur wenige Vergleiche zwischen den auszuwertenden Objekten notwendig sind, um eine definitive Aussage treffen zu können. Eine Grundvoraussetzung zur effizienten Bestimmung von Typbeziehungen ist die Zuordnung von Typinformation zu einem Objekt, die in Form einer Typkennung umgesetzt werden kann. Die Objekte müssen also neben ihrem Namen diese eindeutige Typkennung besitzen, die so aufgebaut ist, dass über alle Vererbungsebenen hinweg eine Aussage über Typbeziehungen möglich wird. Für die Umsetzung einer solchen Typkennung existieren verschiedene Möglichkeiten, die anhand ihrer Vor- und Nachteile zu bewerten sind.

6.1.2.1. Strings als Mittel zur Umsetzung einer Typkennung

Zur Realisierung einer Typkennung kann ein hierarchisch aufgebauter String verwendet werden. Der Aufbau dieses Strings kann mit dem regulären Ausdruck $(0..9)^{+}[(-)(0..9)^{+}]^{*}$ beschrieben werden. Mit jeder Hierarchieebene des Typsystems kommt eine Stufe im String hinzu, jeweils markiert durch ein Trennzeichen, in diesem Fall dem Zeichen `"-"`, wobei die erste Hierarchieebene nur eine Stufe enthält. Die Elemente, die Untertypen des

selben Obertyps sind, werden innerhalb einer Hierarchieebene der Ontologie durchnummeriert, indem die Zahl in der letzten Stufe des Kennungsstrings inkrementiert wird. Ein Untertyp erbt also die Kennung seines Obertyps, wobei an die Kennung des Obertyps eine weitere Stufe angehängt wird. Jeder Untertyp erhält damit eine eindeutige Kennung. Die Kennung eines Typs beginnt immer mit allen Zeichen, die die Kennung eines jeden Obertyps auf dem Vererbungspfad ausmachen. Mit einem einfachen String-Vergleich der Kennungen zweier Objekte ist dann ermittelbar, ob zwei Objekte in einer Typbeziehung stehen oder nicht. Abbildung 6.2 erweitert die Elemente der bekannten Medikamentenontologie um Typkennungen, die auf dem Prinzip des hierarchisch aufgebauten Strings beruhen. Zur Verdeutlichung der Auswertung von Typbeziehungen mit

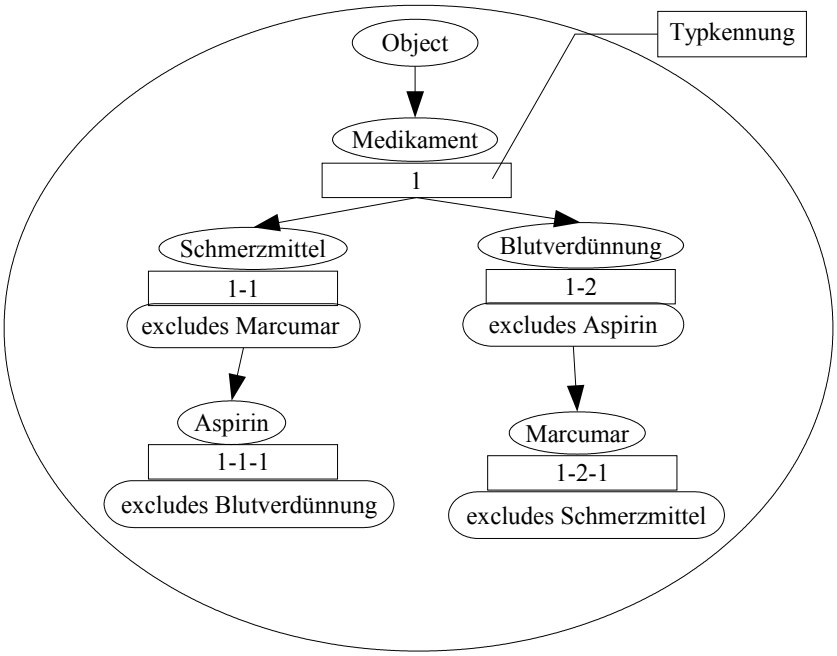


Abbildung 6.2.: Hierarchischer String als Typkennung

Hilfe dieser Typkennung soll ein Beispiel herangezogen werden. Anhand der Ontologie aus Abbildung 6.2 ist folgende Typuntersuchung durchzuführen:

Ist `Aspirin` Untertyp von `Schmerzmittel`?
$\Longrightarrow$ Kennung `Schmerzmittel`: 1-1
$\Longrightarrow$ Kennung `Aspirin`: 1-1-1

Es muss ein Algorithmus entwickelt werden, der zwei Kennungen vergleicht und als Ergebnis die Antwort auf die Frage nach einer Typbeziehung zurückgibt. Der Algorithmus ist in folgendem Pseudocode-Fragment beispielhaft umgesetzt:

```
boolean isSubtype(String subTypeID, String superTypeID){
```

```
    if (subTypeID.startsWith(superTypeID))
            return true;
    else return false;
}
```

Die aufgeworfene Frage nach der Untertypbeziehung zwischen `Aspirin` und `Schmerzmittel` kann dann durch den einfachen Funktionsaufruf `isSubtype(”1-1-1”,”1-1”)` beantwortet werden. Die Funktion liefert `true` zurück. `Aspirin` ist also Untertyp von `Schmerzmittel`, was dem erwarteten Ergebnis entspricht.

Dieser Ansatz erfüllt die Forderung, dass ein Typzusammenhang über möglichst wenige Vergleiche und Schleifendurchläufe bestimmt werden kann. In der Tat wird bei der Realisierung einer Typkennung über den hierarchisch aufgebauten String nur ein Vergleich beider Strings benötigt und das Durchsuchen der Vererbungspfade entfällt.
Die zweite Forderung nach konstantem Speicherplatzbedarf, d.h. jedes Element soll die selben Speicheranforderungen zur Speicherung der Kennung besitzen, wird jedoch durch dieses Konzept verletzt. Der String wächst mit jeder Hierarchieebene der Ontologie. In sehr großen Ontologien mit extrem tiefen Hierarchien kann dies problematisch werden.

Für eine zusammenfassende Bewertung des vorgestellten Verfahrens bleibt festzuhalten, dass es sich um ein sehr einfach zu realisierendes Verfahren handelt und die Bestimmung von Typbeziehungen zulässt, ohne die gesamte Typhierarchie durchsuchen zu müssen. Aufgrund der Nachteile eignet sich das Verfahren jedoch eher für kleinere Ontologien ohne extrem tiefe Hierarchien.

6.1.2.2. Primzahlen als Mittel zur Umsetzung einer Typkennung

Der nächste Ansatz zum Aufbau einer eindeutigen Typkennung beruht darauf, dass jede natürliche Zahl als Produkt von Primfaktoren dargestellt werden kann [MRW92]. Bei der vollständigen Division einer Zahl durch einen bekannten Primfaktor, d.h. einer Division ohne Rest, ergibt sich, dass dieser Primfaktor atomarer Teil der Produktion der Primfaktoren dieser Zahl ist. Primzahlen und Primfaktoren können wie folgt eingesetzt werden, um eine eindeutige Typkennung umzusetzen.

Jeder Typ einer Ontologie erhält eine eindeutige Primzahl zugewiesen, d.h. jede Primzahl wird nur einmal an genau einen Typ vergeben. Diese einem Typ zugeordnete Primzahl wird als weiterer Faktor zu einem Produkt aus Primzahlen hinzugefügt, das vom Obertyp geerbt wird und im Typ als Typkennung gespeichert wird. Jeder Untertyp enthält also ein Produkt aller Primfaktoren aus dem kompletten Vererbungspfad von der Wurzel bis zum Objekt selbst. Dieses Produkt ist durch alle Primfaktoren des Typpfades teilbar.
Soll nun ein Vergleich zweier Objekte vorgenommen werden, um festzustellen, ob diese in einer Typbeziehung stehen, so wird das Produkt des potentiellen Untertyps durch die eindeutige Primzahl des potentiellen Obertyps geteilt. Ist diese Division ohne Rest möglich, so besteht eine Untertypbeziehung. Dies stellt sich folgendermaßen dar:

Dem Element e ist als Typkennung das Produkt $P = p_1 * ... * p_i$ zugeordnet, einem weiteren Element o die Primzahl k. e ist Untertyp von o $\Longleftrightarrow$ k=p_j mit 1≤j≤i.

Abbildung 6.3 erweitert die bekannte Medikamentenontologie um Typkennungen für die Elemente, die auf dem beschriebenen Verfahren beruhen. Zur Verdeutlichung soll

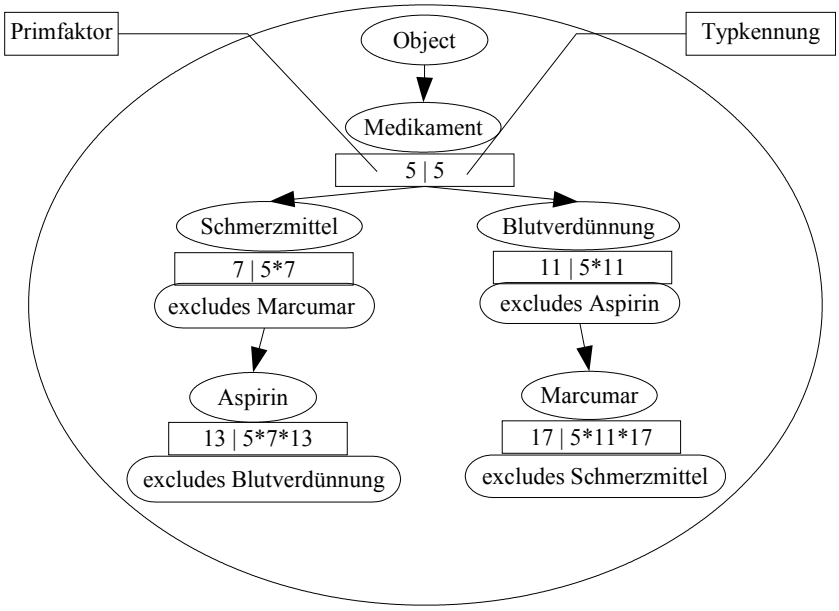


Abbildung 6.3.: Primfaktorprodukt als Typkennung

wiederum ein Beispiel benutzt werden, dass auf der Ontologie aus Abbildung 6.3 beruht. Es soll folgende Typbeziehung untersucht werden:

Ist `Aspirin` Untertyp von `Schmerzmittel`?
$\Longrightarrow$ Primfaktor von `Schmerzmittel`, dem potentiellen Obertyp: 7
$\Longrightarrow$ Kennung von `Aspirin`, dem potentiellen Untertyp: 455 (= 5*7*13)

Das Codefragment, das die Untersuchung auf eine Typbeziehung zwischen zwei Objekten ermöglicht, ist im Folgenden in Pseudocode dargestellt:

```
boolean isSubtype(int primfactor, int product){
    if (product mod primfactor == 0)
            return true;
    else return false;
}
```

Die Frage nach der Typbeziehung zwischen `Aspirin` und `Schmerzmittel` wird durch den Funktionsaufruf `isSubtype(7,455)` beantwortet. In diesem Fall ist das Ergebnis

des Aufrufes `true`, womit die Erwartung, dass `Aspirin` Untertyp von `Schmerzmittel` ist, bestätigt wird.

Dieser Ansatz muss anhand der festgesetzten Kriterien auf seine Praxistauglichkeit untersucht werden. Wie gefordert ermöglicht der Ansatz eine effiziente Untersuchung von möglichen Typbeziehungen zwischen Objekten. Statt die vollständige Typhierarchie zu durchsuchen, genügt eine Modulooperation um festzustellen, ob ein Typ Untertyp eines zweiten Typs ist.
Das Kriterium der stabilen Speicherplatzanforderung wird ebenfalls durch dieses Konzept erfüllt, da für jeden Typ immer genau zwei Zahlen gespeichert werden müssen, egal auf welcher Hierarchieebene sich der Typ befindet.

Trotz der Erfüllung der notwendigen Kriterien besitzt dieser Ansatz Nachteile, die einen sinnvollen Praxiseinsatz nicht ermöglichen. Die Primzahlen zur Zuweisung an die Typen müssen entweder aufwendig berechnet werden, oder über vorberechnete Primzahltabellen zugreifbar sein. Beinahe noch schwerer wiegt jedoch das Problem, dass die Produkte aus den Primfaktoren schnell Größen annehmen, die eine Speicherung in einem Integertyp nicht mehr ermöglichen.

6.1.2.3. Adjazenzmatrix zur Realisierung von Typbeziehungen

Beim im Folgenden beschriebenen Ansatz handelt es sich um eine graphentheoretische Herangehensweise an das Problem der Typbeziehungen. Ein graphentheoretischer Ansatz liegt nahe, da es sich bei einer Typhierarchie um einen gerichteten Graph handelt, wobei die Kanten zwischen den Elementen Obertyp-Untertyp-Beziehungen entsprechen. Die Bestimmung von Obertyp-Untertyp-Beziehungen kann deshalb auch als die Frage nach der Existenz eines Pfades zwischen zwei Elementen des Typsystems betrachtet werden. Existiert ein solcher Pfad, so stehen die beiden Elemente in einer Typbeziehung. Existiert andererseits kein Pfad, so existiert auch keine Typbeziehung zwischen den beiden Elementen.

Die effiziente Bestimmung von Pfaden kann durch den Aufbau der Adjazenzmatrix der transitiven Hülle eines Graphs erfolgen. Die transitive Hülle enthält dabei nicht nur die Kanten zwischen direkten Ober- und Untertypen, sondern ebenso paarweise die Knoten eines Graphen, die innerhalb eines Pfades liegen. Anhand der so gewonnen Informationen kann die Adjazenzmatrix aufgebaut werden. Ein Eintrag der Adjazenzmatrix ist 1, wenn es in der transitiven Hülle ein entsprechendes Tupel mit den zwei Knoten, die den Matrix-Eintrag referenzieren, gibt.
Übertragen auf die Ontologie-Elemente entspricht der Index in diese n×n-Matrix der Typkennung für das Element. Beim Matrixzugriff mit der Typkennung als Zeilenindex wird das Ontologie-Element in seiner Eigenschaft als potentieller Untertyp referenziert. Bei einem Zugriff mit der Typkennung als Spaltenindex hingegen ist die Bedeutung des Elements in seiner Eigenschaft als Obertyp zu suchen. Abbildung 6.4 stellt die Verwendung dieser Adjazenzmatrix beispielhaft dar.

Die Frage "Ist `Aspirin` Untertyp von `Schmerzmittel`" wird nun anhand der Ontolo-

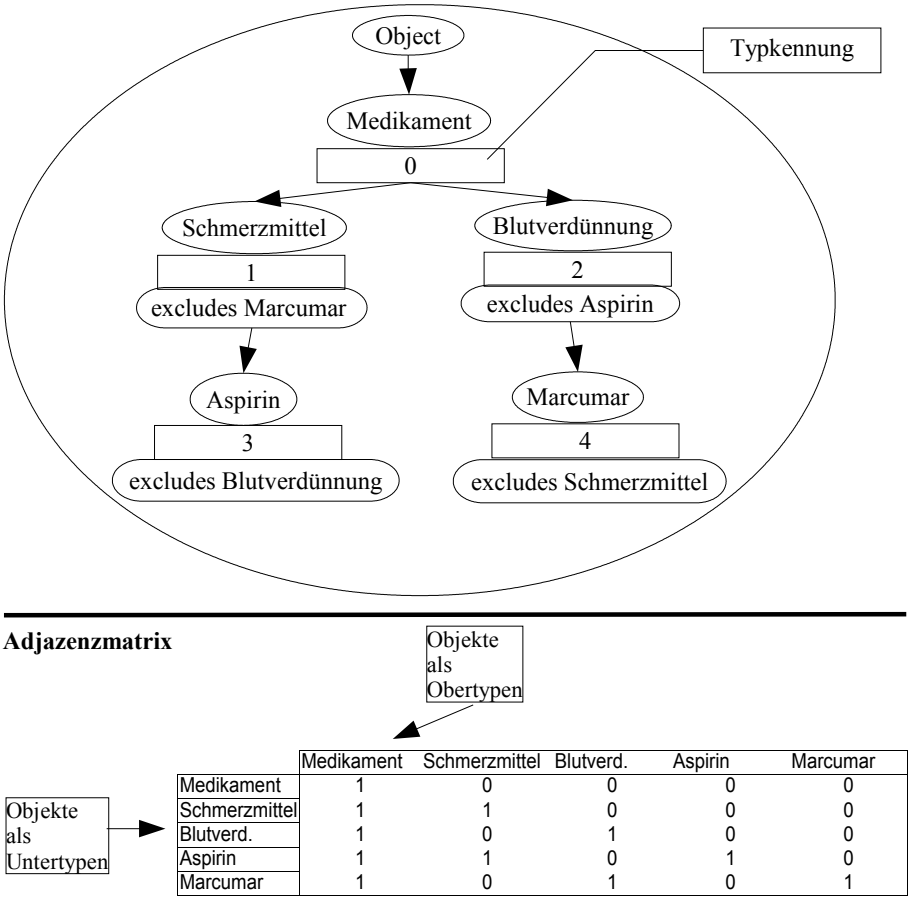


Abbildung 6.4.: Adjazenzmatrix

gie und Adjazenzmatrix aus Abbildung 6.4 untersucht.

$\Longrightarrow$ Kennung `Schmerzmittel`: 1
$\Longrightarrow$ Kennung `Aspirin`: 3

Der Auswertungsalgorithmus stellt sich exemplarisch in Pseudocode folgendermaßen dar:

```
int[][] adjazenzmatrix;
adjazenz = new int[n][n];
/* Adjazenzmatrix berechnen */

boolean isSubtype(int subTypeID, int superTypeID){
    if (adjazenzmatrix[subTypeID][superTypeID] == 1)
            return true;
```

```
    else return false;
}
```

Für die oben aufgeworfene Frage stellt sich der Funktionsaufruf als `isSubtype(3,1)` dar. Korrekterweise ist das Ergebnis der Funktion `true`, die Untertypbeziehung von `Aspirin` zu `Schmerzmittel` wird damit bestätigt.

Die Bewertung anhand der festgelegten Kriterien muss auch für das Konzept der Adjazenzmatrix vorgenommen werden. Die Untersuchung zweier Elemente auf Typzugehörigkeit kann erfolgen, indem auf die Adjazenzmatrix zugegriffen wird, ohne dass eine Durchsuchung der Ontologie notwendig ist. Es genügt also ein Speicherzugriff und ein Vergleich, um eine Aussage über eine Typzugehörigkeit treffen zu können. Das Konzept erfüllt damit das Effizienzkriterium.
Legt man das Kriterium der Speicheranforderung als Maßstab an das Konzept der Adjazenzmatrix an, so muss zur Bewertung des Kriteriums zugrundegelegt werden, dass die Größe der Matrix quadratisch mit der Anzahl der Elemente der Ontologie wächst. Der benötigte Speicherplatz kann jedoch durch eine geschickte Implementierung minimiert werden.

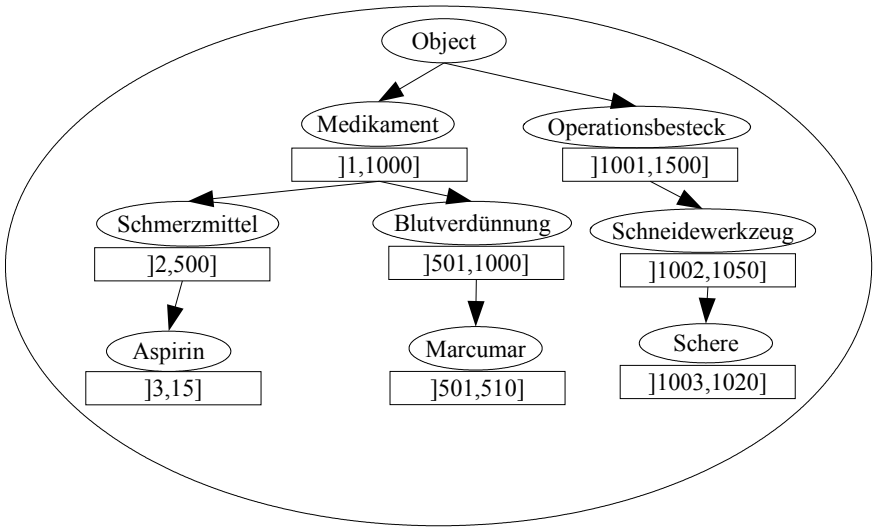
Von den bisher dargestellten Ansätzen ist dieser Ansatz der praxistauglichste. Ein großer Nachteil des Ansatzes gegenüber den bisher vorgestellten Ansätzen ist jedoch, dass die Kennung allein keine direkte Information über Typzugehörigkeiten enthält, sondern tatsächlich nur die Eigenschaft eines Index aufweist. Dadurch kann die Typkennung nicht dazu verwendet werden, um die Objekte eines Prozesses so zu organisieren, dass die Suche nach einem Objekt unter der Berücksichtigung von Untertypen anhand der Typkennung möglich ist. Dies ist, wie sich später zeigen wird, ein entscheidender Nachteil.

6.1.2.4. Partitionierung des Wertebereichs der Typkennungen

Eine Möglichkeit zur Vergabe von Typkennungen an Elemente einer Ontologie baut auf dem Prinzip auf, dass der Wertebereich für die Typkennung eines Elements und der Typkennungen aller seiner Untertypen a priori festgelegt wird. Beginnend bei den Elementen der ersten Hierarchieebene werden die Wertebereiche in Form von Intervallen vergeben, die den Wertebereich für mögliche Untertypen des Elements aufspannen. Das Intervall eines Untertyps muss deswegen innerhalb des Intervalls seines Obertyps liegen. Für die Vergabe der Intervalle müssen folgende Einschränkungen garantiert werden:

- Verschiedene Intervalle dürfen sich nicht überlappen.
- Die Intervalle für Elemente einer Hierarchieebene müssen disjunkt sein.

Eine Untertypbeziehung eines Objekts zu einem zweiten Objekt liegt dann vor, wenn die Kennung des potentiellen Untertyps innerhalb des Intervalls des potentiellen Obertyps liegt. Abbildung 6.5 stellt eine Beispielontologie dar, in der den Elementen Typkennungen in Form von Intervallen zugewiesen werden.



Veranschaulichung der Partitionierung der Wertebereiche für Untertypen

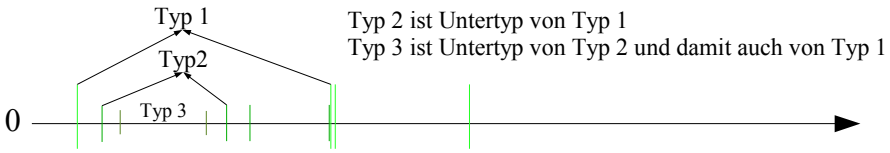


Abbildung 6.5.: Typbeziehungen über Festlegung von Wertebereichen

Wiederum soll die Auswertung von Typbeziehungen anhand eines Beispiels aufgezeigt werden.

Ist `Aspirin` Untertyp von `Schmerzmittel`?
$\Longrightarrow$ Intervall von `Schmerzmittel`: $]2, 500]$
$\Longrightarrow$ Intervall von `Aspirin`:]3,15]

Der Auswertungsalgorithmus wird für das vorliegende Konzept, wie nachfolgend in Pseudocode dargestellt, angepasst:

```
boolean isSubtype(int superTypeMin,int superTypeMax,
                  int subTypeMin, int subTypeMax){
    if ((subTypeMin > superTypeMin)&&(subTypeMax <= superTypeMax))
            return true;
    else return false;
}
```

Für die Untersuchung der Untertypbeziehung von `Aspirin` zu `Schmerzmittel` ist der Funktionsaufruf `isSubtype(2,500,3,15)` notwendig, der korrekterweise als Ergebnis `true` liefert. Die Untertypbeziehung wird also bestätigt.

Der Algorithmus benötigt maximal zwei Vergleiche für den Fall, dass der erste Vergleich `true` liefert. Durch die Vergabe der Kennungsintervalle wird das Durchsuchen der Ontologie vermieden und die entsprechende Schleife verhindert. Der Mechanismus erfüllt das Effizienzkriterium.
Auch das Speicherkriterium wird nicht verletzt, da jedes Element, egal auf welcher Hierarchieebene es sich befindet, für die Kennung zwei Integer benötigt.
Das Konzept der Partitionierung besitzt nichtsdestotrotz auch Nachteile. So begrenzt das Intervall, das einem Typ zugewiesen wird, die maximale Anzahl möglicher Untertypen. Werden mehr Untertypen in der Ontologie festgelegt, entsteht ein Intervall-Überlauf, der besonders behandelt werden muss. Desweiteren muss geklärt werden, wie die Intervalle vergeben werden. Grundsätzlich kann dies manuell geschehen oder durch einen Algorithmus automatisiert werden. Da die manuelle Vergabe in großen Ontologien sehr aufwendig ist, scheint ein automatisiertes Verfahren geeigneter. Dies könnte skizzenhaft so aussehen, dass für jedes Element rekursiv die Anzahl der Nachfolger bestimmt wird. Anhand dieser Daten kann dann die benötigte Intervallgröße festgelegt werden. Das Intervall des Obertyps dient dann als Ausgangspunkt zur Festlegung des Intervalls des untersuchten Elements.

Trotz der genannten Nachteile stellt sich die Möglichkeit der Partitionierung der Wertebereiche für Typkennungen als die am universellsten einsetzbare Alternative dar. Die geforderten Kriterien werden durch das Konzept erfüllt. Ebenso wird sich später zeigen, dass der Mechanismus gut geeignet ist, um die Objekte eines Prozesses effizient zu organisieren und trotzdem bei der Suche nach einem Typ auch Untertypen erkennen zu können. Demzufolge wird von diesem Konzept der Partitionierung der Typkennungen in der weiteren Ausarbeitung ausgegangen.

6.2. Effiziente Organisation von Prozessdaten

Wie bereits angesprochen, ist es insbesondere im Zusammenhang mit Änderungsoperationen wichtig, dass die semantischen Informationen eines Prozesses effizient ausgewertet werden können. Dabei gilt es zwischen zwei verschiedenen Arten relevanter Informationen zu unterscheiden.

1. Die Objekte der im Prozess vorhandenen Aktivitäten müssen für eine effiziente Durchsuchung organisiert werden.
2. Die Regeln der Objekte, die die Aktivitäten des Prozesses semantisch beschreiben, müssen effizient nach ihren jeweiligen Zielobjekten durchsucht werden können.

Der 1. Fall ist relevant, wenn die semantischen Beziehungen von Objekten einer Aktivität ausgewertet werden. Diese semantischen Beziehungen besitzen Zielobjekte, die durch Objekte des Prozesses erfüllt werden können. Deshalb ist es offensichtlich, dass die Durchsuchung der im Prozess vorhandenen Objekte einen Flaschenhals für die Leistungsfähigkeit der Korrektheitstests darstellt und das Potential für Optimierungsmöglichkeiten ausgelotet werden muss.
Die zweite relevante Art von Prozessinformationen werden von den semantischen Regeln gebildet, die den im Prozess vorhandenen Objekten zugeordnet sind. Die Objekte einer Zielaktivität müssen daraufhin untersucht werden, ob sie Zielobjekte von semantischen Beziehungen des Prozesses sind. Deshalb verursacht die Durchsuchung der Semantikregeln des Prozesses potentiell ebenfalls einen Flaschenhals für die Korrektheitstests, so dass auch hier nach Optimierungen gesucht werden muss.

6.2.1. Organisation der Objekte

Die Suche nach Optimierungsmöglichkeiten soll mit der Frage begonnen werden, wie die Objekte, die die Aktivitäten eines Prozesses semantisch beschreiben, organisiert werden können, um eine möglichst effiziente Suche nach dem Vorkommen eines bestimmten Objektes garantieren zu können. Zusätzlich kommt die erschwerende Bedingung hinzu, dass als Ergebnis einer Suche auch Untertypen eines vorgegebenen Objekts als Suchtreffer erkannt werden müssen.

Es bleibt die Frage, ob die Organisationsform ebenso Obertypen als Treffer unterstützen muss. Um diese Frage zu beantworten, müssen die Situationen identifiziert werden, die eine Durchsuchung der Objekte des Prozesses anstoßen. Ausgangspunkt stellt der Korrektheitstest einer Änderungsoperation dar. Es muss geprüft werden, ob die Semantikregeln der Objekte der Zielaktivität durch Objekte des Prozesses erfüllt werden. Die Einordnung der Objekte innerhalb einer Typhierarchie bringt mit sich, dass eine semantische Beziehung nicht nur durch Vorkommen des definierten Zielobjekts im Prozess erfüllt werden kann, sondern ebenso durch Untertypen des Zielobjekts. Andersherum ist es jedoch nicht möglich, eine semantische Beziehung durch die Ausprägung eines Obertyps des definierten Zielobjekts zu erfüllen.
Damit ist die aufgeworfene Frage beantwortet und es ist nicht notwendig, die Objekte so zu organisieren, dass auch Obertypen gesuchter Typen zu Suchtreffern führen.

Als Ausgangspunkt für die Untersuchung effizienter Organisationsformen muss zunächst bestimmt werden, welche Information zur Organisation der Objekte sinnvoll ist. Es wird eine Eigenschaft benötigt, die ein Objekt eindeutig identifiziert. Deshalb kommen zwei Informationen, die mit einem Objekt assoziiert sind, als potentielle Kandidaten in Frage:

- der Typname
- die Typkennung

Gegen die Verwendung des Typnamens bzw. Objektnamens spricht, dass der Objektname keine direkte Information über Typbeziehungen enthält, so dass nicht auf einfache Weise bei der Suche nach einem Objekt auch gleichzeitig Untertypen erkannt werden. Ein weiterer Nachteil bei der Verwendung des Objektnamens ist die schwankende Speicherplatzanforderung, die für ein Element einer entsprechenden Datenstruktur benötigt wird.
Es bleibt also die Typkennung als Ausgangspunkt für die Organisation der Objekte, wobei auf das Intervallverfahren zurückgegriffen wird. Die Typkennung erfüllt das Kriterium der eindeutigen Identifizierung eines Objekttyps und hat insbesondere den Vorteil, dass bei der Suche nach einer Typkennung auf einfache Weise erkannt werden kann, wenn ein untersuchtes Element ein Untertyp des gesuchten Typs ist. Dazu müssen lediglich die Intervallgrenzen des gesuchten Typs mit dem aktuell untersuchten Element verglichen werden.

Wie können nun die Objekte mittels ihrer Typkennung so organisiert werden, dass möglichst effizient nach bestimmten Typen gesucht werden kann?

Zur Beantwortung obiger Frage lohnt ein Blick auf verschiedene Strategien zur Suche in Datensätzen. Bei der Wahl einer entsprechenden Strategie ist für den Kontext PMS weniger relevant, wie sich die Suchkomplexität im besten Fall verhält, sondern es ist vielmehr wichtig, den schlimmsten Fall zu berücksichtigen. Dieser Zusammenhang wird deutlich, wenn als Organisationsform eine einfach verkettete Liste gewählt wird. Im besten Fall ergibt bei einer Suche bereits das erste Element der Liste einen Treffer, im schlimmsten Fall muss jedoch die komplette Liste durchsucht werden. Für PMS ist es wichtig, dass die Zeit der Sperrung von Instanzen eines Prozesses nicht beliebig groß ist. Deshalb ist es unerlässlich, zuverlässige Aussagen darüber treffen zu können, wie sich die Durchsuchung der Objekte des Prozesses auf das Laufzeitverhalten der Korrektheitstests auswirkt. Das Ziel muss sein, die Laufzeit für den schlechtesten Fall durch geeignete Organisation der Objekte des Prozesses zu minimieren. Folgende Alternativen werden einer genaueren Untersuchung unterzogen:

- Hashing-Verfahren
- binäre Bäume

Hashing-Verfahren haben den Vorteil, bei Wahl einer geeigneten Hash-Funktion durchschnittlich eine sehr geringe Anzahl von Speicherzugriffen zu benötigen, um eine Aussage über einen Suchtreffer machen zu können. Das Verfahren hat jedoch entscheidende Nachteile, so dass die Anwendung zur Organisation der Objekte nicht empfehlenswert erscheint. Generell eignen sich Hashingverfahren dann gut, wenn eine recht genaue Vorhersage für die Anzahl der erwarteten Datensätze und die Größe der zu hashenden Werte möglich ist. Ändern sich diese Bedingungen, so kann es schnell zu Überläufen in der Hashtabelle kommen, denen mit geeigneten Verfahren begegnet werden muss. Dies kann sich negativ auf die Laufzeit auswirken. Geht man davon aus, dass Prozesse sehr dynamische Strukturen sind, die sich oft ändern können, so sind Hashing-Verfahren nicht als optimal geeignet zu bewerten. Ein weiterer Nachteil von Hashing ist, dass das zugrundeliegende

Prinzip lediglich Punktanfragen zulässt. So eignet sich das Verfahren naturgemäß nicht, um bei der Suche nach einem Typ auch Objekte von Untertypen des gesuchten Objekts als Ergebnis zu erhalten.

Als zweites muss die Alternative der binären Suchbäume untersucht werden. Von einem gegebenen Knoten N wird in einem solchen Suchbaum nach links verzweigt, wenn entweder die Typkennung des Kindknotens innerhalb des Intervalls von N liegt, oder, bei disjunkten Intervallen, die Intervallgrenzen des Kindknotens kleiner sind als die des Intervalls von N. In allen anderen Fällen wird nach rechts verzweigt.
Das Problem bei einfachen binären Suchbäumen besteht darin, dass ihr Aufbau je nach Reihenfolge des Einfügens der einzelnen Elemente abweichen kann. Schlimmstenfalls entartet ein Baum mit n Elementen zu einem Baum der Höhe n, also zu einer linearen Liste. In diesem Fall würden bei der Suche schlimmstenfalls n Elemente des Baumes zugegriffen, so dass bei Einsatz dieses Verfahrens keine garantierte Verbesserung gegenüber einer sequentiellen Durchsuchung der Objekte gegeben ist. Dies kann an einem Beispiel verdeutlicht werden. Folgende Elemente sollen in der angegebenen Reihenfolge in einen binären Suchbaum eingefügt werden:
$]1, 100],]2, 50],]3, 40],]4, 30]$
Der binäre Baum, der sich dadurch ergibt ist in Abbildung 6.6 dargestellt.
Einfache binäre Suchbäume genügen also nicht den Kriterien, die an eine effiziente Or-

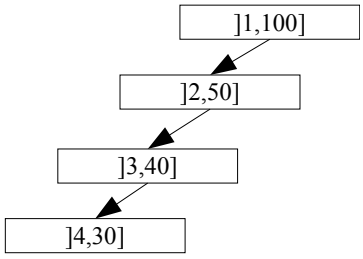


Abbildung 6.6.: Entarteter binärer Baum

ganisationsform für die Objekte eines Prozesses gestellt werden, da sich die Komplexität der Suche je nach Aufbau des Baumes ändert. Bestenfalls hat ein Baum die Höhe log_2(n), so dass maximal log_2(n) Schritte benötigt werden, um eine Aussage über das Vorhandensein eines gesuchten Elementes im Baum treffen zu können. Deshalb müssen spezielle Arten von binären Bäumen untersucht werden, die eine Höhe von log_2(n) garantieren. Diese Art von binären Bäumen findet sich in den balancierten binären Bäumen. Der einfachste Vertreter der balancierten binären Bäume ist der AVL-Baum, der durch die Bedingung definiert wird, dass sich für jeden Knoten die beiden Teilbäume in der Höhe maximal um 1 unterscheiden. Formal ausgedrückt lautet nach [Sch97] die Bedingung für jeden Knoten N des Baumes also $|height(N.left) - height(N.right)| \leq 1$.
Da das Kriterium der Ausbalanciertheit dieser Bäume nicht verletzt werden darf, müssen beim Einfügen und Löschen von Knoten des Baumes besondere Maßnahmen ergriffen werden, wenn ansonsten die Bedingung nicht mehr erfüllt wäre. Nach [Sch97] und

[RP02] ist die Komplexität für das Einfügen wie für das Löschen eines Eintrages bei AVL-Bäumen O(log_2(n)).
Eine Antwort auf die Frage, wie gut sich binäre Suchbäume für Anfragen eignen, die auch Untertypen des gesuchten Objekts als Treffer verlangen, liefert eine Analyse des Suchablaufs innerhalb eines Suchbaumes. Dabei wird das Intervall eines jeden Knotens, der bei der Suche passiert wird, mit dem Intervall des gesuchten Objekts verglichen. Ist das Ergebnis dieses Intervallvergleichs, dass das Intervall des Baumknotens innerhalb des gesuchten Intervalls liegt, so stellt der aktuell untersuchte Baumknoten einen Untertyp des gesuchten Objekts dar.

Binäre Suchbäume werden in der weiteren Ausarbeitung zur Organisation der Objekte eines Prozesses benutzt. Zur Verdeutlichung sollen skizzenhaft anhand einiger Beispiele die Abläufe innerhalb eines Suchbaumes dargestellt werden. Die Grundlage der Beispiele bildet ein durch eine Einfüge-Operation angestoßener Korrektheitstest. Konkret ist die Frage zu beantworten, ob eine Regel der einzufügenden Aktivität durch den Prozess erfüllt wird. Der Ablauf stellt sich wie folgt dar:

- Kennung des Zielobjekts der Regel bestimmen. Diese ist entweder direkt in der Regel codiert oder es wird eine zusätzliche Abbildung Objektname⟶Typkennung benötigt.
- Suche im Objekt-Suchbaum des Prozesses.

Abbildung 6.7 stellt den Suchbaum dar, der als Grundlage für die nachfolgenden Beispiele dienen soll.

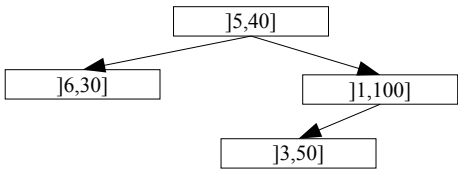


Abbildung 6.7.: Beispiel eines Objekt-Suchbaums

1. Zu prüfende Regel des Objekts o der einzufügenden Aktivität: o excludes]6,30]
   Durchgeführte Suchschritte:
   - 6>5 und 30<40 ⇒ Weiter im linken Teilbaum der Wurzel.
   - 6==6 ⇒ Zielobjekt im Prozess vorhanden. ⇒ Ausschlussbedingung potentiell verletzt.
   - Kein weiteres Element auf dem Suchpfad.
2. Zu prüfende Regel des Objekts o der einzufügenden Aktivität: o excludes]7,25]
   Durchgeführte Suchschritte:
   - 7>5 und 25<40 ⇒ Weiter im linken Teilbaum der Wurzel.

- 7>6 und 25<30 ⇒ Weiter im linken Teilbaum.
- Kein weiteres Element im Suchpfad. ⇒ Weder das Objekt noch ein Objekt eines Untertyps im Prozess vorhanden. ⇒ Ausschlussbeziehung wird nicht verletzt.

3. Zu prüfende Regel des Objekts o der einzufügenden Aktivität: o excludes]2,60] Durchgeführte Suchschritte:
   - 2<5 und 60>40 ⇒ $]5, 40]$ ist Untertyp von $]2, 60]$. ⇒ Objekt eines Untertyps des Zielobjekts im Prozess vorhanden. Die Ausschlussbeziehung ist potentiell verletzt.
     Die Auswertung wird im rechten Teilbaum fortgesetzt.
   - 2>1 und 60<100 ⇒ Weiter im linken Teilbaum.
   - 2<3 und 60>50 ⇒ Weiterer Untertyp gefunden, der den Ausschluss potentiell verletzt.

6.2.2. Organisation der semantischen Regeln

Ausbalancierte binäre Suchbäume wurden für die effiziente Durchsuchung der Objekte eines Prozesses eingeführt. Die zweite Art relevanter Daten für die Semantikprüfung sind die zu den Objekten des Prozesses gehörenden semantischen Beziehungen in Form der Semantikregeln. Bei den verschiedenen Änderungsoperationen müssen die Objekte der Zielaktivität in den Zielobjekten der Regeln des Prozesses gesucht werden, um eventuelle semantische Konflikte erkennen zu können. Die Zielobjekte der Regeln sind also die relevante Information der Regeln, die effizient durchsucht werden müssen. Deshalb können die für die effiziente Durchsuchung der Objekte des Prozesses getroffenen Aussagen direkt auf die Organisation der Zielobjekte der Regeln übertragen werden.

Im Unterschied zur Suche nach Objekten des Prozesses müssen in diesem Fall nicht Untertypen des gesuchten Typs als Treffer erkannt werden. Dies wird deutlich, wenn die Bedeutung einer Regel näher untersucht wird. Ein Objekt erfüllt alle Regeln, deren Zielobjekte seinem Typ entsprechen. Zusätzlich werden alle Regeln erfüllt, deren Zielobjekte Obertypen des Objekts darstellen.
Der Aufbau eines ausbalancierten binären Suchbaums für die Zielobjekte muss gegenüber einem entsprechenden Suchbaum für die Objekte eines Prozesses nicht abgeändert werden. Genauso wie durch einen Vergleich der Intervalle der Knoten des Baumes mit dem Intervall eines gesuchten Typs Untertypen erkannt werden können, können durch Anpassung der Vergleichsstrategie Obertypen erkannt werden. Die Anpassung besteht lediglich darin, dass nach umschließenden Intervallen im Baum in Bezug auf das Intervall des angefragten Typs gesucht wird.

Neben dem Objekt-Suchbaum wird ein zweiter Suchbaum für die Zielobjekte der Regeln etabliert, der im weiteren als Zielobjekt-Suchbaum bezeichnet wird. Analog zur Veranschaulichung des Vorgehens für den Objekt-Suchbaum dienen Beispiele zur Verdeutlichung der Abläufe bei der Suche im Zielobjekt-Suchbaum. Diesmal lautet die zu beantwortende Anfrage, ob ein Objekt einer einzufügenden Aktivität bzw. ein Objekt eines Obertyps als Zielobjekt einer Regel vorkommt, die einem Objekt des Prozesses zugeordnet ist. Der skizzenhafte Ablauf stellt sich dazu wie folgt dar:

- Kennung des Objektes der einzufügenden Aktivität bestimmen. Diese ist entweder direkt in der semantischen Aktivitätenbeschreibung codiert oder es wird eine zusätzliche Abbildung Objektname⟶Typkennung benötigt.
- Suche im Zielobjekt-Suchbaum des Prozesses.

Abbildung 6.8 stellt den Suchbaum dar, der als Grundlage für die nachfolgenden Beispiele dienen soll.

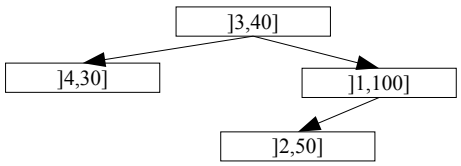


Abbildung 6.8.: Beispiel eines Zielobjekt-Suchbaums

1. zu prüfendes Objekt o der einzufügenden Aktivität:]4,30]
   durchgeführte Suchschritte:
   - 4>3 und 30<40 ⇒ $]3, 40]$ ist Obertyp des Objekts $]4, 30]$. Das Objekt o erfüllt damit potentiell die dem Suchbaumelement zugeordnete Regel. Der Baum muss jedoch noch weiter durchsucht werden, da das untersuchte Objekt Zielobjekt mehrerer Regeln sein kann. Für die weitere Suche wird nach links verzweigt
   - 4==4 ⇒ Objekt o als Zielobjekt einer Regel des Prozesses vorhanden. Das Objekt o erfüllt damit potentiell die dem Suchbaumelement zugeordnete Regel.
   - kein weiteres Element im Baum und damit keine weitere Regel im Prozess vorhanden, die potentiell durch das untersuchte Objekt erfüllt wird.
2. zu prüfendes Objekt o der einzufügenden Aktivität:]101,200]
   durchgeführte Suchschritte:
   - 101>3 und 200>40 ⇒ weiter im rechten Teilbaum der Wurzel
   - 101>1 und 200>100 ⇒ weiter im rechten Teilbaum

- kein weiteres Element auf dem Suchpfad vorhanden ⇒ Weder das Objekt o noch ein Obertyp wurden als Zielobjekt der Regeln des Prozesses gefunden. ⇒ Objekt hat keinen Einfluss auf die semantischen Beziehungen des Prozesses.

6.3. Laufzeitanalyse unter Berücksichtigung der Optimierungen

Nachdem bereits zwei der eingangs des Kapitels aufgeworfenen performance-kritischen Punkte untersucht wurden, ist es Zeit, die Optimierungsansätze in die Laufzeitanalyse mit einzubeziehen, um zu bestimmen, ob sich die Laufzeiteigenschaften positiv verändert haben. Dazu dienen abermals die in Abbildung 6.1 eingeführten Voraussetzungen als Ausgangspunkt. Die Laufzeitanalyse für das naive, unoptimierte Verfahren hatte zur Erinnerung als Ergebnis 2*a*n*m*r*t notwendige Schleifendurchläufe im schlechtesten Fall.

Gegeben sei die Domäne Dom = ($\mathcal{N}$,$\mathcal{O}$, $\mathcal{R}$, SEM_ACT, SEM_RELS). Die Aktivität Act_{n+1} soll in einen Prozess mit dem Schema S = (N,E,D,Dom) eingefügt werden.
TARGETOBJECT: $\mathcal{R} \rightarrow \mathcal{O}$ liefere das Zielobjekt einer Semantikregel.
Der Ablauf beim Einfügen der Aktivität Act_{n+1} besteht grundlegend aus folgenden Schritten.

- Schritt 0: Instanz(en) des Prozesses sperren.
- Schritt 1: Verträglichkeitstest (Semantik)
  `foreach object in` Act_{n+1} `do{`
  (i) Test: Ist `object` das Ziel einer Regel des Prozesses?
  (ii)
  (a) Bestimme die Regelmenge für `object`.
  (b) Prüfe jede der Regeln auf Konflikte mit Objekten, die in den Aktivitäten des Prozesses vorhanden sind.

  }
- Schritt 2: Einfügen der Aktivität nach erfolgreichem Verträglichkeitstest.
- Schritt 3: Einfügen der Regeln und Objekte der neuen Aktivität.
- Schritt 4: Instanz(en) wieder freigeben.

Die bereits in 6.1.1 skizzenhaft eingeführten Algorithmen stellen sich unter Berücksichtigung der Optimierungsansätze folgendermaßen dar. Die zwei verwendeten Funktionen `searchObjectTree` und `searchTargetObjectTree` greifen auf den Objekt- bzw. Zielobjektsuchbaum zu und suchen nach Vorkommen des als Parameter übergebenen Objekts.

```
Schritt 1.(i):
foreach object in SEM_ACT(Act_{n+1}) do {
   searchTargetObjectTree(object);
}

Schritt 1.(ii):
foreach object in SEM_ACT(Act_{n+1}) do {
   foreach rule in SEM_RELS(object) do{
        searchObjectTree(TARGETOBJECT(rule));
   }
}
```

Wie an dem abgeänderten Algorithmus zu erkennen ist, fallen die meisten Schleifen durch das optimierte Erkennungsverfahren für Typbeziehungen und die effiziente Suche nach Objekten und Zielobjekten von Regeln des Prozesses gegenüber dem naiven Verfahren weg. Das Laufzeitverhalten unter Berücksichtigung der Optimierungen stellt sich nun folgendermaßen dar:
a sei die Anzahl der Objekte in der einzufügenden Aktivität, r die maximale Anzahl an Regeln pro Objekt. l_{max} sei die maximale Anzahl an Elementen der zwei Suchbäume. Damit kann die Anzahl der notwendigen Schleifendurchläufe als a*r*$log_2(l_{max})$+a*$log_2(l_{max})$= a*(r+1)*$log_2(l_{max})$ abgeschätzt werden.

Ein Zahlenbeispiel kann verdeutlichen, wie die Optimierungen tatsächlich wirken. Dazu werden folgende Annahmen zugrunde gelegt:
Anzahl der Aktivitäten im Prozess (=n): 1000
maximale Anzahl von Objekten pro Aktivität (=m): 20
maximale Anzahl von Regeln pro Objekt (=r): 20
maximale Tiefe der Typhierarchie (=t): 1000
Anzahl der Objekte der einzufügenden Aktivität (=a): 20
Maximum der Anzahl an Elementen in den Suchbäumen (=l_{max}): 1000*20*20 = 400.000

- Laufzeitverhalten beim naiven Verfahren im schlimmsten Fall:
  2*a*n*m*r*t=2*20*1000*20*20*1000=16.000.000.000 Schleifendurchläufe im schlechtesten Fall
- Abgeschätztes Laufzeitverhalten beim optimierten Verfahren:
  a*(r+1)*$log_2(l_{max})$ = 20*21*log_2(400.000)≈7816 Schleifendurchläufe im schlechtesten Fall

Die Optimierungen bringen eine dramatische Verbesserung des Laufzeitverhaltens für den "worst-case". In Zahlen ausgedrückt beläuft sich die Verbesserung auf einen Faktor von ca. 2.047.082, d.h. die Anzahl der Schleifendurchläufe beim unoptimierten Verfahren beträgt im schlimmsten Fall das 2.047.082-fache des optimierten Verfahrens.
Die Zahlen machen deutlich, dass der "Trade-Off" zwischen praktikablem Laufzeitverhalten und Ausdrucksmächtigkeit durch eine intelligente Organisation der Daten weitestgehend aufgelöst werden kann. Somit ist es möglich, die durch die objektzentrierte Sicht

auf die semantischen Beziehungen sowie durch die Etablierung eines Typsystems erhaltenen Vorteile beizubehalten, ohne dass das Laufzeitverhalten einen sinnvollen Einsatz der Semantikprüfung gefährdet.

6.4. Sperrung von Instanzen - Optimierung

Eine weitere Optimierungsmöglichkeit, die ausgenutzt werden kann, ist noch zu nennen. So ist es nicht notwendig, dass das Einfügen der Objekte und der Zielobjekte der Regeln einer einzufügenden Aktivität bzw. das Löschen der Objekte einer zu löschenden Aktivität in den bzw. aus dem entsprechenden binären Baum, durchgeführt werden muss, solange die Instanzen des Prozesses gesperrt sind. Die Sperre kann freigegeben werden, sobald der eigentliche Korrektheitstest durchgeführt wurde. Die Instanzen können dann bereits weiter ausgeführt werden, während im Hintergrund noch die Verwaltungsaufgaben durchgeführt werden. Eine zweite Sperre muss dann sicherstellen, dass eine weitere Änderungsoperation erst durchgeführt werden kann, wenn alle Datenstrukturen des Prozesses aktualisiert sind. Bei der beschriebenen Möglichkeit handelt es sich um ein zweistufiges Sperrverfahren. Der Ablauf dieses Sperrverfahrens ist in Abbildung 6.9 dargestellt.

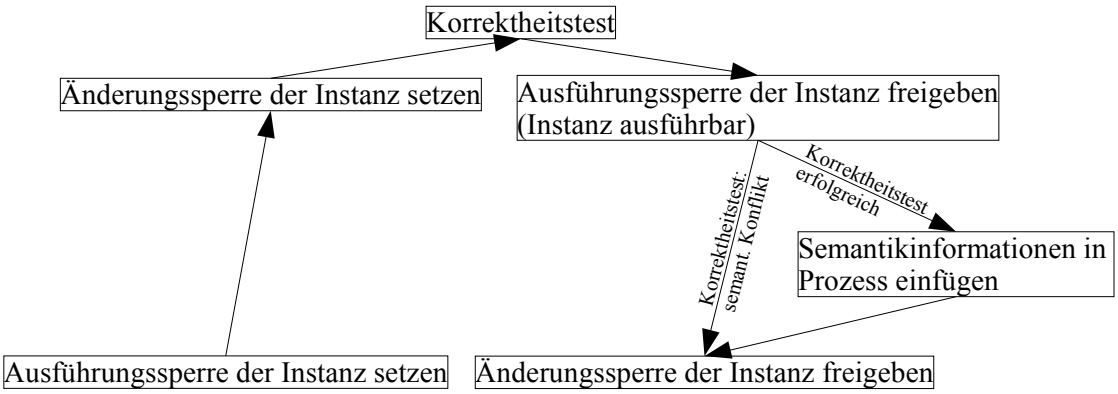


Abbildung 6.9.: Zweistufiges Sperrverfahren

6.5. Bestimmung der Regelmenge

Die bisher angestellten Optimierungen haben die effiziente Bestimmung von Typbeziehungen sowie die effiziente Organisation der Objekte und Zielobjekte der Regeln des Prozesses als Ziel. Außer acht gelassen wurde jedoch bisher ein weiterer wichtiger Aspekt. Bei einer Änderungsoperation müssen für jedes Objekt der Ziel-Aktivität die dem Objekt zugeordneten Regeln bestimmt werden. Die Regeln, die die semantischen Beziehungen

definieren, sind bei den Elementen der Ontologien gespeichert. Zwei mögliche Ausgangspunkte zur Bestimmung der Regelmenge müssen diskutiert werden:

- Es existiert keine Regelvererbung von Obertypen zu Untertypen.
- Regeln werden von Obertypen auf Untertypen vererbt, wobei Regeln aus oberen Hierarchieebenen in unteren Ebenen redefiniert werden können.

Bei beiden möglichen Verfahrensweisen muss zusätzlich berücksichtigt werden, dass nur die Regeln in der tatsächlich auszuwertenden Regelmenge vorhanden sind, die mit dem für das auszuwertende Ontologie-Element innerhalb der semantischen Aktivitätenbeschreibung festgelegten Kontext übereinstimmen. Bereits in Kapitel 4.2.1 wurden die Vorteile diskutiert, die die Regelvererbung mit sich bringt, so dass es wünschenswert ist, Regelvererbung in einer Umsetzung zu unterstützen. Die Vorteile durch Regelvererbung werden jedoch durch Laufzeitnachteile bei der Bestimmung der Regelmenge für ein Ontologie-Element erkauft, da es nicht genügt, wie beim Ansatz ohne Regelvererbung, lediglich auf das entsprechende Element der Ontologie zuzugreifen, sondern ein Zugriff auf alle Elemente des Vererbungspfades notwendig ist, um sämtliche Regeln zu erhalten. Wird zusätzlich auch die Möglichkeit von Mehrfachklassifizierung angeboten, wird dieses Problem noch weiter verschärft.

Die Berücksichtigung von Regelvererbung bei der Bestimmung der Regelmenge eines Ontologie-Elementes stellt also einen weiteren performance-kritischen Flaschenhals dar. Es muss untersucht werden, ob und wie stark dieser Flaschenhals durch Optimierungen verringert werden kann. Zwei Teilaspekte bieten sich für Optimierungsstrategien an und werfen folgende zu analysierende Fragen auf:

- Wie können die Ontologie-Elemente im Speicher gehalten werden, um effizient darauf zugreifen und damit die Regelmenge eines Objektes effizient bestimmen zu können?
- Können die Regeln so organisiert werden, dass der zusätzliche Aufwand, der durch Regelvererbung entsteht, minimiert werden kann?

6.5.1. Speicher-Organisation der Ontologien

Die Elemente einer Ontologie können so organisiert werden, dass jedes Element eine Liste an Zeigern auf die direkten Untertypen besitzt. Angefangen bei der Wurzel kann der Graph mittels der Zeiger durchlaufen werden, um auf ein bestimmtes Element zuzugreifen. Dabei ergibt sich das Problem, dass keine Entscheidung getroffen werden kann, welchem Pfad im Graph gefolgt werden soll, wenn lediglich der Name eines gesuchten Ontologie-Elements bekannt ist. Es bliebe nur die Möglichkeit, sukzessive alle möglichen Pfade zu durchsuchen, bis das gesuchte Objekt gefunden wurde. Es ist offensichtlich, dass dies unter Effizienzgesichtspunkten keine befriedigende Lösung darstellt und gerade bei großen Ontologien zu Performanceeinbrüchen bei der Semantikprüfung führt.

Es stellt sich die Frage, ob die Typkennung nicht bereits implizit die nötige Information beinhaltet, die eindeutig den korrekten Pfad von der Wurzel zu einem Element kennzeichnet. Die Typkennung hat die Eigenschaft, dass diese innerhalb aller Typkennungs-Intervalle der Obertypen liegen muss, von der Wurzel bis zum direkten Obertyp des gesuchten Elements. Über einen Vergleich der Intervalle aller Elemente der ersten Hierarchieebene mit der Typkennung des gesuchten Elements kann somit eindeutig identifiziert werden, welches Element ein Obertyp dieses gesuchten Elements darstellt. Dieses Element bildet den Ausgangspunkt für die Untersuchung der Elemente der nächsten Hierarchieebene, die die direkten Nachfolger dieses Elements darstellen. Statt die Namen von Ontologie-Elementen miteinander zu vergleichen, werden also in diesem Ansatz die Typkennungen verglichen. Die Anzahl der zuzugreifenden Ontologie-Elemente bis zum Zugriff auf das gesuchte Element wird deutlich verringert.

Trotzdem ist für große Ontologien noch eine hohe Anzahl an Elementzugriffen notwendig, da nicht nur die Elemente zugegriffen werden, die auf dem Pfad liegen, sondern für jede Hierarchieebene des Pfades erst das korrekte Pfadelement bestimmt werden muss. Deshalb wäre es wünschenswert, die Elemente einer Ontologie im Speicher so zu organisieren, dass ein direkter Zugriff auf ein gesuchtes Element möglich ist. So könnte direkt auf die Informationen des Elements zugegriffen werden. Wird von Regelvererbung ausgegangen, so muss in diesem Fall der Pfad ausgehend vom zugegriffenen Element rückwärts bis zur Wurzel durchlaufen werden, um die vollständige Regelmenge zu erhalten. Diese "Rückwärtsauswertung" ist jedoch effizienter als das vorwärts gerichtete Durchlaufen der Ontologie, wenn von einer Verkettung der Ontologie-Elemente im Speicher ausgegangen wird, da der Pfad in diesem Fall ausgehend vom zugegriffenen Element bis zur Wurzel eindeutig ist und keine unnötigen Elementzugriffe notwendig sind.

Ein Element einer Ontologie kann nur direkt zugegriffen werden, wenn die Speicherstelle des Elementes bekannt ist. Die direkte Angabe einer Speicheradresse eignet sich jedoch für den Gebrauch nicht, da diese nicht stabil ist. Das bedeutet, dass bei jedem neuen Einlesen der Ontologie in den Speicher einem Element mit hoher Wahrscheinlichkeit eine andere Adresse zugewiesen wird.
Zur Entwicklung eines alternativen Konzepts hilft es, eine Ontologie aus einer anderen Perspektive zu betrachten. So besteht eine Ontologie aus einer horizontalen und einer vertikalen Komponente. In der horizontalen Komponente werden alle Elemente gespeichert, die in der selben Hierarchieebene liegen. Die vertikale Komponente hingegen hält die Adressen der einzelnen Hierarchieebenen vor. Dieser Ansatz kann direkt in eine zweistufige Array-Organisation umgesetzt werden. Für jede Hierarchie-Ebene wird dabei ein Array angelegt, wobei dessen Einträge den einzelnen Elementen der jeweiligen Hierarchieebene entsprechen. Für die vertikale Komponente wird ebenfalls ein Array angelegt, dessen Einträge auf die verschiedenen Hierarchieebenen in der Form von Zeigern auf die entsprechenden Arrays referenzieren. Um in dieser Organisation direkt auf ein bestimmtes Ontologie-Element zuzugreifen ist es lediglich notwendig, die zwei Indizes für die vertikale und horizontale Komponente zu kennen. Diese zwei Indizes zusammen werden im Folgenden als Ontologie-Adresse bezeichnet. Beim Zugriff auf ein Element wird über das vertikale Array auf das korrekte horizontale Array verwiesen, wo mittels des horizon-

talen Index direkt auf das gesuchte Element zugegriffen werden kann. Abbildung 6.10 zeigt die beschriebenen Überlegungen anhand einer Ontologie und der dazugehörigen Speicherorganisation.
Diese Organisationsform einer Ontologie im Speicher hat jedoch einen großen Nachteil.

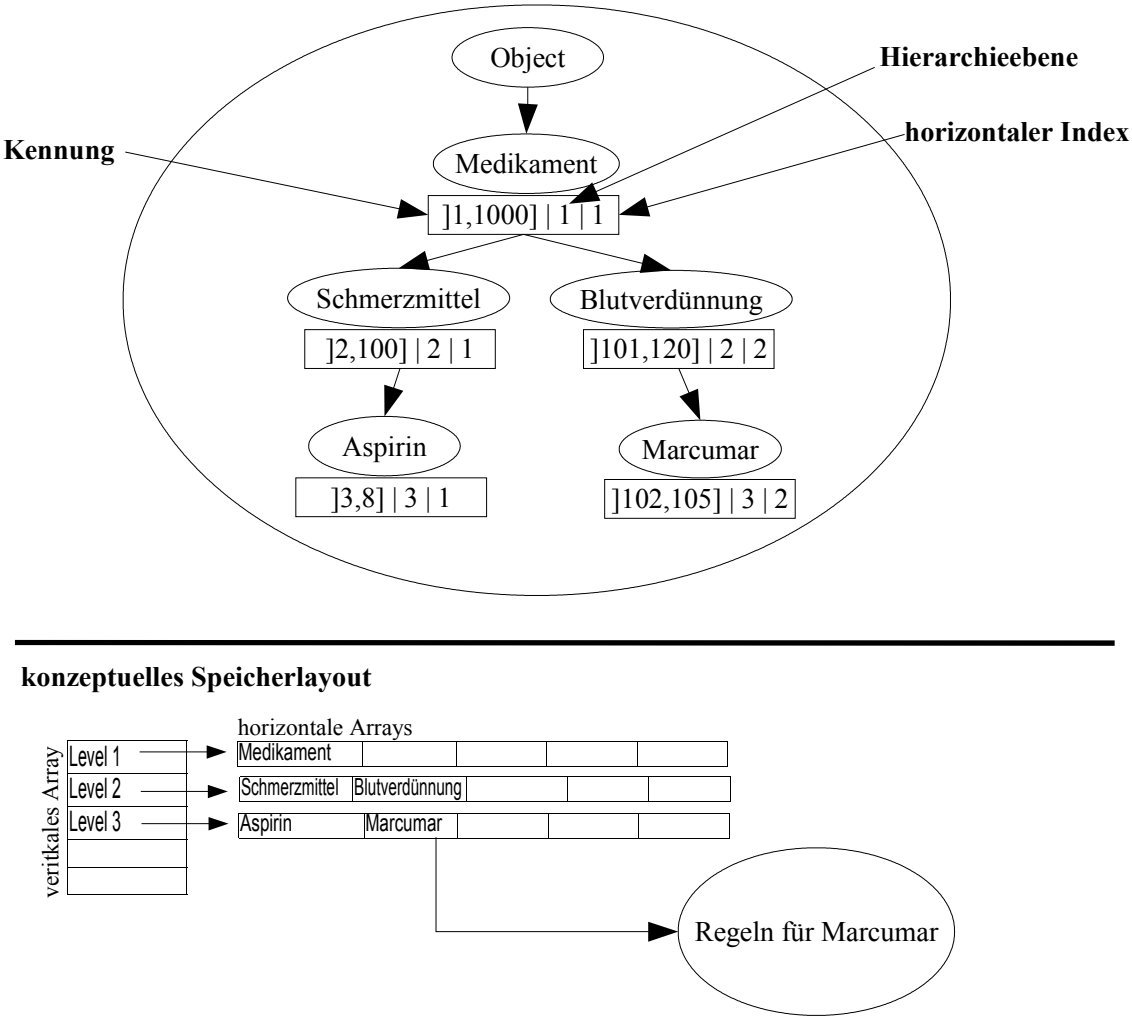


Abbildung 6.10.: Speicherorganisation einer Ontologie

Wie in Abbildung 6.10 leicht zu sehen ist, geht der Typzusammenhang zwischen einem Element und seinem Obertyp in der physikalischen Organisation verloren, wenn auf die direkte Verkettung der Elemente verzichtet wird. Der notwendige Typzusammenhang besteht durch den direkten Zugriff auf ein Element und die ”Rückwärtsauswertung” der Regelmenge in einer Zugriffsmöglichkeit auf den Obertyp. Es ist natürlich möglich, die direkte Zeiger-Verkettung von Unter- zu Obertypen zu benutzen. Eine andere Möglichkeit diesen Typzusammenhang herzustellen, ohne sich ändernde Speicheradressen in Kauf zu

nehmen, findet sich in der zusätzlichen Speicherung der Ontologie-Adresse des Obertyps im jeweiligen Ontologie-Element. Aufgrund der Stabilität der Ontologie-Adressen ist diese Alternative zu bevorzugen. Dieser Ansatz stellt sich wie in Abbildung 6.11 visualisiert dar.
Ein weiterer Vorteil, den die Stabilität der Ontologie-Adresse im Gegensatz zur Spei-

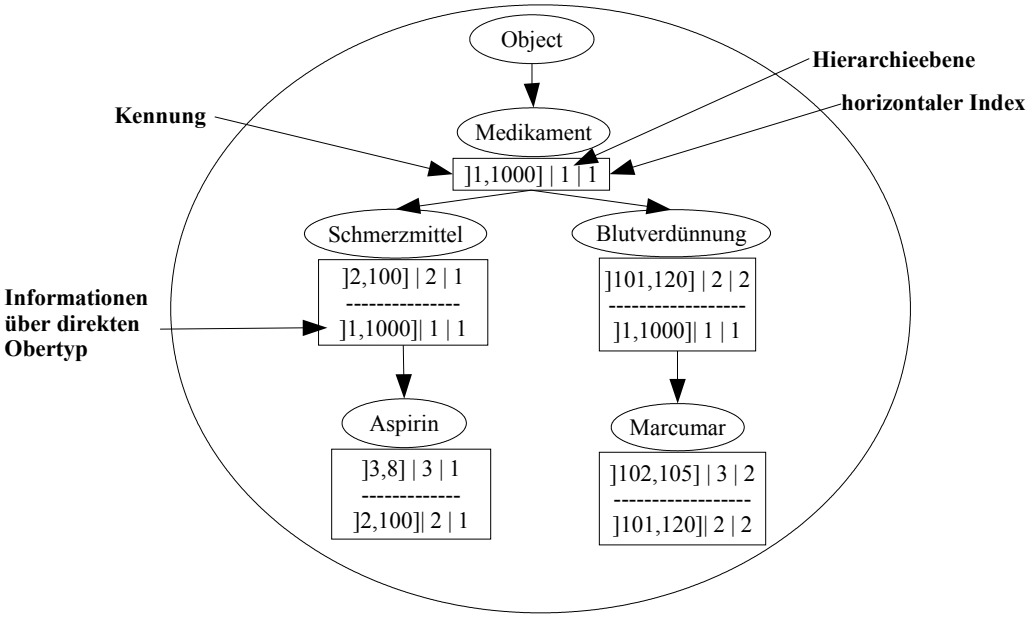


Abbildung 6.11.: Speicherorganisation einer Ontologie

cheradresse bietet, kann sich im Zusammenhang mit dem Aufbau einer Ontologie im Speicher positiv auf den dafür notwendigen Laufzeitaufwand auswirken. Die Ontologie-Adressen der einzelnen Elemente müssen nur einmal bestimmt werden und können dann direkt bei der Beschreibung der Ontologie auf dem Permanentspeicher gespeichert werden. Beim erneuten Einlesen der Ontologie können dann die Elemente direkt an die durch die Ontologie-Adressen festgelegten Positionen innerhalb der Arrays gespeichert werden, ohne dass Verkettungen zwischen den Elementen hergestellt werden müssen. Vor allem wenn man davon ausgeht, dass eine Ontologie im Laufe ihres Lebenszyklus mehrmals eingelesen werden muss, beispielsweise durch einen Neustart des Systems, kann sich dieser Vorteil auswirken, da die einmal berechnete Ontologie-Adresse immer wieder verwendet werden kann.

Die Ontologie-Adresse eines Elementes kann während der gesamten Lebensdauer dieses Elementes konstant bleiben. Werden neue Elemente in die Ontologie eingefügt, so werden diese innerhalb einer Hierarchieebene an das letzte Element angefügt. Einzig durch das Löschen eines Elementes kann es zu Lücken innerhalb einer Hierarchieebene kommen. Deshalb kann es von Zeit zu Zeit, wenn die Anzahl der Lücken zu einer gravierenden Speicherverschwendung durch leere ”Dummy-Einträge” in einem entsprechenden Array

führen, zu einer Reorganisation der Ontologie-Adressen kommen. Es ist fraglich ob es in der praktischen Anwendung tatsächlich desöfteren zu solchen Situationen kommt, da Ontologien, die grundlegende Zusammenhänge innerhalb einer Anwendungsdomäne abbilden, als relativ stabil angenommen werden können und es nicht zu erwarten ist, dass sie sich zumindest gegenüber dem Entfernen von Ontologie-Elementen als hochdynamische Strukturen erweisen.

6.5.2. Abbildung Ontologie-Element → Ontologie-Adressen

Eine zugriffseffiziente Organisationsform der Ontologien wurde eingeführt. Eine Frage wurde in diesem Zusammenhang aber noch nicht beantwortet. Dazu sei angenommen, dass im Zuge eines Korrektheitstests auf die in der semantischen Aktivitätenbeschreibung festgelegten Ontologie-Elemente zugegriffen werden muss, um die auszuwertende Regelmenge zu bestimmen. Für den Zugriff gibt es zwei verschiedene Möglichkeiten:

1. Die Ontologie-Adressen der Elemente sind in der semantischen Aktivitätenbeschreibung hinterlegt.
2. Lediglich die Namen der Ontologie-Elemente sind in der semantischen Aktivitätenbeschreibung genannt.

Beide Fälle haben gemeinsam, dass zu einem Ontologie-Element die zugehörige Ontologie-Adresse bestimmt werden muss. Im ersten Fall geschieht dies im Zuge der semantischen Aktivitätenbeschreibung, im zweiten Fall erst im Zuge des Korrektheitstests. Die erste Möglichkeit scheint auf den ersten Blick sinnvoll zu sein, da direkt beim Zugriff auf die Aktivität die Ontologie-Adressen der dort festgelegten Elemente verfügbar sind. Der zweite Blick offenbart jedoch gewisse Nachteile. Ein Grund, der gegen diese Strategie spricht, ist beispielsweise die erwähnte Problematik, dass es notwendig sein kann, die Ontologie-Adressen zu reorganisieren. In diesem Fall müssten die Adressen auch in allen semantischen Aktivitätenbeschreibungen angepasst werden. In einem komplexen System wie einem PMS sollten jedoch möglichst wenige Querbezüge und Seiteneffekte zwischen den einzelnen Systemkomponenten bestehen, um die Überschaubarkeit zu erhalten und die Fehleranfälligkeit so gering wie möglich zu halten.
Aus diesen Gründen ist die zweite Strategie für den Praxiseinsatz geeigneter. Zur Entkoppelung der Organisation der Ontologien von anderen Systembestandteilen muss eine effiziente Abbildung zwischen Elementnamen und Ontologie-Adressen etabliert werden. Aufgrund der zugriffseffizienten Eigenschaften bietet sich die Organisation der Objektnamen innerhalb eines balancierten binären Suchbaums an. Für jeden Suchbaumeintrag wird dort zusätzlich zum Objektnamen die Ontologie-Adresse gespeichert. Wie Abbildung 6.12 verdeutlicht, wird bei einem Zugriff auf ein Ontologie-Element zuerst die Ontologie-Adresse über den Suchbaum aufgelöst, und dann im zweiten Schritt auf die Daten des Ontologie-Elements über diese Adresse zugegriffen. Aufgrund der Eigenschaften des Logarithmus bleibt der Aufwand für die Bestimmung der Ontologie-Adresse eines Elements gering.

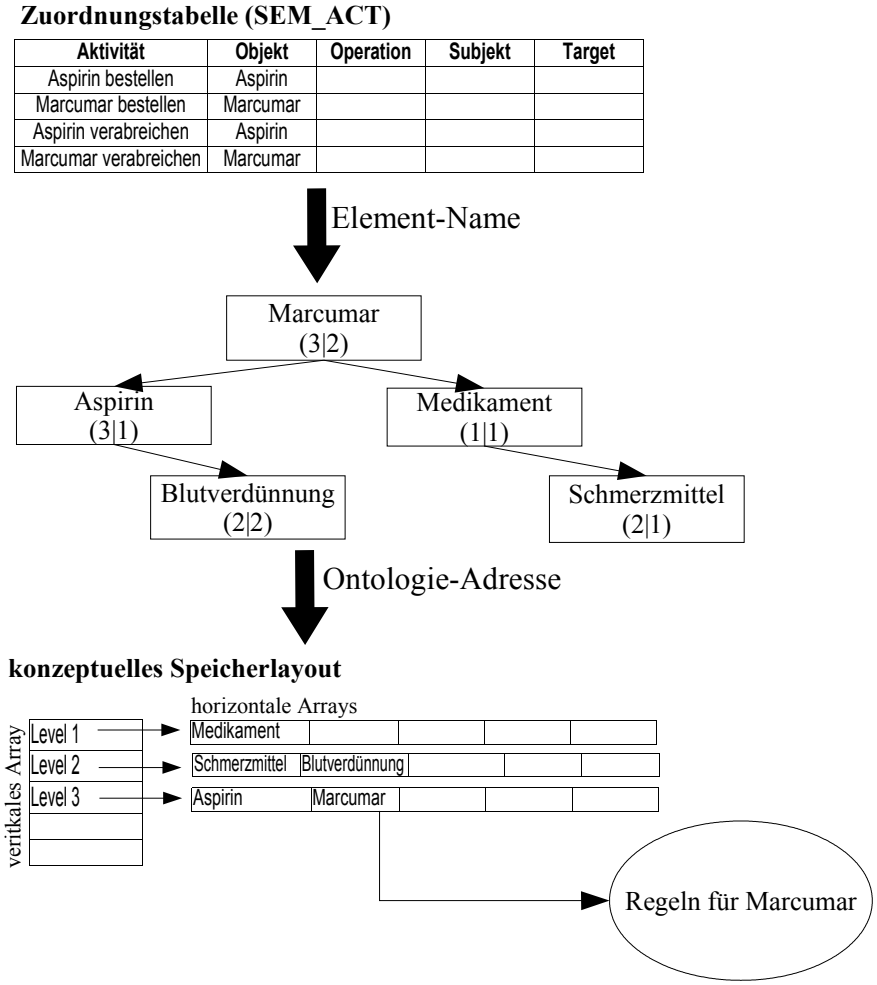


Abbildung 6.12.: Übersetzung Objektname → Ontologie-Adresse

6.5.3. Organisation der Regeln

Der effiziente Zugriff auf einzelne Ontologie-Elemente wird durch die Ontologie-Adressen ermöglicht. Damit ist es möglich, die Regelmenge für ein Ontologie-Element zu bestimmen, indem die Regeln aller Elemente auf dem Pfad bis zur Wurzel bestimmt werden. Es genügt also nicht der Zugriff auf ein einziges Ontologie-Element, sondern es müssen h Elemente zugegriffen werden, wenn h der Hierarchieebene des referenzierten Ontologie-Elementes entspricht. Es bleibt die Frage, ob es Möglichkeiten für die Organisation der Regeln gibt, die eine effizientere Bestimmung der Regelmenge zulassen.

Ein Kompromiss zwischen den Vorteilen der Regelvererbung und den Vorteilen, die sich durch eine direkte Speicherung sämtlicher auszuwertender Regeln innerhalb jeden Elements ergeben ist möglich, wenn innerhalb der Elemente statt der Regeln selbst lediglich

IDs der Regeln gespeichert werden. Dabei werden insbesondere auch die IDs der Regeln gespeichert, die von Obertypen geerbt werden. Die eigentlichen Regeln werden in einem zentralen Regelspeicher gehalten, in den mittels der IDs referenziert wird. Diese Kompromisslösung vereint folgende Vorteile in sich:

- Zentrale Regelspeicherung: Änderung an einer Regel nur an einer Stelle notwendig.
- Direkter Zugriff auf die vollständige Regelmenge eines Elements, ohne dem Vererbungspfad folgen zu müssen.
- Geringerer Speicherverbrauch als im Falle der direkten Speicherung sämtlicher Regeln innerhalb eines Ontologie-Elements, da die IDs nur den Speicherplatz für einen Ganzzahltyp pro Regel benötigen.

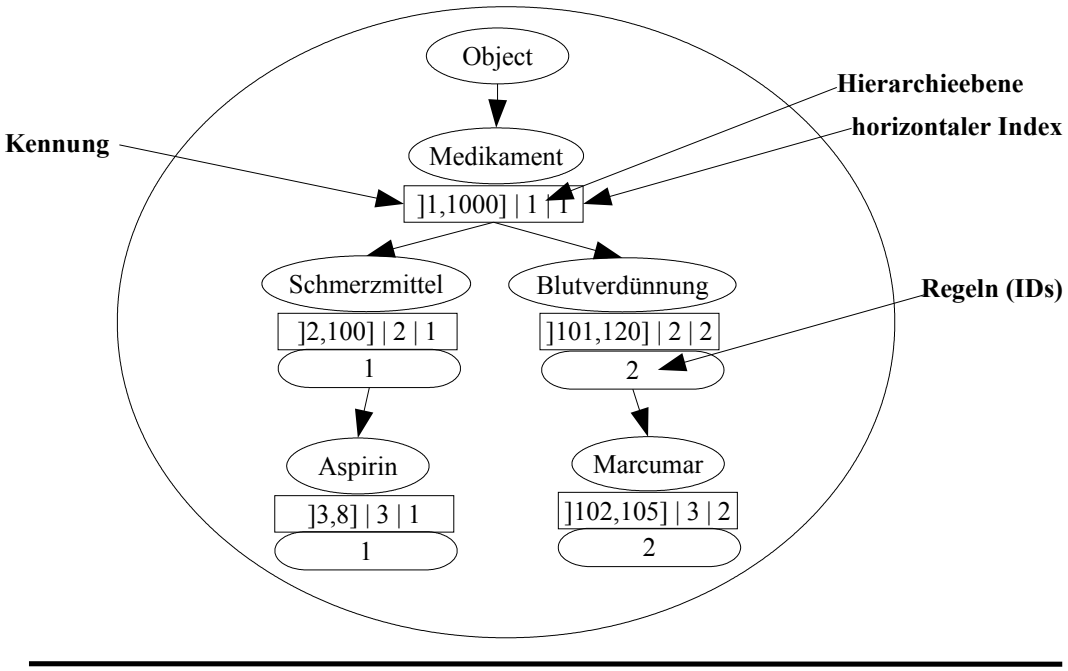
Diese Möglichkeit behält konzeptuell den Ansatz der Regelvererbung bei. Bei der Überführung der definierten Ontologien in entsprechende Laufzeitstrukturen werden jedoch für jedes Element zusätzlich zur eigenen Regelmenge auch die Regeln aller Obertypen gespeichert. Dabei kann überprüft werden, ob das aktuelle Element eine geerbte Regel überschreibt. Es muss dazu auf die tatsächliche Definition der Regeln im Regelspeicher zugegriffen werden. Eine Regel überschreibt eine andere Regel genau dann, wenn sie die selben Zielobjekte besitzen. In diesem Fall gilt die Regel aus dem Element, das auf der tieferen Hierarchieebene liegt. Die Regel aus einer höheren Hierarchieebene wird aus der Regelmenge des Elements entfernt.

Gegenüber echter Regelvererbung hat dieser Ansatz jedoch auch Nachteile. Beim Einfügen einer neuen Regel in ein Ontologie-Element muss die ID dieser Regel in die Regelmengen aller Elemente entlang des Vererbungspfades dieses Elements eingefügt werden. Dies erfordert zusätzlichen Rechenaufwand, der andererseits jedoch bei der Auswertung eines Ontologie-Elements und der dazu notwendigen Bestimmung der Regelmenge eingespart wird. Da die in diesem Ansatz einmal berechnete Regelmenge beliebig oft zugegriffen werden kann, ist die Bilanz des Laufzeitaufwands insgesamt als positiv zu bewerten. Abbildung 6.13 stellt die Visualisierung dieses Konzepts dar.

6.6. Speichereffizienz

Nachdem Überlegungen zur Optimierung des Laufzeitverhaltens angestellt wurden, ist die Frage zu untersuchen, ob diese Optimierungen aus der Sicht der Speicherbelastung sinnvoll sind. Dazu sollen Datenstrukturen der relevanten Elemente beispielhaft eingeführt und anschließend anhand dieser Datenstrukturen Überlegungen zur Speichernutzung angestellt werden.

Die relevanten Datenstrukturen betreffen die Elemente einer Ontologie, die Regeln, sowie die Elemente der Suchbäume für die Objekte und Zielobjekte eines Prozesses. Diese Datenstrukturen werden im Folgenden zunächst in einer C-artigen Syntax definiert.



zentraler Regelspeicher

ID	Regel
1	excludes Blutverdünnung
2	excludes Schmerzmittel
...	...

Abbildung 6.13.: Organisation der Regeln: zentraler Regelspeicher

```
struct ontologyElement{
   char[30] elementName;                                  30 Byte
   int min; //Typkennung: Intervalluntergrenze             4 Byte
   int max; //Typkennung: Intervallobergrenze              4 Byte
   int hierarchyLevel; //Ontologie-Adresse                 4 Byte
   int horizontalPosition;                                 4 Byte
   struct ruleNode* ruleIDs; //Liste der Regel-IDs         4 Byte
   int minParent; //Obertyp: Intervalluntergrenze          4 Byte
   int maxParent; //Obertyp: Intervallobergrenze           4 Byte
   int hLevelParent; //Ontologie-Adresse: Obertyp          4 Byte
   int hPositionParent;                                    4 Byte
}                                                         -------
                                                          66 Byte

struct ruleNode{
   int ruleID;                                             4 Byte
```

```
   struct ruleNode* next;                                    4 Byte
}                                                           -------
                                                            8 Byte
```

Die Struktur `ontologyElement` beschreibt die Daten, die zur Erfassung der Informationen eines Ontologie-Elements notwendig sind. Die Speicheranforderung kann wie dargestellt abgeschätzt werden. Dazu wird von einer Architektur ausgegangen, die pro Character 1 Byte, je Integer 4 Byte und für Pointer-Typen ebenfalls 4 Byte vorgibt. Die Addition der Speicheranforderungen der einzelnen Felder der Struktur ergibt einen Speicherplatzbedarf von 66 Byte pro Ontologie-Element. Zusätzlich muss für jede Regel, die einem Ontologie-Element zugeordnet ist, ein Element der Struktur `ruleNode` alloziert werden. Jedes Element dieser Struktur belegt 8 Byte Speicherplatz.

Ein Zahlenbeispiel kann nun die Auswirkungen der Ontologien auf die Speichernutzung verdeutlichen.
Es seien 1.000 Ontologie-Elemente im Speicher zu halten. Der benötigte Speicherplatz ergibt sich dann wie folgt:
sizeof(ontologyElement) * 1.000 = 66.000 Byte ≈ 64 KByte.
Zusätzlich wird für jede Regel, die einem Ontologie-Element zugeordnet ist, inklusive der geerbten Regeln, eine Struktur `ruleNode` alloziert, die mit jeweils 8 Byte zu Buche schlägt. Angenommen die durchschnittliche Anzahl der je Ontologie-Element gespeicherten Regel-IDs sei 20, dann wird zusätzlich folgender Speicherplatz für die Ontologien belegt:
sizeof(struct ruleNode) * 1.000 * 20 = 156,25 KByte

Die Speicheranforderungen durch die Ontologien sind demnach als moderat zu bezeichnen und stellen für den Praxiseinsatz kein Problem dar.
Im Folgenden wird eine Struktur eingeführt, die die Informationen über eine Semantikregel beinhaltet.

```
struct rule{
   int ruleID;                                              4 Byte
   int hierarchyLevel; //Ontologie-Adresse: Quellobjekt     4 Byte
   int horizontalPosition;                                  4 Byte
   int ruleType; //0: excludes, 1: dependsOn                4 Byte
   int hLevelTarget;//Ontologie-Adresse: Zielobjekt         4 Byte
   int hPositionTarget;                                     4 Byte
   int directionTarget;//0:keine spezifische Richtung       4 Byte
                       //1: post
                       //2: pre
   int hLevelDomain;//Ontologie-Adresse: domain-Element,    4 Byte
                    //z.B. Objekt in Operation-Ontology
   int hPositionDomain;                                     4 Byte
```

```
   int hLevelRelatedOp; //Ontologie-Adresse: Operation   4 Byte
   int hPositionRelatedOp;//für Zielobjekt-Kontext       4 Byte
   int hLevelRelatedSubj; //Ontologie-Adresse: Subjekt   4 Byte
   int hPositionRelatedSubj;//für Zielobjekt-Kontext     4 Byte
   int hLevelRelatedTar; //Ontologie-Adresse: Target     4 Byte
   int hPositionRelatedTar;//für Zielobjekt-Kontext      4 Byte
}                                                        -------
                                                         60 Byte
```

Auch die Struktur `rule`, die alle Daten zu einer semantischen Regel enthält, muss auf ihren Speicherverbrauch untersucht werden. Wie dargestellt belegt jede Ausprägung dieser Struktur 60 Byte. Ein Beispiel kann auch hier Entwarnung für eine Gefährdung eines Praxiseinsatzes geben.
Es seien 1.000 Ontologie-Elemente festgelegt, wobei jedem Element durchschnittlich 10 Regeln zugeordnet seien, also insgesamt 10.000 Regeln existieren. Die Speicherbelegung ergibt sich mit diesen Zahlen als
sizeof(struct rule) * 10.000 $\approx$ 586 KByte.

Das Beispiel zeigt, dass auch die Speicherbelastung durch die Regeln im für den Praxiseinsatz vertretbaren Rahmen bleibt.
Schließlich werden Strukturen eingeführt, die die Einträge für die Objekt- und Zielobjekt-Suchbäume repräsentieren.

```
struct objectNode{
   int min; //Typkennung: Intervalluntergrenze            4 Byte
   int max; //Typkennung: Intervallobergrenze             4 Byte
   struct objectNode* left;                              4 Byte
   struct objectNode* right;                             4 Byte
   struct activityID* acitivityIDs;//Liste der IDs der   4 Byte
         //Aktivitäten des Prozesses, denen das
         //Objekt zugeordnet ist.
}                                                        --------
                                                         20 Byte

struct targetObjectNode{
   int min; //Typkennung: Intervalluntergrenze            4 Byte
   int max; //Typkennung: Intervallobergrenze             4 Byte
   struct objectNode* left;                              4 Byte
   struct objectNode* right;                             4 Byte
   struct ruleActReference* ruleActReferences;           4 Byte
          //Referenzen auf die zugehörigen Regeln
          //und die Aktivitäten
}                                                        -------
                                                         20 Byte
```

```
struct activityID{
   int ID;                                          4 Byte
   struct activityID* next;                         4 Byte
}                                                  --------
                                                    8 Byte

struct ruleActReference{
   int ruleID;                                      4 Byte
   int activityID;                                  4 Byte
   struct ruleActReference* next;                   4 Byte
}                                                  --------
                                                   12 Byte
```

Wie dargelegt ist es aus Effizienzgründen notwendig, die Objekte eines Prozesses, sowie die Zielobjekte der semantischen Regeln in balancierten binären Bäumen zu organisieren. Die einzelnen Elemente eines solchen Suchbaumes benötigen für den Suchalgorithmus die Intervallgrenzen, sowie Zeiger auf die zwei Nachfolger eines Suchbaumknotens. Um weiter das Zutreffen einer Regel auswerten zu können, ist es notwendig, die Position des Objekts bzw. die Position des Quellobjekts einer Regel innerhalb des Prozesses zu kennen. Daher muss in jedem Element der Suchbäume eine Liste mit Referenzen auf die Aktivitäten des Prozesses gespeichert werden, denen das jeweilige Objekt in der semantischen Aktivitätenbeschreibung zugeordnet ist. Zusätzlich zur Referenz auf die Aktivität müssen im Zielobjekt-Suchbaum auch die Regel-IDs der Regeln eines Zielobjekts gespeichert werden. Wie dargestellt benötigt ein Element eines Suchbaums 20 Byte. Weiter kommen die Speicheranforderungen durch die Aktivitätsreferenzen und im Falle des Zielobjekt-Suchbaums die Referenzen auf die Regeln hinzu. Ein Zahlenbeispiel anhand des Objektsuchbaums verdeutlicht die Dimension der Speicherbelastung durch die Suchbäume:
1.000 verschiedene Objekte seien in einem Prozess enthalten. Der Speicherbedarf ergibt sich in diesem Fall als
sizeof(struct objectNode) * 1.000 $\approx$ 20 KByte.
Weiter sei angenommen, dass jedes Objekt in maximal drei unterschiedlichen Aktivitäten vorkommt. Der zusätzliche Speicherbedarf ergibt sich dann als
sizeof(struct activityID) * 1.000 * 3 $\approx$ 23 KByte.

Diese Zahlen belegen, dass die zur Effizienzsteigerung notwendigen binären Suchbäume keine den Praxiseinsatz gefährdenden Speicheranforderungen stellen.

Zusammenfassend zeigt die Untersuchung der Speicheranforderungen, dass diese auch unter der Berücksichtigung der zur Effizienzsteigerung notwendigen Hilfsstrukturen wie den binären Suchbäumen moderat bleiben. Die Zahlen, die genannt wurden, sind nicht als genaue Werte einer konkreten Implementierung zu verstehen und können je nach Architektur und System unterschiedlich ausfallen. Trotzdem zeigen die Untersuchungen

anhand konkreter Zahlen die ungefähren Ausmaße in denen sich die Speicherbelastung bewegt.

6.7. Besonderheiten durch Mehrfachklassifizierung

In den bisherigen Überlegungen zur Umsetzbarkeit der theoretischen Konzepte wurde die Mehrfachklassifizierung nicht berücksichtigt. Mehrfachklassifizierung, die in Kapitel 4.5 eingeführt wurde, bedeutet, dass ein Element einer Ontologie mehr als einem Obertyp zugeordnet sein kann. So kann `Aspirin` beispielsweise als Untertyp sowohl von `Schmerzmittel` als auch von `Blutverdünnungsmittel` definiert werden. Wie in Kapitel 4.5 dargestellt, bringt die Mehrfachklassifizierung das Problem inkonsistenter Regelmengen mit sich. Bei der Berechnung der Regelmengen müssen deshalb die dort beschriebenen Mechanismen angewendet werden, um konfliktfreie Regelmengen zu erhalten. Dies geschieht bei der Überführung einer Ontologie in die Laufzeitstrukturen, so dass kein zusätzlicher Laufzeitaufwand bei der Auswertung der Ontologie-Elemente im Zuge eines semantischen Korrektheitstests entsteht. Im Gegensatz dazu ist die effiziente Erkennung von Typbeziehungen und das effiziente Auffinden von Objekten und Zielobjekten des Prozesses entscheidend für das Laufzeitverhalten eines Korrektheitstests. Es stellt sich nun die Frage, wie die eingeführten Konzepte zur effizienten Umsetzung dieser performance-kritischen Aspekte auf den Fall der Mehrfachklassifizierung übertragen werden können und welche neuen Probleme sich aus der Mehrfachklassifizierung ergeben.

Die effiziente Erkennung von Typbeziehungen beruht auf einer Typkennung, die in Form eines Intervalls vollständig in den Intervallen der Obertypen des einfachen Vererbungspfades liegt. Der Wertebereich für verschiedene Untertypen eines Typs wird also entsprechend der Intervallorganisation partitioniert. Beim Vergleich zweier Intervalle gilt immer eine der beiden folgenden Möglichkeiten:

- Eines der Intervalle liegt vollständig innerhalb des anderen Intervalls.
- Die Intervalle sind disjunkt.

Ist ein Ontologie-Element mehrfach-klassifiziert, so werden ihm mindestens zwei verschiedene direkte Obertypen zugeordnet. Wird nun dem mehrfach-klassifizierten Element eine Typkennung zugewiesen, stellt sich die Frage, in welchem der Intervalle der Obertypen die neue Typkennung liegen soll. Gibt es genau eine Typkennung, so ergibt sich offensichtlich das Problem, dass die Typbeziehung auf einfache und effiziente Weise nur mit einem der Obertypen bestimmt werden kann.

Um Typbeziehungen mit jedem Obertyp eines Elements ausdrücken zu können, ist es notwendig, dass ein Element mehrere verschiedene Typkennungen in Form von Intervallen besitzen kann. Für jeden direkten Obertyp wird dann dem Element eine Typkennung als Intervall zugewiesen, das vollständig im Typintervall des jeweiligen Obertyps liegt.

Werden von dem mehrfach-klassifizierten Element wiederum Untertypen gebildet, so besitzen diese mindestens soviele Typkennungen, wie das Element vorgibt.

Müssen zwei Elemente auf eine Typbeziehung untersucht werden, so werden alle Typkennungen der zu vergleichenden Typen miteinander verglichen, solange bis entweder eine Typbeziehung festgestellt wurde oder alle Typkennungen ausgewertet wurden. Mehrfachklassifizierung führt also zu einem erhöhten Laufzeitaufwand durch Auswertung aller Typkennungen eines Objekts. Dieser zusätzliche Aufwand hält sich jedoch in Grenzen wenn man berücksichtigt, dass in einem praxistauglichen und übersichtlichen Typsystem ein Element nicht zu beliebig vielen anderen Typen in einer direkten Untertyp-Beziehung stehen sollte.

Die bereits eingeführte Datenstruktur `ontologyElement` muss wie folgt abgeändert werden, um Mehrfachklassifizierung mit mehreren Typkennungen zu unterstützen.

```
struct ontologyElement{
   char[30] elementName;                                    30 Byte
   struct typeInfo* typeIDs;//Typinformationen für           4 Byte
                            //jeden Obertyp
   struct ruleNode* ruleIDs; //Liste der Regel-IDs           4 Byte
}                                                          --------
                                                           38 Byte

struct typeInfo{
   int min; //Typkennung: Intervalluntergrenze               4 Byte
   int max; //Typkennung: Intervallobergrenze                4 Byte
   int hierarchyLevel; //Ontologie-Adresse                  4 Byte
   int horizontalPosition;                                  4 Byte
   int minParent; //Obertyp: Intervalluntergrenze           4 Byte
   int maxParent; //Obertyp: Intervallobergrenze           4 Byte
   int hLevelParent; //Ontologie-Adresse: Obertyp          4 Byte
   int hPositionParent;                                     4 Byte
   struct typeInfo* next;                                   4 Byte
}                                                          --------
                                                           36 Byte

struct ruleNode{
   int ruleID;                                              4 Byte
   struct ruleNode* next;                                   4 Byte
}                                                          -------
                                                           8 Byte
```

Die Informationen über die Obertypen und über die einzelnen Typkennungen sind in einer eigenen Struktur `typeInfo` ausgelagert. Die einzelnen Typinformationen werden

in einer Liste gespeichert, die in der Struktur `ontologyElement` referenziert wird. Die Speicherbelastung erhöht sich durch die Mehrfachklassifizierung, da für ein Element die Typinformationen zu jedem Obertyp gespeichert werden müssen. Diese zusätzliche Belastung des Speichers ist jedoch linear zur Anzahl der im Element mündenden Vererbungsstränge und stellt damit kein Problem dar. Mehrfachklassifizierung kann also sowohl unter dem Aspekt der Laufzeiteffizienz, als auch unter dem Aspekt der Speicherbelastung bedenkenlos umgesetzt werden. Abbildung 6.14 zeigt eine Ontologie, die mehrfachklassifizierte Elemente enthält.
Mit dieser Ontologie können folgende Fragen potentieller Typbeziehungen beantwortet

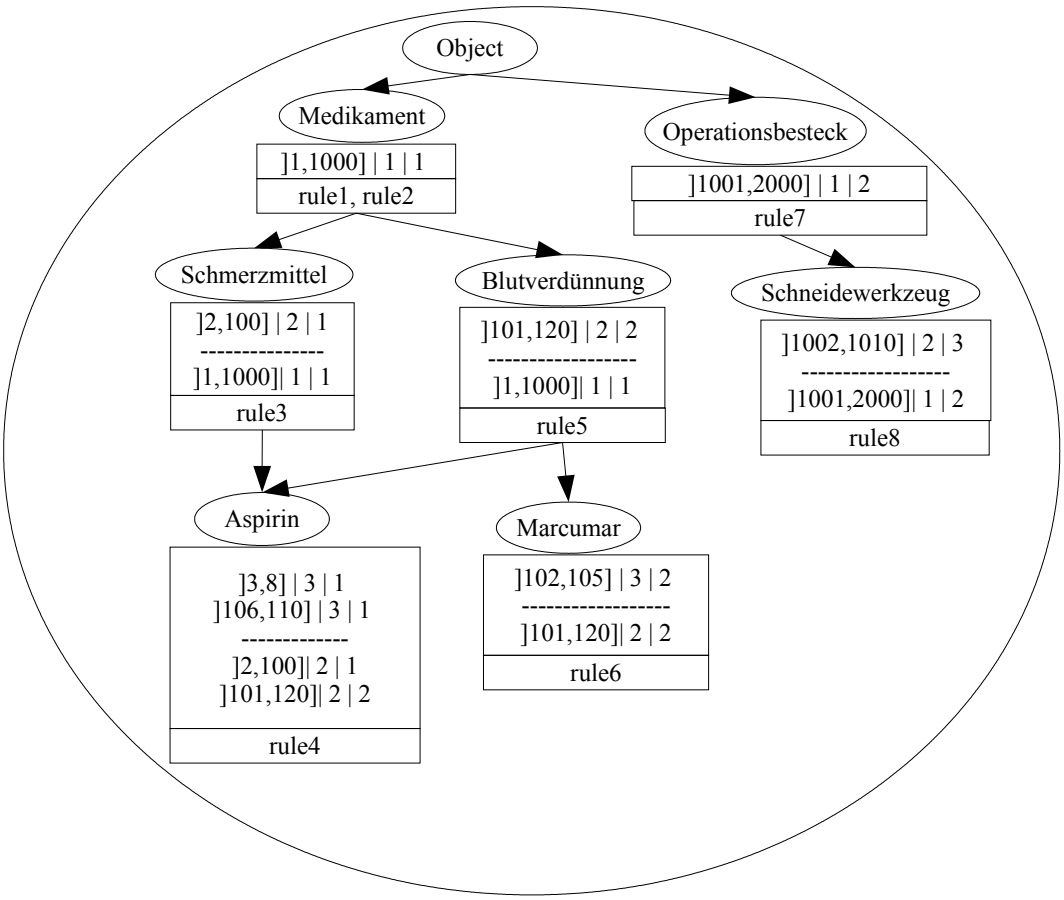


Abbildung 6.14.: Ontologie mit Mehrfachklassifizierung

werden:

1. Ist Aspirin ein Untertyp von `Schmerzmittel`?
   Typkennung `Schmerzmittel`: $]2, 100]$
   Typkennungen `Aspirin`: $]3, 8]$, $]106, 110]$
   Auswertung: $3>2$ und $8<100 \Longrightarrow$ Aspirin ist Untertyp von `Schmerzmittel`.

2. Ist Aspirin ein Medikament zur Blutverdünnung?
   Typkennung `Blutverdünnung`: $]101, 120]$
   Typkennungen `Aspirin`: $]3, 8]$, $]106, 110]$
   Auswertung:

   a) $3<101 \Longrightarrow$ Typkennung liegt nicht im Intervall des Typs `Blutverdünnung`. Weiter mit nächster Typkennung.

   b) $106>101$ und $110<120 \Longrightarrow$ `Aspirin` ist Untertyp des Typs `Blutverdünnung`.

Im Zuge der Mehrfachklassifizierung wurde die Umsetzung innerhalb der Ontologien betrachtet. Ob diese Umsetzung der Mehrfachklassifizierung Auswirkungen auf die Organisation der Suchbäume eines Prozesses haben, die anhand der Typkennung aufgebaut werden, muss im folgenden Kapitel untersucht werden.

6.7.1. Speicherung in Suchbäumen

Die Speicherung von Objekten und Zielobjekten der Regeln des Prozesses geschieht in balancierten binären Bäumen mittels der Typkennung. Mehrfach klassifizierte Objekte besitzen jedoch mehrere Typkennungen, so dass die Frage aufgeworfen wird, wie solche Objekte in den entsprechenden Suchbäumen gespeichert werden können. Es ergeben sich dazu zwei Möglichkeiten

1. Verfahren: Die erste Kennung in der Liste der Typkennungen des Objektes wird als primäre Kennung verwendet. Diese primäre Kennung wird als Kriterium für das Einfügen des Objektes in den Suchbaum verwendet.
2. Verfahren: Ein Objekt, das mehrere Typkennungen besitzt, wird mehrfach in den jeweiligen Suchbaum eingefügt, so dass für jede der Typkennungen ein Suchbaumeintrag existiert.

Muss im ersten Verfahren das Objekt gesucht werden, so kann einfach die erste Typkennung als Suchkriterium verwendet werden. Es ergibt sich allerdings ein Problem, wenn das Objekt als Untertyp einer Suchanfrage erkannt werden soll. In diesem Fall wird das Objekt lediglich für denjenigen Obertyp als Untertyp erkannt, dessen Typkennung die primäre Kennung des mehrfach-klassifizierten Objekts enthält. Für alle anderen Obertypen wird das mehrfach-klassifizierte Objekt nicht als Ergebnis der Suche geliefert. Deutlich wird dies an folgendem Beispiel:
Es seien folgende Objekte im Prozess vorhanden: Object1 $]11, 20]$, Object5 $]21, 28]$ und das zweifach klassifizierte Objekt Object11 $]2, 5],]32, 34]$.
Daraus ergibt sich nach dem ersten Verfahren der in Abbildung 6.15 dargestellte Suchbaum.
Wie zu sehen ist, wird das mehrfach klassifizierte Objekt anhand der ersten Kennung $]2, 5]$ in den Suchbaum eingefügt. Im Zuge einer Änderungsoperation soll nach Vorkommen von Objekten des Typs $]30, 40]$ gesucht werden, wobei auch Untertypen als Treffer erkannt werden sollen. Object11 müsste in der Trefferliste als Untertyp auftauchen. Da

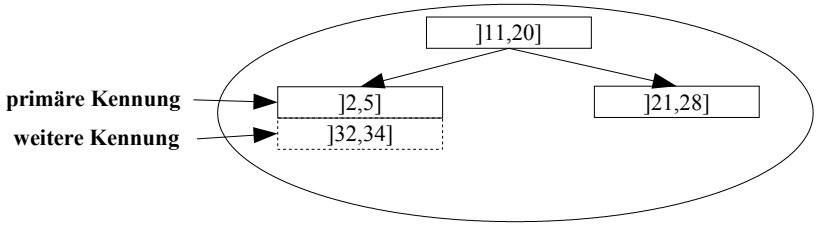


Abbildung 6.15.: Suchbaum - mehrfach-klassifiziertes Objekt (1. Verfahren)

Object11 jedoch nicht anhand der Typkennung $]32, 34]$ in den Suchbaum eingefügt wurde, die vollständig in der Typkennung des gesuchten Typs enthalten ist, wird Object11 nicht als Treffer erkannt.

Das erste Verfahren eignet sich wie am Beispiel deutlich wurde nicht für eine Umsetzung, da ein mehrfach-klassifiziertes Objekt nicht als Untertyp aller seiner Obertypen gefunden wird. Die zweite Möglichkeit geht den Weg, das Objekt für jede seiner Typkennungen in den Suchbaum einzufügen. Damit wird bei der Suche nach jedem Obertyp das Objekt als Untertyp gefunden. Abbildung 6.16 zeigt den Aufbau des Suchbaums nach dem zweiten Verfahren. Wird das bereits eingeführte Beispiel darauf angewendet, so wird dieses Mal Object11 als Treffer gefunden. Die zweite Alternative ermöglicht also die vollständige Einbindung mehrfach-klassifizierter Objekte in die bestehenden Suchbäume und stellt damit die Alternative der Wahl für eine Umsetzung dar. Es muss jedoch beachtet werden, dass beim Einfügen eines mehrfach-klassifizierten Objekts oder beim Löschen eines solchen Objekts mehrere Einträge in den Suchbaum eingefügt bzw. aus dem Suchbaum gelöscht werden müssen. Die Mehrfachklassifizierung wird also mit einem erhöhten Laufzeitaufwand gegenüber einfach klassifizierten Objekten erkauft.

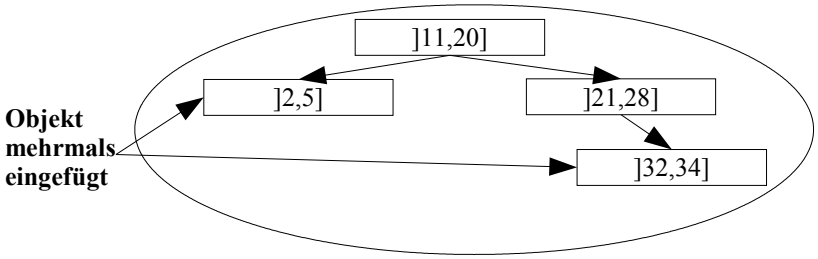


Abbildung 6.16.: Suchbaum - mehrfach-klassifiziertes Objekt (2. Verfahren)

6.8. Umgang mit Intervallüberläufen

Ontologien sind dynamische Strukturen, die auch im laufenden Betrieb erweitert werden können. Aufgrund der Organisation der Typkennungen in Form von Intervallen kann es vorkommen, dass durch Einfügen neuer Elemente in eine Ontologie ein Intervallüberlauf auftritt, d.h. alle möglichen Kennungen für Untertypen eines Obertyps sind bereits vergeben. Grundsätzlich existieren folgende Möglichkeiten, um mit solchen Intervallüberläufen umzugehen:

- Reorganisation der Intervalle.
- Umgehen der Reorganisation durch gesonderte Behandlung übergelaufener Intervalle.

Der Vorteil der Neuberechnung liegt in der sauberen Neuordnung der Typkennungen, ohne dass zur Auswertung der Typbeziehungen die Algorithmen angepasst werden müssen. Die Reorganisation der Intervalle ist jedoch aufwendig und zieht auch Anpassungen der Einträge in den Suchbäumen der Prozesse nach sich, da diese anhand der Typkennungen aufgebaut sind. Änderungsoperationen an einem Prozess bzw. an Instanzen dieses Prozesses sind solange nicht möglich, bis die Suchbäume entsprechend der neuen Typkennungen angepasst wurden. Diese Laufzeitnachteile lassen die Strategie der Neuberechnung für den Praxiseinsatz als nicht geeignet erscheinen.

Die zweite Strategie vermeidet nach einem Intervallüberlauf infolge des Einfügens eines Ontologie-Elements die Reorganisation der Typkennungen. Der Intervallüberlauf wird zugelassen und ist dadurch gekennzeichnet, dass die Intervalluntergrenze die Intervallobergrenze übersteigt. Abbildung 6.17 zeigt eine Beispiel-Ontologie, in der mehrmals ein Intervallüberlauf aufgetreten ist. Object4, Object5, Object6 und Object7 sind alles Elemente, deren Intervalluntergrenzen größer sind als deren Intervallobergrenzen.

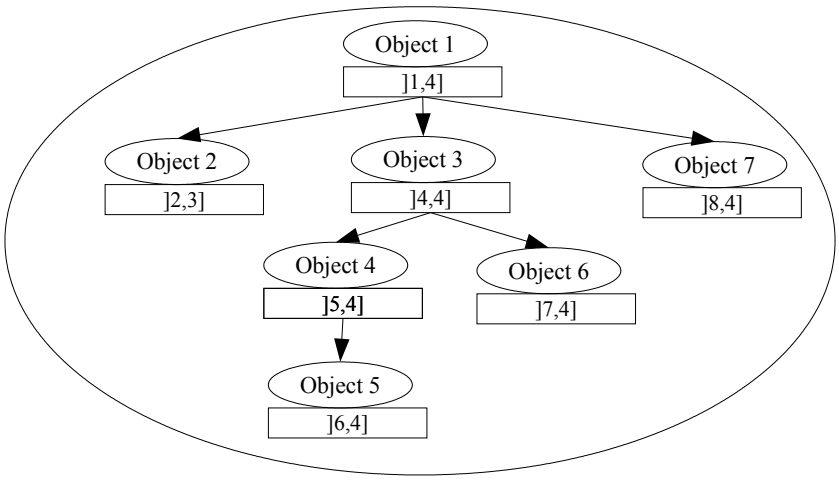


Abbildung 6.17.: Intervallüberlauf

Die Objekte zeigen ein Problem, wenn Intervallüberläufe einfach zugelassen werden. So ist nicht mehr klar, in welchen Typverhältnissen die Objekte stehen. Beispielsweise kann anhand der Typintervalle von Object4 $(]5, 4])$, Object5 $(]6, 4])$ und Object6 $(]7, 4])$ nicht mehr entschieden werden, welche Typbeziehungen zwischen diesen Objekten bestehen. Um die korrekte Auswertung der Typbeziehungen weiter garantieren zu können, müssen die Ontologie-Elemente um weitere Informationen erweitert werden:

- Speicherung des letzten regulären Obertyps, der durch ein reguläres, nicht übergelaufenes Intervall gekennzeichnet ist.
- Speicherung einer zweiten Kennung des Elements, das für alle Untertypen des letzten regulären Obertyps eindeutig ist.

Die Speicherung des letzten regulären Obertyps ist notwendig, um das Element mit einem beliebigen anderen Typ aus der Ontologie auf eine potentielle Typbeziehung untersuchen zu können. Folgende Situationen können dabei auftreten:

- Das Intervall des zweiten Typs ist regulär und es ist disjunkt zum letzten regulären Intervall des Elements mit übergelaufener Typkennung.
- Das Intervall des zweiten Typs ist regulär und es enthält den als letztes reguläres Intervall angegebenen Eintrag des Elements mit übergelaufener Typkennung.
  $\Longrightarrow$ Der zweite Typ stellt einen Obertyp zum Element mit übergelaufener Typkennung dar.
- Das Intervall des zweiten Typs ist regulär und es liegt innerhalb des regulären Intervalls des Elements mit übergelaufener Typkennung.
  $\Longrightarrow$ Die beiden Typen stehen in keiner Typbeziehung.
- Beim zweiten Typ handelt es sich ebenfalls um ein Element mit übergelaufener Typkennung. Ein Vergleich der beiden Einträge für das jeweils letzte reguläre Intervall ergibt eine Aussage, ob eine potentielle Typbeziehung besteht. Es besteht eine potentielle Typkennung, falls diese beiden Einträge identisch sind.

Die zweite Kennung dient dazu, die im letzten genannten Fall auftretende potentielle Typbeziehung endgültig zu klären. Für eine Umsetzung dieser zweiten Typkennung kann die bereits in Kapitel 6.1.2.1 eingeführte Idee der hierarchisch aufgebauten Strings benutzt werden. Die Nachteile, die dafür angeführt wurden, sind für den Fall des Intervallüberlaufs nicht mehr praxisrelevant. Eine Ontologie, bei der so viele Einfügungen mit der Folge von Intervallüberläufen aufgetreten sind, dass der String-Vergleich zu Performance-Engpässen führt, sollte vollständig reorganisiert werden. Abbildung 6.18 stellt die bereits in Abbildung 6.17 eingeführte Ontologie mit den erweiterten Informationen dar.

Die Typbeziehungen zwischen Object4, Object5, Object6 und Object7 können nun wieder ausgewertet werden. Ein Vergleich der Einträge für das letzte reguläre Intervall zwischen Object4 und Object6 ergibt eine potentielle Typbeziehung. Deshalb wird die zweite Kennung miteinander verglichen, d.h. die Strings ”1” und ”2”. Das Ergebnis dieses Vergleichs ist, dass keine Typbeziehung zwischen Object4 und Object6 besteht. Die selbe

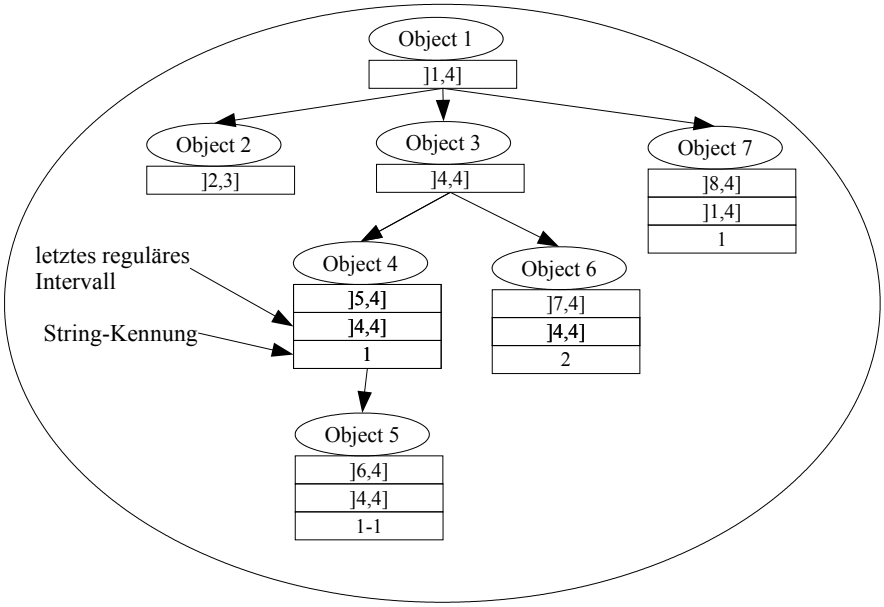


Abbildung 6.18.: Intervallüberlauf - Zusatzinformationen

Auswertung zwischen Object4 und Object5 ergibt, dass Object5 ein Untertyp von Object4 ist.
Muss die Frage nach einer Typbeziehung zwischen Object4 und Object7 beantwortet werden, ergibt bereits der Vergleich der Einträge für das jeweils letzte reguläre Intervall, dass kein Typzusammenhang zwischen Object4 und Object7 besteht.

Bisher wurden Intervallüberläufe nur für den Fall der einfachen Klassifizierung beschrieben. Das Vorgehen kann jedoch direkt auf den Fall der Mehrfachklassifizierung übertragen werden. In einem mehrfach-klassifizierten Element können mehrere der Typkennungen gleichzeitig überlaufen. In einer solchen Situation müssen dann für jeden Obertyp, für den das Intervall überläuft, die beschriebenen Maßnahmen ergriffen und die zusätzlichen Informationen gespeichert werden.

Das Zulassen von Intervallüberläufen hat Auswirkungen auf die Suchbäume eines Prozesses. Abbildung 6.19 zeigt einen Beispielsuchbaum, der Elemente enthält, deren Typkennungen übergelaufen sind.
Die Suche nach einem Objekt in einem Suchbaum vollzieht sich nun nach folgendem Muster:

1. Fall: Beide Intervalle, also das zu suchende und das Intervall des aktuellen Suchbaumeintrages sind reguläre Intervalle. Die Auswertung unterscheidet sich nicht vom bisherigen Vorgehen.

2. Fall: Es handelt sich bei mindestens einem der Intervalle um ein nicht reguläres Intervall. Bei jedem Suchschritt tritt eine der folgenden Situationen auf:

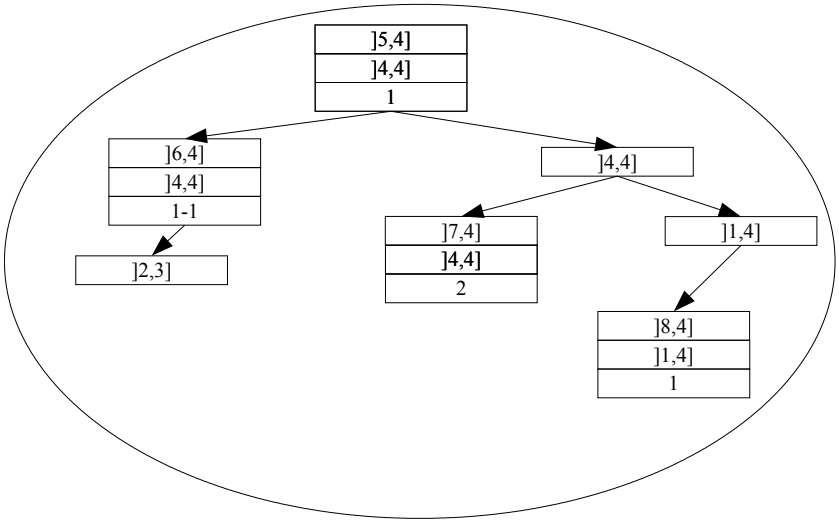


Abbildung 6.19.: Suchbaum - übergelaufene Intervalle

a) Die verglichenen Intervalle sind identisch.
$\Longrightarrow$ gesuchtes Objekt gefunden

b) Eins der Intervalle ist regulär: Vergleich dieses Intervalls mit dem Eintrag für das letzte reguläre Intervall im zweiten Element und Untersuchung auf eine Typbeziehung.

c) Beide Intervalle sind nicht regulär: Vergleich der beiden Einträge für das letzte reguläre Intervall. Die Auswertung kann verschiedene Ergebnisse haben:

   i. nicht identisch: Es besteht keine Typbeziehung zwischen den Objekten.
   $\Longrightarrow$ Für die Entscheidung, wie weiter verzweigt werden soll, dienen die Einträge für die letzten regulären Intervalle als Grundlage.

   ii. identisch: Die String-Kennungen der zwei Objekte müssen ausgewertet werden. Diese Auswertung führt zu einer der folgenden Situationen:

   - Strings sind identisch: das Objekt wurde gefunden
   - Das gesuchte Objekt ist ein Untertyp.
   $\Longrightarrow$ Es wird nach links verzweigt.
   - Das gesuchte Objekt ist ein Obertyp.
   $\Longrightarrow$ Es wird nach rechts verzweigt
   - Es handelt sich um disjunkte Strings:
   - Nach links verzweigen, wenn die Intervalluntergrenze des gesuchten Objekts kleiner als die des Suchbaumelements ist.
   - Andernfalls nach rechts verzweigen.

7

Verwandte Arbeiten

Die Einbeziehung der Semantik basiert auf existierenden Ansätzen zur Modellierung und Ausführung von Prozessen und deren dynamische Änderung im laufenden Betrieb und greift auf entsprechende Forschungsarbeiten zurück [AB02, CCPP98, RRD04, Wes01]. Diese Ansätze konzentrieren sich dabei auf syntaktische Fragestellungen, um beispielsweise Verklemmungssituationen nach Änderungen zur Laufzeit zu vermeiden oder um die korrekte Versorgung von Aktivitäten mit Eingabeparametern sicherzustellen. Die Sicherstellung der semantischen Korrektheit geht über diese syntaktischen Fragestellungen hinaus. Die Auswertung der semantischen Korrektheit setzt die Erfassung der Semantik der einzelnen Aktivitäten voraus. Dies stellt einen Schnittpunkt zum prozessorientierten Wissensmanagement dar. Auch hier sind die einzelnen Prozessschritte Gegenstand der zugrundeliegenden Mechanismen. [Hei01, S.6] schreibt dazu:

> *Ausgehend von unserer Annahme, dass wir Wissen in unserer täglichen Arbeit in den Geschäftsprozessen nutzen, sind die Geschäftsprozesse auf der Detaillierungsebene Arbeitsaufgabe zu beschreiben.*

Das Interesse des prozessorientierten Wissensmanagements richtet sich jedoch nicht auf die Sicherstellung der Korrektheit von Prozessen, sondern auf die Unterstützung von Benutzern bei wissensintensiven Arbeitsschritten. [Goe02, S.3-4] beschreibt die Herausforderungen des prozessorientierten Wissensmanagements als die "*[...] bedarfsgerechte Verfügbarkeit von unternehmensrelevantem Wissen*". Ansätze dazu finden sich in prozessbasierten Wissens-Management-Systemen wie DECOR [ABN01], KnowMore [ABH00] und Frodo TaskMan [EAB03]. Die Unterstützung geschieht durch die an der konkreten

Bearbeitungssituation ausgerichtete Bereitstellung relevanter Informationen aus den unterschiedlichsten Datenquellen, wie zugeordneten Dokumenten, Datenbanken und Datenwerten des Prozesskontexts. FRODO TaskMan geht über den von KnowMore verfolgten Ansatz hinaus und ermöglicht die Modellierung und Ausführung unvollständig spezifizierter Prozesse, die zur Laufzeit vervollständigt werden können. FRODO TaskMan unterstützt überdies die dynamische Adaption von Prozessinstanzen zur Laufzeit, wohingegen keine Möglichkeit zur Änderung von Prozessschemata geboten wird.

Auch im Bereich des fallbasierten Schließens (CBR) existieren Ansätze, die sich mit Prozess-Management auseinandersetzen [KSL02, MZ03]. So kann CBR beispielsweise für die Modellierung von Prozessen, für die Konfiguration komplexer Prozess und für den Umgang mit Ausnahmesituationen verwendet werden. Eine Variante stellt das Conversational Case-Based Reasoning (CCBR) dar [WRWR05, RWRW05]. Auch diese Herangehensweise hat zum Ziel, die Qualität der Prozesse zu verbessern. Dies wird durch die Annotierung von Änderungen erreicht. Diese Benutzerinformationen werden in Form von Frage-Antwort-Paaren im System hinterlegt. Damit kann in einer ähnlichen Situation bei der Änderung einer weiteren Instanz bestimmt werden, ob in der Vergangenheit bereits eine ähnliche Änderung durchgeführt wurde. Zusätzlich können die Benutzer Änderungen bewerten. Es ist sehr wahrscheinlich, dass eine Änderung, die zu einem unerwünschten Ergebnis führt, durch die Benutzer des Systems negativ bewertet wird. Damit wird eventuell von der entsprechenden Änderung einer weiteren Instanz oder von der Änderung des Prozessschemas Abstand genommen. Dies kann die Qualität der Prozesse verbessern. Dieser Mechanismus des CCBR und die semantische Überprüfung von Prozessen ergänzen sich auf ideale Weise. So ist trotz semantischer Korrektheit eines Prozesses nicht sichergestellt, dass dieser Prozess optimal strukturiert ist. Diese Beurteilung bleibt den Bearbeitern mit ihrem persönlichen Erfahrungsschatz vorbehalten.

Die in dieser Arbeit dargestellten Konzepte zur Umsetzung der Semantik beschränken sich auf statische Aspekte. Das bedeutet, dass die Bewertung der semantischen Korrektheit eines Prozesstyps oder einer Instanz anhand der statischen Beschreibung der Prozessschritte durchgeführt wird. Nicht berücksichtigt werden dabei Aspekte der Semantik, die sich beispielsweise in Form von dynamischen Geschäftsregeln darstellen [BK05, AA01]. Dieser Aspekt kann dadurch abgegrenzt werden, dass die Evaluierung solcher Geschäftsregeln während der Ausführung einer Instanz anhand konkreter Datenwerte vorgenommen wird. So ist beispielsweise anhand eines konkreten Kontostands eines Kunden in einem Bankprozess eine Bewertung vorzunehmen, wie im Prozess weiter verfahren werden soll. Diese Bewertung mittels situationsabhängiger Daten stellt ein orthogonales Konzept zur statischen semantischen Auswertung dar. Beide Aspekte der Semantik ergänzen sich zu einem vollständigeren Gesamtbild verschiedener Semantikeigenschaften.

Ein weiteres Thema mit Bezug zu den Konzepten zur Integration von Semantik in PMS, ist die maschinennahe Beschreibung der Semantikinformationen in einem Repräsentationsformalismus. Dazu existiert eine Vielzahl von Ansätzen im Kontext des Semantic Web [BDFS04]. Beispielsweise kämpfen Suchmaschinenbetreiber im WWW mit dem Problem,

wie die Relevanz von Suchtreffern für eine konkrete Anfrage verbessert werden kann. Dazu wurden Mechanismen entwickelt, um Ressourcen wie Webseiten semantisch annotieren zu können. Ansätze dazu finden sich in Beschreibungssprachen wie RDF [BG04] und OWL [MH04], die eine flexible Beschreibung von Ressourcen bieten. Aufbauend auf den Grundkonzepten dieser Beschreibungsformalismen kann für die Erfassung von Semantik in PMS eine Repräsentationssprache mit eigenem, der PMS Thematik angepasstem Vokabular entwickelt werden.

8

Zusammenfassung und Ausblick

Diese Ausarbeitung hatte das Ziel, die Integration semantischer Aspekte in adaptiven PMS zu untersuchen, um zusätzlich zur syntaktischen Korrektheit auch die semantische Korrektheit von Prozessen überprüfen zu können. Die Einbeziehung der Semantik geht über die applikationsneutrale Betrachtung von Prozessen hinaus. Damit können adaptive PMS eine neue Ebene für die Korrektheit und Qualität der unterstützten Prozesse erreichen.

8.1. Zusammenfassung der Ergebnisse

Die Auswertung der semantischen Korrektheit von Prozessen erfordert, dass diese semantisch erfasst werden. Um die Auswirkungen von Änderungen auf die semantische Korrektheit feststellen zu können, ist die Beschreibung der Semantik auf der Ebene von Aktivitäten durchzuführen. Die semantische Aktivitätenbeschreibung lehnt sich an die Satzbestandteile der natürlichen Sprache an. Subjekt, Prädikat, Akkusativ- und Dativ-Objekt finden als Subjekt, Operation, Objekt und Target Entsprechungen in der Aktivitätenbeschreibung. Den Objekten kommt eine Sonderrolle zu. Sie dienen nicht nur der Beschreibung der Aktivitäten, sondern setzen die Aktivitäten eines Prozesses in Beziehung zu anderen Aktivitäten. Dazu werden semantische Beziehungen zwischen den Objekten etabliert, die in Form von Abhängigkeits- und Ausschlussregeln formuliert werden. Diese Regeln bilden die Grundlage für das semantische Korrektheitskriterium. Wird eine Abhängigkeits- oder eine Ausschlussregel in einem Prozess verletzt, so existiert ein semantischer Konflikt und der untersuchte Prozess ist semantisch nicht korrekt.

Eine Abhängigkeitsregel fordert das Vorhandensein eines bestimmten Zielobjekts im Prozess. Andernfalls wird die Abhängigkeit nicht erfüllt und die Abhängigkeitsregel wird verletzt. Im Gegensatz dazu drückt eine Ausschlussregel aus, dass die festgelegten Quell- und Zielobjekte nicht zusammen in einem Prozess vorhanden sein dürfen. Ist dies der Fall, wird die Ausschlussregel verletzt.

Die Einträge der Aktivitätenbeschreibung für Operation, Subjekt und Target dienen dazu, das Objekt eines Eintrags in einen spezifischen Kontext zu setzen. Damit ist es möglich, semantische Beziehungen eines zugewiesenen Objekts für verschiedene Anwendungssituationen festzulegen. So können beispielsweise unterschiedliche semantische Beziehungen für ein Objekt Gültigkeit besitzen, je nachdem, mit welcher Operation das Objekt verwendet wird.

Die Objekte, Subjekte, Operationen und Targets, die innerhalb einer Domäne festgelegt sind, werden innerhalb getrennter Ontologien organisiert. Es gibt also eine Objekt-, eine Subjekt-, eine Operation- und eine Target-Ontologie. Zwischen den Elementen einer Ontologie bestehen Typbeziehungen. Damit können semantische Beziehungen nicht nur durch das definierte Zielobjekt erfüllt werden, sondern auch durch Vorkommen von Untertypen. Innerhalb der Ontologien wird Vererbung unterstützt. Das bedeutet, dass ein Untertyp die semantischen Beziehungen seiner Obertypen erbt. Damit werden Redundanzen in der Festlegung der Semantikregeln vermieden. Untertypen können zusätzlich zu den geerbten Regeln weitere Regeln festlegen, so dass eine inkrementelle Spezialisierung von Untertypen ermöglicht wird.

Aufbauend auf den Grundlagen zur Sicherstellung der semantischen Korrektheit wurde untersucht, welche Besonderheiten bei Ad-hoc-Änderungen von Instanzen und bei Änderungen auf der Prozessschema-Ebene existieren. Werden Instanzen ad-hoc geändert, so werden die Änderungen an den semantischen Informationen gegenüber dem Prozessschema in einer Delta-Liste der Instanz festgehalten. Diese Delta-Liste stellt eine logische Optimierung gegenüber der Speicherung der vollständigen semantischen Information zu einer Instanz dar. Bei Ad-hoc-Änderungen kann mittels dieser Delta-Liste untersucht werden, ob die Änderungen verträglich zu einer Instanz mit individuellen Änderungen sind. Auch im Zuge einer Prozesstyp-Änderung und der anschließenden Prüfung auf Migrierbarkeit der Instanzen werden die in der Delta-Liste festgehaltenen Abweichungen in die Entscheidung miteinbezogen.

Die Konzepte zur Sicherstellung der semantischen Korrektheit sind unter Effizienzgesichtspunkten umsetzbar. Ein potentieller Leistungsengpass entsteht durch die Bestimmung von Typbeziehungen bei der Auswertung von Semantikregeln. Um diesen Engpass zu umgehen, werden Ontologie-Elementen Typkennungen zugewiesen, die eine Aussage über eine Typbeziehung zwischen zwei Ontologie-Elementen auf effiziente Weise zulassen. Ein weiterer Leistungsengpass kann durch die Auswertung der semantischen Informationen eines Prozesses im Zuge eines Korrektheitstests verursacht werden. Dies kann verhindert werden, indem die im Prozess vorhandenen Objekte und die Zielobjekte der Regeln des Prozesses zugriffseffizient organisiert werden. Balancierten binäre Bäume

wurden in der Ausarbeitung aufgrund ihrer zugriffseffizienten Eigenschaften zur Speicherung der semantischen Informationen eines Prozesses vorgeschlagen. Weiter wurde die Untersuchung der Speicherbelastung durch die semantischen Informationen untersucht. Diese bleiben mit den dargestellten Ansätzen zur Zugriffsoptimierung moderat und stehen somit einer praxistauglichen Umsetzung nicht im Wege.

8.2. Ausblick

Die Betrachtungen zur Semantik stützen sich auf das in Kapitel 2 eingeführte eingeschränkte Metamodell. In einem weiteren Schritt ist der Ausbau des zugrundeliegenden Metamodells notwendig, wobei untersucht werden muss, wie sich Schleifen und SYNC-Kanten auf die Semantikprüfung auswirken. Beispielsweise wurde in Kapitel 3.6 davon ausgegangen, dass keine Aussage über die Reihenfolge der Abarbeitung zweier Aktivitäten in unterschiedlichen Ausführungszweigen eines AND-Splits möglich ist. Dies kann sich unter Hinzunahme von SYNC-Kanten ändern. Im Zusammenhang mit LOOP-Kanten ist die zentrale Frage, wie sich Schleifen auf über `directionTarget`-Attribute der Regeln angegebene Richtungsbeziehungen auswirken.

Die in Kapitel 4.4 eingeführten Ontologien zur Unterscheidung verschiedener Anwendungssituationen berücksichtigen Objekte, Operationen, Subjekte und Targets zur Unterscheidung verschiedener Kontexte. Damit kann die Grundstruktur von Sätzen der natürlichen Sprache nachgebildet werden. Dieses Konzept kann leicht durch weitere Aspekte erweitert werden. Beispielsweise kann der Ort, an dem eine Aktivität durchgeführt wird, Einfluss auf die Semantik nehmen. Es ist zu untersuchen, ob es noch weitere Aspekte gibt, die eine Verfeinerung von Anwendungssituationen ermöglichen.

Semantische Beziehungen können im entwickelten Modell über eine Richtungsbeziehung zwischen dem Quell- und Zielobjekt der Regel detaillierter formuliert werden. Es ist zu untersuchen, ob weitere Aspekte wie Kardinalitätsvorgaben für das Auftreten eines Zielobjekts im Prozess eine sinnvolle Ergänzung der Regeln darstellen. Ein weiterer wichtiger Punkt der untersucht werden muss, ist die Einbeziehung von Zeitaspekten in die Semantik. So kann es für die Erfüllung einer semantischen Regel von Belang sein, wieviel Zeit zwischen der Ausführung zweier, durch ihre zugeordneten Objekte in Beziehung stehenden, Aktivitäten liegt. Dabei ist zu entscheiden, ob dieser eigentlich eher dynamische Aspekt in die statische Beschreibung und Überprüfung der Semantik miteinbezogen werden kann.

In den Ausführungen von Kapitel 3.6 wurden einfache AND-/XOR-Elemente berücksichtigt. Es ist zu untersuchen, wie spezialisiertere Varianten, wie z.B. parallele Verzweigung mit finaler Auswahl, auf die Semantikprüfung einwirken.

Die globalen Ontologien im vorgestellten Ontologien-Konzept umfassen im besten Fall die semantische Beschreibung einer vollständigen Domäne. Nichtsdestotrotz kann es zu

Situationen kommen, in denen ein Prozess abweichende semantische Informationen benötigt. Um auch diese Flexibilität zu ermöglichen ist es vorstellbar, dass über prozesslokale Einstellungen die Angaben aus den Ontologien und den Aktivitätenbeschreibungen der verwendeten Aktivitäten überschrieben werden. So könnte für einen Prozess z.B. festgelegt sein, dass die Subjekt-Einträge sämtlicher Aktivitätenbeschreibungen auf ein bestimmtes Subjekt gesetzt werden. Es ist zu untersuchen, welche prozesslokalen Einstellungen wünschenswert sind und wie diese umgesetzt werden können.

Wie anhand der hier aufgeführten Punkte zu erkennen ist, ist das Gebiet der Semantik mit dieser Ausarbeitung noch nicht umfassend behandelt. Viele weitere interessante Aspekte sind zu durchleuchten, um umfassende semantische Korrektheitsgarantien für Prozesse zu verwirklichen.

Literaturverzeichnis

[AA01] Antoniou, G. ; Arief, M.: Executable Declarative Business Rules and Their Use in Electronic Commerce. In: *Int'l Journal of Intelligent Systems in Accounting, Finance & Management* 10 (2001), Nr. 4, S. 211–223

[Aal01] Aalst, W.M.P. van d.: Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change. In: *Information Systems Frontiers* 3(3) (2001), S. 297–317

[AB02] Aalst, W.M.P. van d. ; Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. In: *Theoretical Computer Science* 270 (2002), S. 125–203

[ABH00] Abecker, A. ; Bernardi, A. ; Hinkelmann, K. ; Kühn, O. ; Sintek, M.: Context-Aware, Proactive Delivery of Task-Specific Knowledge: The KnowMore Project. In: *Information Systems Frontier* 2 (2000), Nr. (3/4), S. 39–162

[ABN01] Abecker, A. ; Bernardi, A. ; Ntioudis, S. ; Mentzas, G. ; Herterich, R. ; Houy, C. ; Müller, S. ; Legal, M.: The DECOR Toolbox for Workflow-Embedded Organizational Memory Access. In: *Proc. Int'l Conference on Enterprise Information Systems (ICEIS'01)* (2001)

[AJ00] Aalst, W.M.P. van d. ; Jablonski, S.: Dealing with workflow change: Identification of issues and solutions. In: *Int'l Journal of Computer Systems Science & Engineering* 15 (2000), Nr. 5, S. 267–276

[BDFS04] Bussler, C. ; Davies, J. ; Fensel, D. ; Studer, R.: *The Semantic Web: Research and Applications*. Springer Verlag, 2004

[BG04] Brickley, D. ; Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema - W3C Recommendation 10 February 2004. In: *[Internet]* (2004). – URL: http://www.w3.org/TR/rdf-schema/

[BK05] Bajec, M. ; Krisper, M.: A methodology and tool support for managing business rules in organisations. In: *Information Systems 30* (2005), S. 423–443

[Caw03] Cawsey, Alison: *Künstliche Intelligenz Im Klartext*. Pearson Studium, 2003

[CCPP98] Casati, F. ; Ceri, S. ; Pernici, B. ; Pozzi, G.: Workflow Evolution. In: *Data & Knowledge Engineering* 24 (1998), S. 211–238

[DHH01] Degen, W. ; Heller, B. ; Herre, H.: Contributions to the Ontological Foundation of Knowledge Modelling / Institut für Informatik, Universität Leipzig. 2001 (02). – Technical Report

[EAB03] Elst, L. ; Aschoff, F.R. ; Bernardi, A. ; Maus, H. ; Schwarz, S.: Weakly-structured Workflows for Knowledge-intensive Tasks: An Experimental Evaluation. In: *Proc. Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'03)*, 2003, S. 340–345

[GN87] Genesereth, M.R. ; Nilsson, N.J.: *Logical Foundations of Artificial Intelligence*. San Mateo, CA : Morgan Kaufmann Publishers, 1987

[Goe02] Goesmann, Thomas: *Ein Ansatz zur Unterstützung wissensintensiver Prozesse durch Workflow-Management-Systeme*, Fakultät IV - Elektrotechnik und Informatik der Technischen Universität Berlin, Diss., 2002

[Gru93] Gruber, T. R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing / Knowledge Systems Laboratory, Stanford University. 1993 (KSL 93-04). – Technical Report

[Hei01] Heisig, Peter: Business Process Oriented Knowledge Management - Methode zur Verknüpfung von Wissensmanagement und Geschäftsprozessgestaltung. In: *Proc. 1. Konferenz Professionelles Wissensmanagement (WM 2001)* (2001)

[KSL02] Kim, J. ; Suh, W. ; Lee, H.: Document-based workflow modeling: a case-based reasoning approach. In: *Expert Systems with Applications* (2002), Nr. 23, S. 77–93

[MH04] McGuinness, D. L. ; Harmelen, F. van: R OWL Web Ontology Language Overview - W3C Recommendation 10 February 2004. In: *[Internet]* (2004). – URL: http://www.w3.org/TR/owl-features/

[MRW92] Müller, T. ; Rost, H.-P. ; Wolf, D.: *Das grosse Mathematik Buch*. Naumann & Göbel, 1992

[MZ03] Madhusudan, T. ; Zhao, J.: A case-based framework for workflow model management. In: *Proc. Int'l Conf. Business Process Management (BPM'03)* (2003), S. 354–369

[RD98] Reichert, M. ; Dadam, P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. In: *Journal of Intelligent Information Systems* 10 (1998), March/April, Nr. 2, S. 93–129

[RDB03] Reichert, M. ; Dadam, P. ; Bauer, T.: Dealing With Forward and Backward Jumps in Workflow Management Systems. In: *Int'l Journal Software and Systems Modeling (SoSyM)* 2 (2003), Nr. 1, S. 37–58

[Rei00] Reichert, Manfred: *Dynamische Ablaufänderungen in Workflow-Management-Systemen*, Universität Ulm, Fakultät für Informatik, Diss., 2000

[Rin04] Rinderle, Stefanie: *Schema Evolution in Process Management Systems*, Universität Ulm, Fakultät für Informatik, Diss., 2004

[RP02] Rechenberg, P. ; Pomberger, G.: *Informatik-Handbuch*. 3. Auflage. Hanser, 2002

[RRD02] Rinderle, S. ; Reichert, M. ; Dadam, P.: Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata. In: *Informatik - Forschung und Entwicklung* 17 (2002), S. 177–197

[RRD04a] Rinderle, S. ; Reichert, M. ; Dadam, P.: Correctness Criteria for Dynamic Changes in Workflow Systems - A Survey. In: *Data and Knowledge Engineering* 50(1) (2004), S. 9–34

[RRD04c] Rinderle, S. ; Reichert, M. ; Dadam, P.: Disjoint And Overlapping Process Changes: Challenges, Solutions, Applications. In: *Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'04)* (2004), Nr. LNCS 3290, S. 101–121

[RRD04] Rinderle, S. ; Reichert, M. ; Dadam, P.: Flexible Support of Team Processes by Adaptive Workflow Systems. In: *Distributed and Parallel Databases - An International Journal* 16(1) (2004), S. 91–116

[RRD04b] Rinderle, S. ; Reichert, M. ; Dadam, P.: On Dealing With Structural Conflicts Between Process Type and Instance Changes. In: *Proc. 2nd Int'l Conf. Business Process Management (BPM'04)* (2004), Nr. LNCS 3080, S. 274–289

[RWRW05] Rinderle, S. ; Weber, B. ; Reichert, M. ; Wild, W.: Integrating Process Learning and Process Evolution - A Semantics Based Approach. In: *Proc. 3rd Int'l Conf. Business Process Management (BPM'05)* (2005), Nr. LNCS 3649, S. 252–267

[Sch97] Schöning, U.: *Vorlesungsskript - Praktische Informatik I*. 7. Auflage. Universität Ulm - Abt. Theoretische Informatik, 1997

[Ste99] Steyer, Ralph: *Java2 - Referenz und Anwendungen*. Markt&Technik, 1999

[Wes01] Weske, M.: Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In: *Proc. 34th Annual Hawaii International Conference on System Sciences (HICSS-34)* (2001)

[Wik05] Wikipedia: Ontologie - Wikipedia. In: *[Internet]* (30.09.2005). – URL: http://de.wikipedia.org/wiki/Ontologie

[WM97] Wilhelm, R. ; Maurer, D.: *Übersetzerbau*. 2. Auflage. Springer, 1997

[WRWR05] Weber, B. ; Rinderle, S. ; Wild, W. ; Reichert, M.: CCBR-Driven Business Process Evolution. In: *Proc. 6th Int'l Conf. on Case-Based Reasoning (ICCBR'05)* (2005), Nr. LNCS 3620, S. 610–624

A
Definitionen

Definition A.1 (Vorgänger- und Nachfolger-Funktionen)

In der Ausarbeitung werden folgende Funktionen zur Bestimmung von Vorgänger- und Nachfolgerbeziehungen zwischen Aktivitäten aus [Rin04, S. 228] verwendet.

c_succ(S,n)/c_pred(S,n)	die Menge der direkten Nachfolger/Vorgänger der Aktivität n
c_succ*(S,n)/c_pred*(S,n)	die Menge aller direkten und indirekten Nachfolger/Vorgänger der Aktivität n

Definition A.2 (Objekt-Ontologie)

Gegeben sei eine Domäne $\mathcal{D} = (\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS)$*. Eine Objekt-Ontologie OBJ_ONT stellt sich als Tupel* $(\mathcal{O}, R, \mathcal{R}\_\mathcal{OBJ}, K, ASSIGN\_K)$ *dar, mit*

- $\mathcal{O}$ *entspricht der Menge der Objekte der Domäne* $\mathcal{D}$
- $R := \{(s,t)|s,t \in \mathcal{O}\}$ *gibt die Menge der Typbeziehungen an*
- $\mathcal{R}\_\mathcal{OBJ} \subseteq \mathcal{R}$ *ist die Menge der semantischen Beziehungen der Objekt-Ontologie*
- $K := (\{relatedOp\} \times \mathcal{OP}) \cup (\{relatedSubj\} \times \mathcal{S}) \cup (\{relatedTar\} \times \mathcal{T}))$ *stellt die Menge der Kontextattribute für die Einschränkung semantischer Beziehungen auf bestimmte Anwendungssituationen dar*

- *ASSIGN_K:* $\mathcal{R}\_\mathcal{OBJ} \rightarrow \mathcal{P}(K)$ *ordnet den semantischen Beziehungen den Quell- und Zielobjekt-Kontext zu*

Definition A.3 (Operation-Ontologie)

Gegeben sei eine Domäne $\mathcal{D} = (\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS)$*. Eine Operation-Ontologie OP_ONT stellt sich als Tupel* $(\mathcal{OP}, R, \mathcal{R}\_\mathcal{OP}, K, ASSIGN\_K)$ *dar, mit*

- $\mathcal{OP}$ *entspricht der Menge der Operationen der Domäne* $\mathcal{D}$
- $R := \{(s,t)|s,t \in \mathcal{OP}\}$ *gibt die Menge der Typbeziehungen an*
- $\mathcal{R}\_\mathcal{OP} \subseteq \mathcal{R}$ *ist die Menge der semantischen Beziehungen der Operation-Ontologie*
- $K := (\{domain\} \times \mathcal{O}) \cup (\{relatedOp\} \times \mathcal{OP}) \cup (\{relatedSubj\} \times \mathcal{S}) \cup (\{relatedTar\} \times \mathcal{T}))$ *stellt die Menge der Kontextattribute für die Einschränkung semantischer Beziehungen auf bestimmte Anwendungssituationen dar*
- *ASSIGN_K:* $\mathcal{R}\_\mathcal{OP} \rightarrow \mathcal{P}(K)$ *ordnet den semantischen Beziehungen den Quell- und Zielobjekt-Kontext zu*

Definition A.4 (Subjekt-Ontologie)

Gegeben sei eine Domäne $\mathcal{D} = (\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS)$*. Eine Subjekt-Ontologie SUBJ_ONT stellt sich als Tupel* $(\mathcal{S}, R, \mathcal{R}\_\mathcal{SUBJ}, K, ASSIGN\_K)$ *dar, mit*

- $\mathcal{S}$ *entspricht der Menge der Subjekte der Domäne* $\mathcal{D}$
- $R := \{(s,t)|s,t \in \mathcal{S}\}$ *gibt die Menge der Typbeziehungen an*
- $\mathcal{R}\_\mathcal{SUBJ} \subseteq \mathcal{R}$ *ist die Menge der semantischen Beziehungen der Subjekt-Ontologie*
- $K := (\{domain\} \times \mathcal{OP}) \cup (\{objContext\} \times \mathcal{O}) \cup (\{relatedOp\} \times \mathcal{OP}) \cup (\{relatedSubj\} \times \mathcal{S}) \cup (\{relatedTar\} \times \mathcal{T}))$ *stellt die Menge der Kontextattribute für die Einschränkung semantischer Beziehungen auf bestimmte Anwendungssituationen dar*
- *ASSIGN_K:* $\mathcal{R}\_\mathcal{SUBJ} \rightarrow \mathcal{P}(K)$ *ordnet den semantischen Beziehungen den Quell- und Zielobjekt-Kontext zu*

Definition A.5 (Target-Ontologie)

Gegeben sei eine Domäne $\mathcal{D} = (\mathcal{N}, \mathcal{O}, \mathcal{OP}, \mathcal{S}, \mathcal{T}, \mathcal{R}, SEM\_ACT, SEM\_RELS)$*. Eine Target-Ontologie TAR_ONT stellt sich als Tupel* $(\mathcal{S}, R, \mathcal{R}\_\mathcal{TAR}, K, ASSIGN\_K)$ *dar, mit*

- $\mathcal{T}$ *entspricht der Menge der Targets der Domäne* $\mathcal{D}$
- $R := \{(s,t)|s,t \in \mathcal{T}\}$ *gibt die Menge der Typbeziehungen an*
- $\mathcal{R}\_\mathcal{TAR} \subseteq \mathcal{R}$ *ist die Menge der semantischen Beziehungen der Target-Ontologie*

- $K := (\{domain\} \times \mathcal{OP}) \cup (\{objContext\} \times \mathcal{O}) \cup (\{relatedOp\} \times \mathcal{OP}) \cup (\{relatedSubj\} \times \mathcal{S}) \cup (\{relatedTar\} \times \mathcal{T}))$ *stellt die Menge der Kontextattribute für die Einschränkung semantischer Beziehungen auf bestimmte Anwendungssituationen dar*
- *ASSIGN_K:* $\mathcal{R\_TAR} \rightarrow \mathcal{P}(K)$ *ordnet den semantischen Beziehungen den Quell- und Zielobjekt-Kontext zu*

B

Abkürzungen

CCBR Conversational Case-Based Reasoning
CBR Case-Based Reasoning
KI künstliche Intelligenz
OO Objektorientierung
OOP objektorientierte Programmierung
OWL Web Ontology Language
PMS Prozess-Management-System
RDF Ressource Description Framework
WWW World Wide Web

Erklärung

Michael Nahler, Matrikel-Nr.: 428243

Ich erkläre, dass ich die Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Ulm, den 31. Oktober 2005