Process Mining

Bestehende Ansätze und weiterführende Aspekte

Diplomarbeit

Diplomand:	Linh Thao Ly
Fachbereich:	Informatik
Fachrichtung:	Medieninformatik
Betreuer:	Dr. Manfred Reichert, Dr. Stefanie Rinderle
Zweitkorrektor:	Prof. Dr. Peter Dadam
Abgabedatum:	2. Mai 2005

Danksagung

An erster Stelle danke ich Stefanie Rinderle und Manfred Reichert dafür, dass sie die sicherlich nicht einfache Aufgabe der Betreuung meiner Diplomarbeit übernommen haben. Auch für die Zusammenarbeit zur raschen Fertigstellung des Papers bin ich ihnen sehr dankbar.
Meinen Eltern, Thi Lam Huong Pham und Hoang Khoi Ly, danke ich für ihre Unterstützung während meiner ganzen Studienzeit.
Mein besonderer Dank geht an meinen Freund, Kevin Göser, der mir immer zur Seite stand.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Zielsetzung

Seit den 80er Jahren besteht bei der Organisation der Unternehmensstruktur ein Trend weg von Funktionsorientierung hin zur Prozessorientierung. Im Zuge dieser Entwicklung, die aktuelle Schlagwörter wie BPR (*Business Process Reengineering*) und BPM (*Business Process Management*) hervorbrachte, sind auch viele Anwendungen entstanden, die die prozessorientierte Sicht unterstützen. Insbesondere die Workflow-Technologie hat in den letzten Jahren eine enorme Entwicklung durchgemacht. Insbesondere für den Einsatz von Workflow-Management-Systemen ist das Vorliegen eines Prozessmodells notwendig.

Allerdings liegt das Prozesswissen in Unternehmen oftmals nicht explizit vor, etwa als graphisches Prozessmodell, sondern steckt in Form von Lokalwissen indirekt in den Köpfen der beteiligten Mitarbeiter. Diese wissen aus ihrem Arbeitsalltag heraus, wie Aufgaben aus ihrem jeweiligen Ressort abzulaufen haben. Das Wissen über den gesamten globalen Prozess fehlt jedoch in den meisten Fällen.
Gerade weil Prozesse immer komplexer werden und mehrere Funktionseinheiten oder gar mehrere Unternehmen involviert sein können, ist die Explikation des Prozesswissens eine aufwendige und teure Angelegenheit. Mit *Process Mining* werden Methoden bereitgestellt, um Prozesswissen aus Log-Daten vergangener Prozessausführungen zu extrahieren.

Erstmals ist mit *Process Mining* eine kostengünstige und objektive Alternative zu traditionellen Techniken der Wissensakquisition verfügbar. Im Rahmen dieser Arbeit wird ein Überblick in dieses neue Thema gegeben.

Die vorliegende Arbeit verfolgt zweierlei Ziele. Zum einen soll mit dieser Arbeit ein umfassender Überblick über bestehende *Process-Mining*-Methoden gegeben werden. Im Unterschied zu anderen Arbeiten [86, 87], die nur eine Übersicht über einige Ansätze darstellen,

beschäftigen wir uns mit allen bestehenden Ansätzen[1]. Eine vergleichbare Arbeit gibt es daher nicht. Weitere Arbeiten stellen eher einzelne Ansätze vor, als dass sie einen vergleichenden Überblick über bestehende Lösungsansätze bieten. Insbesondere bieten wir einen systematischen Zugang zu *Control Flow Mining*, der wichtigsten Teilfragestellung von *Process Mining*.

Zum anderen verfolgt die vorliegende Arbeit das Ziel, die bisher von *Process Mining* vernachlässigten organisatorischen Aspekte stärker zu berücksichtigen. Wir führen im Rahmen dieser Arbeit die Fragestellung ein, Bearbeiterzuordnungsregeln aus Verlaufsdaten abzuleiten, und erarbeiten eine Lösung für dieses Problem.

1.2 Gliederung der Arbeit

Nachdem in Kapitel 2 wichtige Begriffe und relevante Grundlagen erläutert werden, gehen wir in Kapitel 3 auf das Thema *Process Mining* ein. In Kapitel 4 wird auf die Verlaufsdaten eingegangen. Es werden damit wichtige Grundlagen für das darauf folgende Kapitel 5 gegeben. Kapitel 6 gibt einen Überblick über Ansätze zu organisatorischen Aspekten. In Kapitel 7 stellen wir unseren eigenen Ansatz zur Ableitung von Bearbeiterzuordnungsregeln aus Verlaufsdaten vor.

[1] In der Endphase dieser Arbeit wurde eine neue Arbeit gefunden, die sich mit Workflow Mining beschäftigt. Diese ist jedoch noch nicht publiziert und konnte aus Zeitgründen nicht mehr integriert werden. Der Vollständigkeit wegen verweisen wir an dieser Stelle auf die Arbeit in [79]

Kapitel 2

Grundlagen

In diesem Kapitel werden für das Verständnis der Arbeit relevante Grundlagen umrissen. Der erste Teil geht kurz auf das Thema Workflow-Management ein. Im Zweiten Abschnitt befassen wir uns mit der Ablauforganisation. Der dritte Teil behandelt die Aufbauorganisation und die Zuordnung von Bearbeitern. Im Letzten Teil wird auf die graphische Darstellung von Prozessen eingegangen.

2.1 Geschäftsprozesse und Workflow-Management-Systeme

Ein Workflow ist ein Ablauf, im allgemeinen wird dieser Begriff im Zusammenhang mit Geschäftsprozessen verwendet. Zur computergestützten Ausführung der Arbeitsabläufe werden Workflow-Management-Systeme (WfMS) eingesetzt.
Ein WfMS muss alle Vorgänge rund um Arbeitsabläufe unterstützen. Dazu gehört die Modellierung von Abläufen, deren Ausführung, das Ansteuern von aufzurufenden Anwendungen, die Verteilung von Arbeitslisteneinträgen an Bearbeiter, die Administration sowie die Überwachung der Prozessausführung [45, 58].

Um die Ausführung eines Prozesses in einem WfMS zu ermöglichen, benötigen die meisten Workflow-Management-Systeme ein graphisch modelliertes Prozessmodell, auch Workflow-Schema genannt. Die Modellierung ist vom verwendeten WfMS abhängig. In dieser Arbeit verwenden wir hauptsächlich Konstrukte, wie sie in Abbildung 2.1 an dem Beispielprozessmodell dargestellt sind.

Eine Aktivitätszuordnungsfunktion bildet zwischen den Knoten des Prozessmodells und einer Menge von Aktivitäten ab. So wird Knoten 1 beispielsweise auf die Aktivität a abgebildet. Der Übersichtlichkeit wegen, werden wir in zukünftigen Abbildungen auf die Zuordnungsfunktion verzichten und Aktivitäten direkt den Knoten des Prozessmodells zuordnen. Knoten 2 stellt in der verwendeten Prozessmodellierungssprache einen OR-Split dar, Knoten 6 einen

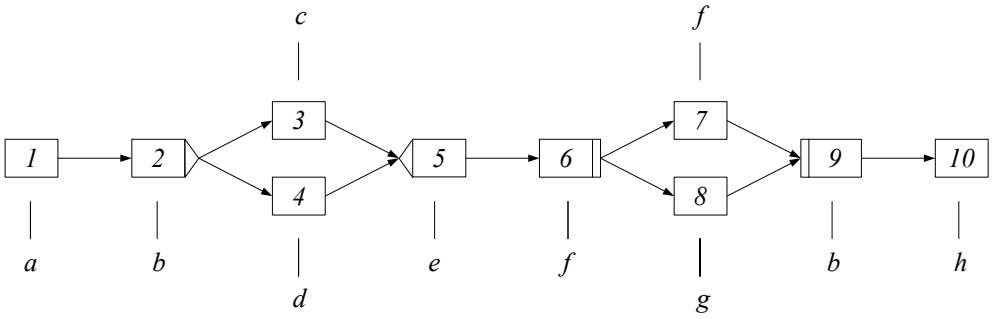

Abbildung 2.1: Ein Prozessmodell in der verwendeten Syntax

AND-Split. Knoten 5 und Knoten 9 sind jeweils ein OR- und ein AND-Join. Die von uns verwendete Syntax ist an die Syntax von ADEPT [70], dem Workflow-Management-System der Abteilung Datenbanken und Informationssysteme der Universität Ulm, angelehnt.

Neben der oben erwähnten Darstellung werden wir, da die in dieser Arbeit vorgestellten Ansätze es erfordern, auch gerichtete Graphen sowie Petri-Netze [88] einsetzen.

2.2 Aufbauorganisation und Bearbeiterzuordnung

Um einen effizienten und reibungslosen Ablauf zu ermöglichen, ist es wichtig, dass Arbeitsschritte sinnvoll an Bearbeiter verteilt werden. Hierfür können unter anderem organisatorische Einheiten, Rollen und Fähigkeiten eine Rolle spielen. Ein Arbeitsschritt kann allerlei Anforderungen an die Qualifikation seines Bearbeiters stellen. Damit das WfMS die Aufgaben angemessen verteilen kann, müssen an den Knoten eines Prozesses komplexe Bearbeiterzuordnungsregeln hinterlegt werden können, welche die Qualifikationen angemessen beschreiben. Wird der entsprechende Knoten bei der Prozessausführung aktiviert, werden die hinterlegten Zuordnungsregeln aufgelöst (*Role Resolution*). Die Menge der entsprechenden Bearbeiter wird identifiziert. Arbeitslisteneinträge können dann erstellt und in die Arbeitslisten dieser Mitarbeiter gelegt werden.
Wichtig ist hierbei die Unterscheidung zwischen den Bearbeiterzuordnungsregeln und dem Organisationsmodell, in dem die Qualifikationen der Mitarbeiter modelliert sind. Die Verwaltung des Organisationsmodells sowie die Auflösung der Bearbeiterzuordnungsregeln kann auch in einem externen System realisiert werden [9, 58].

Kapitel 3

Process Mining

3.1 Überblick

Unter dem Begriff *Process Mining* werden Methoden und Techniken verstanden, die Prozesswissen aus Verlaufsdaten vergangener Prozessausführungen extrahieren. Im Workflow-Kontext wird oft auch der Begriff *Workflow Mining* verwendet. Dieser meint insbesondere die Ableitung von Workflow-Modellen aus Verlaufsdaten.

Process Mining ist ein sehr junges Thema. Die Anfänge gehen zurück auf Arbeiten von Cook und Wolf [15, 18, 19], die sich mit der Ableitung von Prozessmodellen im Kontext von Softwareprozessen aus ereignisbasierten Daten beschäftigten. Diese Arbeit fassten sie unter dem Begriff *Process Discovery* zusammen.

Heute ist *Process Mining* ein hochaktuelles Thema. Viele Forschungsarbeiten wurden dazu veröffentlicht, die meisten in jüngster Zeit. Dabei kann zwischen zwei Sichten unterschieden werden [82]:

- Prozesssicht
- Organisationssicht

Die Prozesssicht ist sicherlich eines der wichtigsten Aspekte von *Process Mining* und konzentriert sich auf die Ableitung des Kontrollflusses von Prozessen. Diese Fragestellung ist auch unter dem Begriff *Control Flow Mining* bekannt. Die meisten Publikationen zu *Process Mining* widmen sich diesem Thema.
Die Organisationssicht hingegen konzentriert sich auf die Ableitung organisatorischer Aspekte.

3.2 Anwendungen von Process Mining

Es gibt zahlreiche Gebiete, in denen Process-Mining-Techniken sinnvoll eingesetzt werden können. Schimm nennt dazu in [76] unter anderem Wissensmanagement, wo Process Mining eingesetzt werden kann, um Prozesswissen zu erfassen, sowie die Erstellung von Wissensbasen für Existenzsysteme. In der Literatur wird *Process Mining* vor allem hinsichtlich der Einsatzmöglichkeit im Kontext von Workflow-Management-Systemen betrachtet [86, 42].

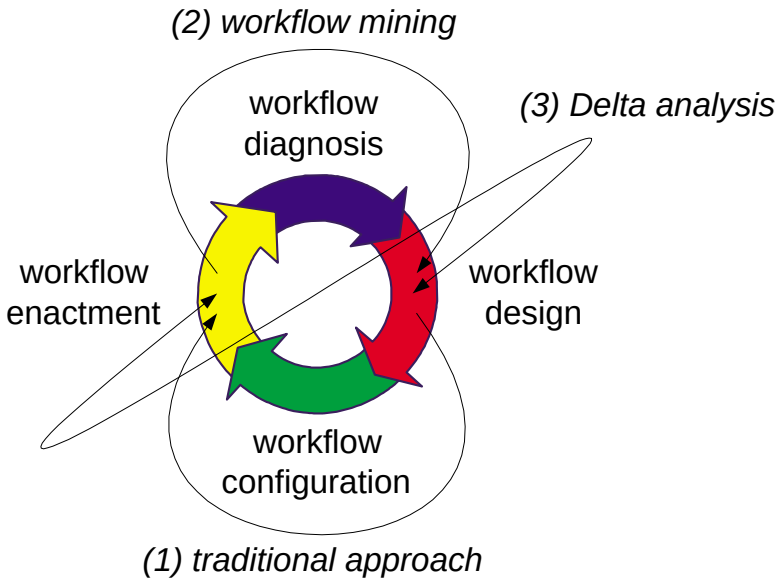

Abbildung 3.1: Workflow-Life-Cycle

Die Akquisation von Prozesswissen in der Design-Phase ist ein aufwendiges Unterfangen. Grund dafür ist vor allem, dass Prozesswissen insbesondere in größeren Unternehmen in der Regel auf viele Beteiligte verteilt ist, die jeweils über lokales Prozesswissen über ihr Ressort verfügen. Die Aufgabe der Modellierung von Prozessvorlagen wird daher vornehmlich von Prozessexperten bzw. -designern übernommen. Unter Verwendung von bekannten Techniken, z.B. Interviews, versuchen Prozessdesigner, lokales Prozesswissen zu sammeln und zu einem globalen Prozesswissen zu aggregieren [58]. Dennoch stellt die Akquisition von Prozesswissen auch für den Fachmann eine große Herausforderung dar, die mit vielen Hindernissen verbunden ist. Aussagen von Beteiligten sind stets subjektiv und nicht immer sind Beteiligte auch wirklich daran interessiert, ihr Wissen öffentlich zugänglich zu machen [42]. Aus seiner Erfahrung mit Workflow-Projekten bei DaimlerChrysler schätzt Herbst den Aufwand für die Akquisition und Validierung von Prozesswissen bei der Realisierung von Workflow-Management-Anwendungen auf 60%[42]. An diesem Punkt kann Process Mining ansetzen. Modelle, die mit Workflow-Mining-Techniken gewonnen werden, sind, anders als Modelle,

die mittels herkömmlichen Techniken ermittelt werden, objektiv und spiegeln so den Prozess wieder, wie er wirklich ausgeführt wurde [86].

Einen sinnvollen Einsatz kann *Process Mining* auch in der Diagnose-Phase finden. Bei der Prozessmodellierung lassen sich nicht immer alle möglichen Ausnahmen von vornherein modellieren [72]. Moderne Workflow-Management-Systeme, wie z.B. ADEPT [73, 71], erlauben daher auch vom Workflow-Schema abweichende Prozessausführungen. Nicht zuletzt deshalb spielt die Diagnose-Phase eine große Rolle.
Häufige Abweichungen vom Schema stellt die Angemessenheit des Schemas in Frage. Eine rasche und stetige Anpassung des Prozesses an neue Anforderungen ist u.U. geschäftskritisch [52]. *Process Mining* kann hierbei für die Adaption des Prozessmodells verwendet werden. Insbesondere die Delta-Analyse ist in diesem Kontext zu nennen [82]. Bei der Delta-Analyse wird das vorliegende Workflow-Schema mit dem abgeleiteten Prozessmodell verglichen. Dadurch können Diskrepanzen zwischen dem Ist-Zustand und dem Soll-Zustand der Prozessausführung aufgedeckt werden. Da *Process Mining* als ein Tool zur Verbesserung der Prozessqualität durch Ableitung von Prozesswissen betrachtet werden kann, wird *Process Mining* daher oft im Zuge mit aktuellen Schlagwörter, wie *Business Process Intelligence*, *Business Process Reengineering* und *Business Process Management*, genannt [23, 21, 90, 13].

Kapitel 4

Verlaufsdaten als Input für Process-Mining-Methoden

4.1 Einleitung

Verlaufsdaten, auch *Audit Trail*, *History Data* oder *Audit Data* genannt, stellen die Grundlage für Process-Mining-Methoden dar. In diesem Kapitel gehen wir daher genauer sie ein. Zunächst führen wir die Grundbegriffe Ereignis, Spur und Log in Abschnitt 4.2 ein. In Abschnitt 4.3 gehen wir auf die Akquisition und Aufbereitung von Verlaufsdaten ein und schließen das Kapitel mit einer Zusammenfassung in Abschnitt 4.4.

4.2 Grundbegriffe

4.2.1 Ereignisse

Ereignisse markieren Änderungen, z.B. Änderungen des Zustands einer Aktivität, die bei der Ausführung einer Prozessinstanz auftreten. Workflow-Management-Systeme protokollieren alle wichtigen Ereignisse bei der Ausführung von Prozessinstanzen. MQSeries Workflow [50, 49] beispielsweise, ein WfMS von IBM, verzeichnet alle Ereignisse, die mit der Statusänderung eines Knotens oder einer Prozessinstanz assoziiert sind.
Die Verwendung ereignisbasierter Daten zum Protokollieren ist allerdings nicht nur typisch für den Workflow-Kontext. Auch andere prozessorientierte Systeme, z.B. Systeme für *Enterprise Resource Planning* wie SAP oder Kollaborationssysteme wie Caramba, protokollieren Abläufe in dieser Form.

Tabelle 4.2.1 zeigt typische Verlaufsdaten, wie sie von vielen Workflow-Management-Systemen, z.B. Staffware Process Suite [81], in ähnlicher Form erzeugt werden. Jede Zeile

der Tabelle stellt ein Ereignis dar.

Instanznummer	*Ereignistyp*	*Aktivitat*	*Benutzer*	*Zeitstempel*
1	start	a	User12	15.01.2005 12:30
1	complete	a	User12	15.01.2005 15:00
1	start	b	User13	15.01.2005 17:30
1	start	c	User14	15.01.2005 18:00
1	complete	c	User14	15.01.2005 19:30
1	complete	b	User13	15.01.2005 20:00
2	start	a	User7	16.01.2005 12:30
2	complete	a	User7	16.01.2005 15:30
...	...	...	...	...

Tabelle 4.1: Ein Beispiel für Verlaufsdaten von zwei Prozessinstanzen

Wie auch in Tabelle 4.1 verdeutlicht, beinhalten Ereignisse typischerweise folgende Informationen:

- Eine Instanznummer, evt. auch in Kombination mit einer Prozessnummer, die eine eindeutige Zuordnung des Ereignisses zu einer Prozessinstanz erlaubt
- Einen Ereignistyp, z.B. Start, auf den wir später noch eingehen werden
- Eine Aktivität, die mit dem Ereignis assoziiert ist, z.B. eine Aktivität, die gestartet wurde
- Einen Zeitstempel
- Einen Benutzernamen, für den Benutzer, der mit dem Ereignis assoziiert wird, z.B. der Bearbeiter einer Aktivität

Es ist anzumerken, dass Ereignisse auch mit Prozessen oder internen Aspekten assoziiert sein können, z.B. ein Ereignis für den Start einer Prozessinstanz. Für bisherige *Process-Mining*-Ansätze sind vor allem Ereignisse interessant, die mit einer Aktivität assoziiert sind. Insbesondere für *Control Flow Mining*, worauf wir im Kapitel 5 eingehen, sind vor allem die ersten vier Informationen von Ereignissen, von Interesse.
Neben den aufgeführten Informationen können mit Ereignissen auch weitere Daten protokolliert werden, z.B. eine Knotennummer für den ausgeführten Knoten im Prozessmodell. Da dies für die in dieser Arbeit vorgestellten Ansätze jedoch keine Rolle spielt, gehen wir nicht weiter darauf ein.

Wie bereits erwähnt, haben Ereignisse einen Typ. In der Tabelle 4.1 werden beispielsweise nur die Ereignistypen *start* und *complete* verwendet. *start* ist das Startereignis und *complete* das Endereignis einer Aktivität.

Abbildung 4.1 zeigt einen Zustandsautomaten für ein allgemeines Ereignismodell. Die Er-

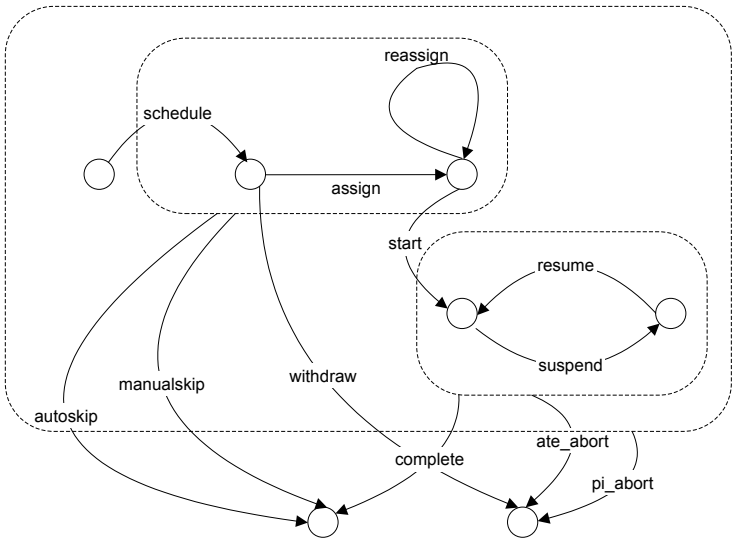

Abbildung 4.1: Zustandsautomat eines Ereignismodells nach van der Aalst et al.

eignisse in der Abbildung stellen Ereignistypen dar, wie sie so oder in ähnlicher Form, z.B. mit einer anderen Benennung, in gängigen Workflow-Management-Systemen implementiert sind. Das Ereignis *withdraw* bedeutet beispielsweise, dass eine bereits aktivierte Aktivität wieder zurückgenommen wird und entsprechende Arbeitslisteneinträge aus den Arbeitlisten entfernt werden. Damit überführt es die Aktivität, wie auch das Ereignis *complete*, in einen Endzustand.
Bisherige *Process-Mining*-Ansätze verwenden vor allem die bereits genannten Start- und Endereignisse von Aktivitäten.

Viele Workflow-Management-Systeme, wie z.B. MQ Workflow von IBM [50, 49], erlauben Optionen, um die Granularität der protokollierten Daten einzustellen. Dies erscheint angesichts der großen Mengen an Ereignisdaten, die bei Prozessausführungen anfallen können, sinnvoll. Auch für *Process Mining* Zwecke ist dies nicht verkehrt, da nicht alle möglicherweise auftretenden Ereignisse für *Process-Mining*-Methoden von Bedeutung sind.

Ereignisbasierte Daten können nicht nur für den Workflow-Kontext verwendet werden. Wolf und Rosenblum haben in [94] beispielsweise speziell auf Software-Prozesse zugeschnittene Ereignistypen definiert, die sie im Rahmen ihrer Untersuchung von Software-Prozessen verwendeten.

4.2.2 Spuren

Ereignisse können anhand ihrer Instanznummer und ihres Zeitstempels zu Ereignisspuren (*Event Traces*) zusammengefasst werden. Eine Ereignisspur stellt den sequentiellen Verlauf

einer Prozessinstanz dar. Da der Zeitstempel in der Regel sehr feingranular ist, kann praktisch ausgeschlossen werden, dass zwei Ereignisse denselben Zeitstempel tragen.
Die Ereignisspur für die Prozessinstanz 1 aus Tabelle 4.1 sieht wie folgt aus:

$$< a^+, a^-, b^+, c^+, c^-, b^- >$$

Mit a^+ und a^- werden wir für den restlichen Verlauf dieser Arbeit das Start- und das Endereignis einer Aktivität a bezeichnen.

Viele der in dieser Arbeit in Kapitel 5 vorgestellten Ansätze abstrahieren von konkreten Ereignissen und arbeiten auf Aktivitätsspuren (*Activity Traces*). Aus der obigen Spur erhalten wir z.B. die Aktivitätsspur $< a, c, b >$ für Instanz 1, indem nur die Endereignisse berücksichtigt werden. Man beachte, dass Aktivität c hier vor b vorkommt, da c vor b beendet wurde.

Auf Grundlage von Ereignis- und Aktivitätsspuren definieren wir einige Beziehungen, die im restlichen Verlauf dieser Arbeit verwendet werden. Die Semantik dieser Beziehungen ist leicht nachvollziehbar. Wir gehen dabei von konsistenten Spuren aus, also Spuren, in denen zu jedem Startereignis auch ein entsprechendes Endereignis existiert.

Definition 4.1 (Folgebeziehung bezüglich Ereignisspuren). *Eine Aktivität b folgt einer Aktivität a bezüglich einer Menge von Ereignisspuren S, wenn b in jeder Spur in S, in der a und b gemeinsam vorkommen, nach dem Endereignis von a gestartet wird.*

Definition 4.2 (Direkte Folgebeziehung bezüglich Ereignisspuren). *Eine Aktivität b folgt direkt einer Aktivität a bezüglich einer Menge von Ereignisspuren S, wenn b in jeder Spur in S, in der a und b gemeinsam vorkommen, nach dem Endereignis von a gestartet wird und keine andere Aktivität zwischen dem Ende von a und dem Start von b gestartet und beendet wurde.*

Definition 4.3 (Folgebeziehung bezüglich Aktivitätsspuren). *Eine Aktivität b folgt einer Aktivität a bezüglich einer Menge von Aktivitätsspuren A, wenn b in jeder Spur in A, in der a und b gemeinsam vorkommen, nach a auftritt.*

Definition 4.4 (Direkte Folgebeziehung bezüglich Aktivitätsspuren). *Eine Aktivität b folgt direkt einer Aktivität a bezüglich einer Menge von Aktivitätsspuren A, wenn b in jeder Spur in A, in der a und b gemeinsam vorkommen, direkt nach a auftritt.*

Die Verwendung von Aktivitätsspuren stellt einen Informationsverlust gegenüber der Verwendung von Ereignisspuren dar. Laut der Definitionen der direkten Folgebeziehung bezüglich Aktivitätsspuren folgt Aktivität b direkt auf Aktivität c. Nach der Definition der Folgebeziehung bezüglich Ereignisspuren würden b und c jedoch nicht miteinander in Beziehung stehen, da sie sich zeitlich überlappen.

4.2.3 Log

Ein Log fasst eine Menge von Spuren zusammen. Grundsätzlich kann ein Log Spuren verschiedener Prozesse enthalten. In dieser Arbeit werden wir uns jedoch auf ein Log als eine Menge von Spuren eines einzelnen Prozesses beziehen. Es ist klar, dass ein Log mit Spuren von unterschiedlichen Prozessen leicht in Logs aufgeteilt werden kann, die jeweils nur Spuren eines Prozesses enthalten.

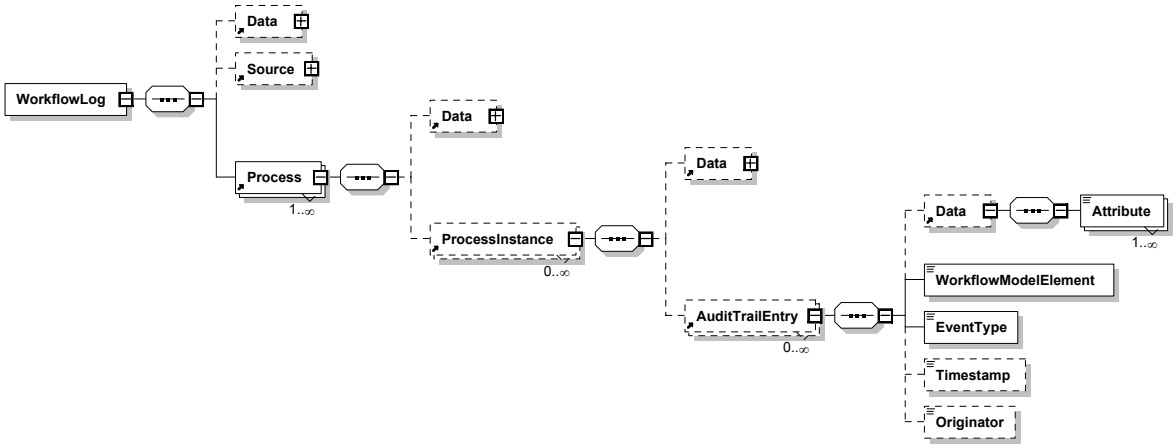

Abbildung 4.2: Die Struktur eines Logs in Form eines XML-Schemas aus [30] in XMLSpy (www.xmlspy.com)

Abbildung 4.2 zeigt den Aufbau eines XML-Schemas für Log-Daten. Dieses Format wird von einigen Anwendungen verwendet, die in dieser Arbeit vorgestellt werden, z.B. ProM (siehe Abschnitt 5.7.3). Das Log-Format enthält neben bereits genannten Elementen auch Konstrukte, in denen Zusatzdaten integriert werden können (*Data*).

4.3 Akquisition und Aufbereitung von Verlaufsdaten

Die Anwendbarkeit von *Process Mining* hängt davon ab, ob eine entsprechende Menge an verwertbaren ereignisbasierten Daten zur Verfügung steht. Qualität und Vollständigkeit der Verlaufsdaten sind dabei maßgeblich ausschlaggebend für die Qualität der Ergebnisse von *Process Mining*. Eine wichtige Frage ist daher, wie Verlaufsdaten gesammelt und für eine Weiterverwendung aufbereitet werden können. Insbesondere wenn Verlaufsdaten von verteilten Systemen benötigt werden, ist die Bereitstellung von verwertbaren Daten in einer entsprechenden Form keine einfache Aufgabe.

4.3.1 Beispiele in der Literatur

In der Literatur existieren einige Arbeiten, die aufzeigen, wie Verlaufsdaten unterschiedlicher Systeme oder aus verteilten Umgebungen für die Anwendung von *Process Mining* aufbereitet werden kann.
In [30] stellen Dustdar, Hoffmann und van der Aalst die Anwendung TeamLog vor. Diese wurde entwickelt, um Log-Daten des Kollaborationswerkzeugs Caramba aufzubereiten, so dass die *Process-Mining*-Anwendung EMiT (siehe Abschnitt 5.7.3) von der Arbeitsgruppe von van der Aalst auf die Log-Daten angewendet werden kann.

Maruster et al. zeigen in [62] am Beispiel von Zulieferketten, wie Daten von verteilten Prozessen gesammelt werden können. Voraussetzung ist dabei eine globale Referenz, z.B. eine Bestellnummer, so dass die Log-Daten stets einer Bestellung und damit einer Prozessinstanz zugeordnet werden können.

Die genannten Beispiele zeigen, dass es möglich ist, Log-Daten entsprechend für *Process Mining* aufzubereiten, auch wenn diese Daten nicht von einem WfMS generiert wurden. Nichtsdestotrotz ist es notwendig, die sich Log-Daten jeweils eindeutig einer Prozessinstanz zuordnen lassen.

4.3.2 Integration in den KDD-Prozess

Das Problem, entsprechende Verlaufsdaten für die Anwendung von *Process Mining* bereitzustellen, ist ein grundsätzliches Problem, welches die Anwendbarkeit von *Process Mining* maßgeblich bestimmt. Dennoch wird es eher als ein Rahmenproblem angesehen und als außerhalb des Themas *Process Mining* selbst betrachtet.
Da *Process Mining* allerdings als eine Data-Mining-Methode angesehen wird und damit auch im Kontext von *Knowledge Discovery in Databases* (KDD) betrachtet werden kann, werden die vorverarbeitenden Schritte zur Datensammlung und Datenaufbereitung von KDD auch für *Process Mining* in Anspruch genommen [76].
Darüber hinaus gibt es einige Vorschläge, Verlaufsdaten in einem Data Warehouse zu verwalten, um einen effizienten Zugriff zu ermöglichen [54, 11, 31, 97, 99, 67].

4.4 Zusammenfassung

Das Potential von *Process Mining* liegt nicht zuletzt auch darin, dass wenig Prämissen bezüglich der erforderlichen Eingabedaten gemacht werden. Vor allem die für den Einsatz von *Control Flow Mining* notwendigen Daten sind in vielen Einsatzszenarien, sei es im Workflow-Kontext oder auch in anderen Umgebungen, direkt verfügbar oder können ohne weiteres auf die benötigten ereignisbasierten Daten abgebildet werden.

Ein offenes Problem bleibt allerdings die Bereitstellung von Daten, wenn unterschiedliche Systeme oder verteilte Prozesse involviert sind. Da es notwendig ist, die Log-Daten eindeutig Prozessinstanzen zuzuordnen, werden in diesem Fall globale Referenzen benötigt. In einem Bestellprozess wäre dies beispielsweise eine Bestellnummer.
Leider wird das Problem der Datenbereitstellung und -aufbereitung als außerhalb des Kontextes von *Process Mining* betrachtet. Für eventuelle Lösungen wird daher stets auf den KDD-Prozess verwiesen.

Kapitel 5

Control Flow Mining

5.1 Einleitung

Die bisherige Forschung zu *Process Mining* legt den Fokus auf die Ableitung des Kontrollflusses von Prozessen aus Verlaufsdaten. Dementsprechend viel Literatur ist zu diesem Thema vorhanden. Nachdem wir in Kapitel 3 bereits auf Anwendungsmöglichkeiten von *Control Flow Mining* eingegangen sind, soll dieses Kapitel theoretische Grundlagen einführen und einen Überblick über bestehende Ansätze geben.

Zunächst gehen wir in Abschnitt 5.2 genauer auf die Problemstellung von *Control Flow Mining* ein. Besondere Eigenschaften der Problemstellung werden in Abschnitt 5.3 erläutert. Dabei geht es insbesondere darum, schwierige Aspekte und grundsätzliche Lösungsansätze näher zu bringen. In Abschnitt 5.4 gehen wir auf die Gebiete Grammatische Inferenz sowie Mining häufiger Muster als verwandte Fragestellungen ein. Viele Ansätze zu *Control Flow Mining* bedienen sich Techniken aus diesen Gebieten. In Abschnitt 5.5 werden Eigenschaften bestehender Ansätze vorgestellt. In Abschnitt 5.6 werden Ansätze vorgestellt, die das Ziel verfolgen, häufig ausgeführte Prozessfragmente zu finden. Ansätze, die Prozessmodelle aus Verlaufsdaten ableiten, werden in Abschnitt 5.7 vorgestellt. Alle Ansätze im Detail zu erläutern würde den Rahmen dieser Arbeit sprengen. Daher sollen nur die grundsätzlichen Funktionsprinzipien der Ansätze erläutert werden. In Abschnitt 5.8 gehen wir auf die Ableitung von Transitionsbedingungen ein. Abschnitt 5.9 behandelt einige Ansätze zur Unterstützung der Evaluation der abgeleiteten Prozessmodellen. Ergebnisse von Studien über den Einsatz von *Control Flow Mining* in der Praxis werden in Abschnitt 5.10 zusammengefasst. Das Kapitel schließt in Abschnitt 5.11 mit einem Ausblick.

5.2 Problemstellung

Bei *Control Flow Mining* geht es darum, eine strukturierte Prozessbeschreibung aus Verlaufsdaten vergangener Prozessausführungen zu extrahieren [87]. Eine strukturierte Prozessbeschreibung kann dabei, je nach Ansatz und zur Verfügung stehender Informationen, unterschiedlich genau ausfallen. So kann dies beispielsweise ein einfacher gerichteter Graph sein, der die Ordnungsbeziehungen von Aktivitäten modelliert, oder auch ein Prozessmodell mit Kontrollflusskonstrukten und Performanzwerten.

Wir unterscheiden dabei zwischen der Ableitung vollständiger Prozessmodelle, die das gesamte Log abdecken, und der Ableitung häufig ausgeführter Prozessmuster. Letzteres bezeichnen wir als *Workflow Pattern Mining*.

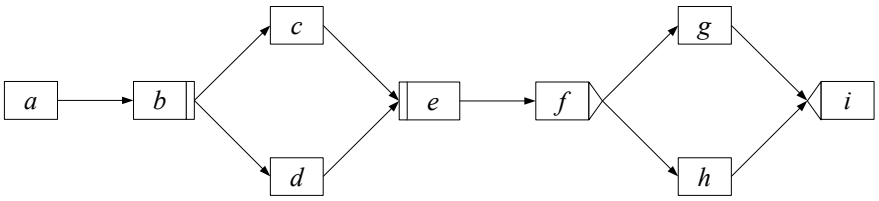

Abbildung 5.1: Ein Beispielprozessmodell

Im folgenden sind mögliche Aktivitäts- und Ereignisspuren für das Prozessmodell in Abbildung 5.1 aufgeführt.

$< a, b, c, d, e, f, g, i >$
$< a, b, d, c, e, f, g, i >$
$< a^+, a^-, b^+, b^-, c^+, d^+, d^-, c^-, e^+, e^-, f^+, f^-, g^+, g^-, i^+, i^- >$

Ziel von *Control Flow Mining* ist es, ein Prozessmodell bzw. ein Prozessmuster auf Grundlage solcher Spuren zu rekonstruieren. Das abgeleitete Prozessmodell muss in erster Linie konsistent zum Log sein. Das bedeutet, dass das Modell in der Lage ist, die entsprechenden Spuren zu generieren. Für *Workflow Pattern Mining* sind speziell häufig auftretende Spurteile interessant.

Eine Menge von Spuren kann in der Regel jedoch von unterschiedlichen Modellen generiert werden. Ein weiterer wichtiger Aspekt ist daher die Ableitung eines möglichst minimalen Modells. Das Modell soll möglichst keine falschen Abhängigkeiten enthalten. Da in unserem Beispielprozess keine Kante und damit auch keine direkten Abhängigkeiten zwischen Aktivität a und Aktivität c besteht, soll das abgeleitete Modell diese auch nicht enthalten.

Eine weitere Fragestellung von *Control Flow Mining* ist die Ableitung von Transaktionsbedingungen. In Verbindung mit Informationen über den Werteverlauf von Variablen kann

beispielsweise für den Split beim Knoten der Aktivität f abgeleitet werden, unter welchen Bedingungen welche Folgeaktivität (g oder h) ausgeführt wird. Die Ableitung des Kontrollflusses schließt diese Information nicht mit ein. Daher können die Transitionsbedingungen in einem separaten Schritt abgeleitet werden. Auf die Ableitung von Transitionsbedingungen gehen wir in Abschnitt 5.8 genauer ein.

5.3 Eigenschaften der Problemstellung

5.3.1 Kausale Beziehungen

Die wesentliche Aufgabe bei der Ableitung des Kontrollflusses besteht darin, kausale Beziehungen zwischen Aktivitäten anhand der Spuren aufzudecken. Eine kausale Beziehung besteht zwischen zwei Aktivitäten a und b, wenn b erst nach der Beendigung von a ausgeführt werden kann. Interessant sind vor allem direkte kausale Beziehungen, da diese in einem Prozessmodell Kanten darstellen. Eine direkte kausale Beziehung zwischen zwei Aktivitäten a und b bezeichnen wir mit $a \rightarrow b$.

Anhaltspunkte, die auf eine kausale Beziehung zwischen zwei Aktivitäten hindeuten, finden sich in den Folgebeziehungen der Aktivitäten in den Spuren. Kommen zwei Aktivitäten a und b stets in der Sequenz ab im Log vor, so deutet das darauf hin, dass $a \rightarrow b$ gilt. Das ist beispielsweise in der Abbildung 5.1 der Fall. Bei einem fehlerfreien Log wird die Sequenz ab in jeder Spur vorkommen.

Ein Weg, um kausale Beziehungen festzustellen, ist die Menge aller potentiellen kausalen Beziehungen, also z.B. alle Folgebeziehungen im Log, zu betrachten. Diese Menge kann Schritt für Schritt verringert werden, indem Beziehungen zwischen voneinander unabhängigen Aktivitäten aus der Menge entfernt werden. Im Prozessmodell in Abbildung 5.1 sind beispielsweise die Aktivitäten c und d unabhängig voneinander. Sie können parallel ausgeführt werden und stehen daher in keiner kausalen Beziehung. Dieses Vorgehen wird von vielen der in dieser Arbeit vorgestellten Ansätze verwendet.

Falls sowohl die Sequenz ab als auch die Sequenz ba im Log vorkommen, ist es nicht einfach zu entscheiden, ob zwischen diesen Aktivitäten eine kausale Beziehung besteht. Aspekte, die hier mit hineinspielen, sind Rauschdaten, parallele Abläufe und zyklische Abläufe. So könnten a und b z.B. in einem Zyklus stehen. Dann bestünden sowohl die Beziehung $a \rightarrow b$ als auch die Beziehung $b \rightarrow a$. Andererseits ist es auch möglich, dass a und b parallel ausgeführt werden und aufgrund der absoluten Ordnung in den Aktivitätsspuren in diesen beiden Reihenfolgen erfasst wurden. Kommt die Sequenz ab sehr häufig im Log vor und die Sequenz ba hingegen nur sehr selten, könnte auch ein fehlerhaftes Log für letztere Sequenz verantwortlich sein. Die Sequenz ba würde in dem Fall kein Hinweis für die Beziehung $b \rightarrow a$ sein.

All die genannten Aspekte aber auch weitere Aspekte, z.B. nicht-eindeutige Namen und kausale Beziehungen zwischen nicht direkt benachbarten Aktivitäten, machen es zu einer nicht-trivialen Aufgabe, kausale Beziehungen aus den Verlaufsdaten abzuleiten. In den folgenden Abschnitten gehen wir genauer auf die einzelnen Aspekte ein.

5.3.2 Parallele Abläufe

Da die während der Prozessausführung mitprotokollierten Ereignisse in sequentieller Form vorliegen, ist es nicht einfach, parallele Ausführungspfade zu erkennen. Werden Start- und Endereignisse von Aktivitäten berücksichtigt[1], können diese direkt Aufschluss über parallele Ausführungspfade liefern. Überschneiden sich z.B. die Start- und Endereignisse zweier Aktivitäten, so ist dies ein Zeichen dafür, dass diese Aktivitäten parallel ausgeführt werden. Abbildung 5.2 veranschaulicht die Situation.

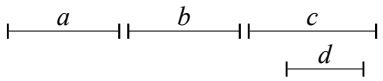

Abbildung 5.2: Zeitliche Darstellung der Ereignisspur für das Prozessmodell in Abbildung 5.1

Nur einige der in dieser Arbeit vorgestellten Ansätze nutzen diese Information, um direkt auf Parallelität zu schließen. Oft wird von den konkreten Start- und Endereignissen abstrahiert. Aktivitäten werden entweder als atomare Einheit betrachtet, oder es werden jeweils nur die Endereignisse berücksichtigt (siehe auch Kapitel 4).

Um von sequentiellen Daten auf Parallelität zu schließen, gehen viele Ansätze von bestimmten Annahmen aus. Aktivitäten, die parallel ausgeführt werden können, werden typischerweise nicht stets in derselben Reihenfolge protokolliert. Daher kann das verschränkte Vorkommen zweier Aktivitäten (*Interleavings*) in den Spuren auf parallele Abläufe hindeuten. Betrachten wir noch einmal die Aktivitätsspuren für das Prozessmodell in Abbildung 5.1:

$< a, b, c, d, e, f, g, i >$
$< a, b, d, c, e, f, g, i >$

In unserem Beispiel wird c in der ersten Spur vor d ausgeführt. In der zweiten Spur tritt der umgekehrte Fall auf. Daraus schließen viele Algorithmen, z.B. der α-Algorithmus (siehe Abschnitt 5.7.3), auf Parallelität dieser Aktivitäten.

Ein Problem bei dieser Annahme besteht darin, dass nicht alle möglichen Verschachtelungen auftreten müssen. Bei 10 Aktivitäten, die parallel ausgeführt werden können, sind

[1] Anzumerken ist, dass dazu kein konkreter Zeitstempel der Ereignisse erforderlich ist. Es genügt eine zeitliche Ordnung der Ereignisse.

10! = 3628800 Verschränkungen möglich [86]. Darum ist es unwahrscheinlich, dass alle Verschränkungen in den Verlaufsdaten auftreten.

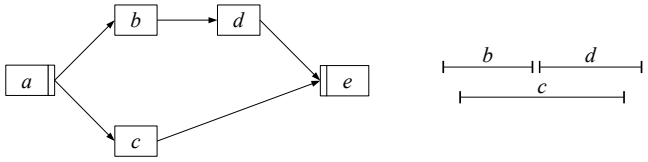

Abbildung 5.3: Aktivitäten mit unterschiedlichen Ausführungszeiten

Darüber hinaus können Aktivitäten unterschiedliche Ausführungszeiten haben, was ebenfalls die Wahrscheinlichkeit des Auftretens aller möglichen Verschachtelungen reduziert. In Abbildung 5.3 besitzt Aktivität c eine wesentlich längere Ausführungszeit als Aktivität b und d. Ein verschränktes Vorkommen von b und c ist daher sehr unwahrscheinlich[2].

5.3.3 Nicht-Injektive Aktivitätszuordnungsfunktion

Das Problem der nicht-injektiven Aktivitätszuordnungsfunktion [42] ist auch unter den Namen *Non-unique Names* oder *Duplicate Tasks* bekannt. Es bezeichnet die Situation, in der eine Aktivität unterschiedlichen Knoten des Prozessmodells zugeordnet ist. Folglich kann diese Aktivität mehrfach in einer Spur auftreten. Einige Workflow-Management-Systeme, z.B. Staffware [81], lassen zwei gleichnamige Aktivitäten in einem Prozessmodell nicht zu. Moderne WfMS wie ADEPT [73, 70, 71] unterscheiden explizit zwischen Knoten im Prozessmodell und den Knoten zugeordneten Aktivitäten. Dies erfolgt über eine Aktivitätszuordnungsfunktion, die auch die Zuordnung einer Aktivität zu mehreren Knoten im Prozessmodell erlaubt. Dass dies sinnvoll ist, zeigt das folgende Beispiel.

Abbildung 5.4 zeigt einen Prozess, bei dem die Aktivitäten d und e jeweils mehreren Knoten zugeordnet sind. Dieser Prozess könnte beispielsweise einen Reisebuchungsprozess darstellen. Entweder werden Flugticket (d) und Übernachtung (e) separat oder zusammen in einem Paket gebucht[3].
Fast alle in dieser Arbeit vorgestellten Ansätze setzen eine injektive Aktivitätszuordnungsfunktion voraus.

[2]Dies gilt insbesondere, wenn zur Bildung der Aktivitätsspur nur die Endereignisse der Aktivitäten berücksichtigt werden.

[3]Dieses Beispiel ist der Arbeit von de Medeiros et al. [25] entnommen.

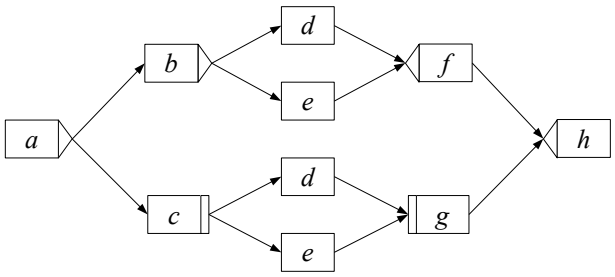

Abbildung 5.4: Ein Prozess mit nicht-injektiver Aktivitätszuordnungsfunktion (nach de Medeiros et al. [25])

5.3.4 Zyklische Abläufe

Wie sich die Ableitung zyklischer Abläufe aus Verlaufsdaten gestaltet, hängt auch wesentlich von den Logmechanismen des Systems ab, welches die Verlaufsdaten erzeugt.

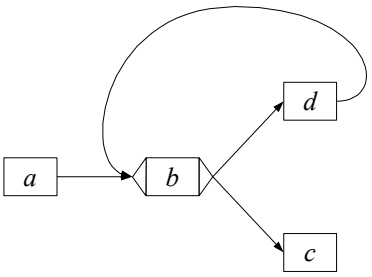

Abbildung 5.5: Prozessmodell mit einem Zyklus der Länge zwei

Einige Systeme ermöglichen es, Zyklendurchläufe direkt zu erkennen, da sie bei wiederholter Ausführung von Aktivitäten entsprechende Numerierungen im Log-Eintrag vornehmen. Solch eine Spur für das Prozessschema aus Abbildung 5.5 ist unten aufgeführt. Ein solcher Protokollierungsmechanismus zur Identifizierung von Zyklen ist zwar hilfreich, kann jedoch nicht vorausgesetzt werden.

$$< a, b, d, b_1, d_1, b_2, d_2, b_3, c >$$

Einige *Control-Flow-Mining*-Ansätze, z.B. der Ansatz von Agrawal (siehe Abschnitt 5.7.2.1), führen eine künstliche Unterscheidung der Aktivitäten durch Numerieren mehrfacher Vorkommen einer Aktivität herbei und erhalten damit Spuren wie vorangehend beschrieben. Nach der Mining-Prozedur werden die mehrfachen Vorkommen wieder auf die jeweiligen Aktivitäten abgebildet. Ein solches Vorgehen ist allerdings nur möglich, wenn mehrfache Vorkommen einer Aktivität innerhalb einer Spur nur durch Zyklen hervorgerufen werden können. Dies ist

der Fall, wenn die Aktivitätszuordnungsfunktion injektiv ist (siehe auch Abschnitt 5.3.3) oder entsprechende Unterscheidungsmechanismen vorliegen.

5.3.5 Non-Free-Choice-Konstrukte

Non-Free-Choice-Konstrukte [26] sind aus dem Bereich der Petri-Netze bekannt.

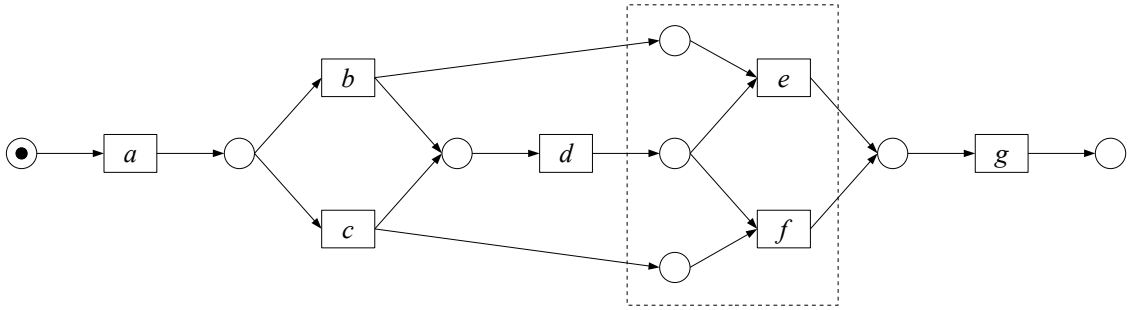

Abbildung 5.6: Ein Prozessmodell mit einem Non-Free-Choice-Konstrukt

Das sind Konstrukte, die Auswahl und Synchronisation kombinieren (siehe Abbildung 5.6). Die einzigen Spuren, die vom Prozessmodell der obigen Abbildung erzeugt werden, sind:

$< a, b, d, e, g >$
$< a, c, d, f, g >$

Die Entscheidung, ob Aktivität e oder Aktivität f ausgeführt wird, hängt auch davon ab, ob Aktivität a oder Aktivität b ausgeführt wurde. In den Spuren stehen b und e allerdings in keiner direkten Folgebeziehung, d.h. die Sequenz be wird bei einem rauschfreien Log nicht auftreten. Dennoch stehen b und e in einer kausalen Beziehung zueinander.

Kausale Beziehungen zwischen nicht direkt benachbarten Aktivitäten sind schwierig aufzudecken. Lokale Mining-Strategien, die nur direkte Folgebeziehungen in den Spuren berücksichtigen, um kausale Beziehungen abzuleiten, z.B. der α- und β-Algorithmus (siehe Abschnitt 5.7.3), können folglich keine kausalen Beziehungen zwischen nicht direkt benachbarten Aktivitäten aufdecken. Non-Free-Choice-Konstrukte, wie in Abbildung 5.3.5 dargestellt, können damit nicht erkannt werden. Globalere Strategien haben bessere Chancen, mit solchen Konstrukten umgehen zu können [87] (siehe auch Abschnitt 5.5.2).

5.3.6 Unvollständige Verlaufsdaten

Unvollständige Verlaufsdaten sind ein wichtiger Aspekt bei *Control Flow Mining*. Gerade bei komplexen Prozessen mit vielen Verzweigungen, aber auch aufgrund dessen, dass die Ausführungswahrscheinlichkeiten unterschiedlicher Pfade nicht notwendigerweise gleich sind,

ist es unwahrscheinlich, dass ein vollständiges Log vorliegt. Vollständigkeit ist dabei für jeden Ansatz anders aufzufassen und soll als die Bereitstellung einer Grundmenge an Log-Information verstanden werden, die notwendig ist, um den Prozess korrekt nach ansatzspezifischen Kriterien zu rekonstruieren. Beim α-Algorithmus ist beispielsweise ein vollständiges Log hinsichtlich der direkten Folgebeziehung zweier Aktivitäten in den Spuren notwendig, um den Prozess rekonstruieren zu können. Liegt kein vollständiges Log vor, entwickelt der Algorithmus zwar ein Modell, jedoch muss dieses nicht dem tatsächlichen Prozessmodell entsprechen.

Generell gilt, dass nur Verhalten, das auch im Log sichtbar wird, aufgedeckt werden kann. Pfade, die nur sehr selten oder nie ausgeführt werden, laufen daher Gefahr, nicht entdeckt zu werden.

5.3.7 Rauschdaten

Aktivitäten, die manuell verrichtet und daher von der Protokollierungsumgebung nicht erfasst werden, kommen folglich nicht in den Spuren vor und können daher auch nicht berücksichtigt werden. Dies ist ein grundsätzliches Problem und nicht unbedingt als Rauschen anzusehen. Der Umgang mit fehlerhaften Daten ist jedoch ein wichtiger Aspekt von *Control Flow Mining*.

Unterschiedlichste Situationen können zu Rauschdaten im Log führen. So können Ereignisse bei der Protokollierung verloren gehen. Aufgrund von Verzögerungen können Ereignisse auch in einer falschen Reihenfolge protokolliert werden. Auch für die Prozessausführung irrelevante Ereignisse, z.B. ein privates Telefonat, können mitprotokolliert werden. In einem modernen Workflow-Management-System ist die letztere Situation höchst unwahrscheinlich, da Prozessinstanzen auf Grundlage eines Schemas ausgeführt werden und das WfMS die Protokollierung der Ereignisse regelt. In anderen Umgebungen kann es durchaus vorkommen, dass irrelevante Ereignisse im Log auftauchen. Die genannten Situationen können die Folgebeziehungen der Aktivitäten in den Spuren verändern und damit Kausalitätsbeziehungen zwischen Aktivitäten verwischen oder die Ableitung falscher Kausalitätsbeziehungen bewirken.

Viele Ansätze verwenden Schwellenwerte, um Rauschdaten herauszufiltern. Die Annahme dabei ist, dass Rauschdaten willkürlich und nicht häufig vorkommen werden. Geht bei dem Prozess in Abbildung 5.1 beim Protokollierungsvorgang beispielsweise das der Aktivität b zugeordnete Ereignis verloren, so könnte eine Spur dieses Prozesses wie folgt aussehen:

$$< a, c, d, e, f, g, i >$$

Da Aktivität c in dieser Spur nun direkt auf Aktivität a folgt, könnte die kausale Beziehung $a \rightarrow c$ fälschlicherweise geschlossen werden. Da die Sequenz ac jedoch nur selten vorkommt, kann sie mit Hilfe eines Schwellenwertes für die Häufigkeit der Sequenzen herausgefiltert werden.

5.3.8 Ausnahmefälle und fehlerhafte Instanzen

Ein weiteres Problem besteht bei Verlaufsdaten von Instanzen, die eine Ausnahmeausführung oder eine fehlerhafte Ausführung darstellen (siehe auch Abschnitt 5.3.9).

Unter fehlerhaften Ausführungen werden Instanzen verstanden, die ihr Ziel verfehlen oder nicht effizient sind. Unter einer Ausnahmeausführung wird dagegen eine korrekte Instanz verstanden, deren Ausführung jedoch von der üblichen Arbeitsweise abweicht [42]. Anders als Rauschdaten, die durch Fehler beim Protokollierungsvorgang hervorgerufen werden, sind Ausnahmefälle und fehlerhafte Instanzen für eine Evaluation des Prozesses sicherlich von größerem Interesse.

Es gibt unterschiedliche Ansichten, wie mit diesen Daten verfahren werden soll. Zum einen besteht der Anspruch, den Prozess mit all den aufgetretenen Ausnahmen und Fehlerfällen so wiederzugeben, wie er wirklich ausgeführt wurde. Dies ist etwa bei Herbst (siehe Abschnitt 5.7.7) der Fall. Herbst weist darüber hinaus in [42] darauf hin, dass es nicht sinnvoll sei, die Entscheidung, ob ein bestimmter Fall in das resultierende Workflow-Modell aufgenommen werden soll, allein von der Häufigkeit seines Auftretens abhängig zu machen. Die Begründung dafür ist, dass es in der Praxis durchaus Fälle geben kann, die zwar selten auftreten, aber dennoch aufgrund ihrer Wichtigkeit in das Workflow-Modell aufgenommen werden sollten. Dies würde implizieren, dass Ausnahmefälle nicht als Rauschdaten interpretiert werden.
Zum anderen gibt es die Ansicht, dass Ausnahmen nicht vom abgeleiteten Prozessmodell abgedeckt werden sollen und demnach wie Rauschdaten behandelt werden [84].

In jedem Fall aber ist es schwer, wenn nicht gar unmöglich, Ausnahmen von Rauschdaten zu unterscheiden, da sich Ausnahmeausführungen ebenso wie Rauschdaten z.B. in veränderten Folgebeziehungen der Aktivitäten im Log äußern können.
Daher wird es eher der Fall sein, dass ein *Control-Flow-Mining*-Ansatz entweder den Anspruch hegt, gegenüber Ausnahmefällen und Rauschdaten robust zu sein oder eben alle Daten, einschließlich Rauschdaten, zu berücksichtigen.

5.3.9 Positive und negative Beispiele

Die Aufgabe, ein Prozessmodell aus den Verlaufsdaten abzuleiten, kann als ein Lernproblem (siehe z.B. [66]) aus Beispielen aufgefasst werden. Die Spuren der Verlaufsdaten stellen dabei die Beispiele dar.
Es kann prinzipiell zwischen positiven und negativen Beispielen unterschieden werden. Positive Beispiele wären in diesem Fall Verlaufsdaten von korrekten Prozessinstanzen. Negative Beispiele wären Verlaufsdaten von ungültigen Instanzen.

Theoretisch besteht auch bei *Control Flow Mining* die Möglichkeit, sowohl von positiven

als auch von negativen Beispielen zu lernen. Die Verwendung von negativen Beispielen hat den Vorteil, dass einer Übergeneralisierung entgegen gewirkt wird. Praktische Hürden sprechen jedoch gegen die Verwendung von negativen Beispielen. Die Unterscheidung zwischen positiven und negativen Beispielen wäre zu kompliziert, da Fehler auf verschiedenen Ebenen auftreten können: auf der inhaltlichen Ebene (innerhalb einer Aktivität) sowie auf der Koordinationsebene (vgl. [42]). Instanzen mit Fehler auf der inhaltlichen Ebene können auf der Koordinationsebene dennoch korrekt sein. Zudem ist die Bereitstellung einer größeren Menge von negativen Beispielen in der Praxis sehr schwierig und aufwändig und würde daher eine große Hürde für den praktischen Einsatz von *Control Flow Mining* darstellen.

Alle bisherigen Ansätze zu *Control Flow Mining* arbeiten ausschließlich mit positiven Beispielen.

5.4 Verwandte Gebiete

5.4.1 Grammatische Inferenz

Im Zusammenhang mit *Control Flow Mining* wird oft das Gebiet der Grammatischen Inferenz (*Grammatical Inference*) erwähnt. Grammatische Inferenz (siehe z.B. [63]) ist ein Teilgebiet des induktiven Lernens (siehe [66]) und befasst sich mit dem Erlernen von Grammatiken aus Wortbeispielen [42, 18]. Zu Wortbeispielen, z.B. *abcd*, als Eingabe soll eine Grammatik für die Sprache der Beispiele erzeugt werden.
Es ist leicht nachzuvollziehen, dass die Spuren der Instanzen in den Verlaufsdaten als Beispielwörter einer Sprache aufgefasst werden können. Die Grammatik ist im Fall von *Control Flow Mining* das den Instanzen zugrundeliegende Prozessmodell.

Besonders die FSM-Synthese, ein Teilgebiet der Grammatischen Inferenz, erscheint für *Control Flow Mining* interessant. Bei der FSM-Synthese geht es um die Generierung eines endlichen Zustandsautomaten (*Final State Machine*), kurz FSM, als Grammatik für gegebene Beispielwörter.

Ein wesentlicher Unterschied zwischen *Control Flow Mining* und FSM-Synthese besteht jedoch darin, dass parallele Abläufe bei der FSM-Synthese keine Rolle spielen. Für den Prozesskontext von *Control Flow Mining* sind parallele Abläufe jedoch unabdingbar. Daher ist es nicht möglich, Ansätze der FSM-Synthese direkt für *Control Flow Mining* einzusetzen.

5.4.2 Mining häufiger Muster

Ein weitere Fragestellung, die mit *Control Flow Mining* verwandt ist, ist das Auffinden häufiger Muster (*Mining Frequent Patterns*) aus einer Datenmenge. Dies ist ein Teilgebiet von Data Mining.

Eine typische Fragestellung von *Mining Frequent Patterns* ist beispielsweise, welche Kombinationen von Büchern häufig von Kunden gekauft werden (Warenkorbanalyse). Als häufig gilt ein Muster dann, wenn dessen Vorkommen im Log (*Support*) einen Schwellenwert (*Minimal Support)* erreicht. Um solche Muster zu finden, werden beispielsweise Einkaufstransaktionen der Kunden untersucht.
Im Kontext von *Control Flow Mining* ist jede Spur einer Instanz vergleichbar mit den gekauften Produkten eines Kunden. Häufige Muster in den Spuren entsprechen häufig ausgeführten Teilen des Prozessmodells.

Ist bekannt, welche Elemente in Mustern (*Items*), z.B. Bücher, vorkommen können, kann eine naive Herangehensweise um Muster aufzufinden die sein, alle möglichen Sequenzen zu erzeugen und deren Häufigkeit in den Log-Daten zu überprüfen. Das Erzeugen aller möglichen Kombinationen von Büchern und diese beispielsweise gegen die Log-Daten von Transaktionen eines größeren Online-Buchhändlers zu prüfen ist jedoch undenkbar. Eine klügere Herangehensweise ist daher, die Eigenschaften von häufigen Mustern auszunutzen. Tritt ein Muster, in diesem Fall eine Kombination von Büchern, häufig im Log auf, so müssen entsprechend auch all seine Teilmengen, z.B. jedes einzelne Buch des Musters, häufig sein. Der Algorithmus im Listing 5.1, der dieses Prinzip ausnutzt, wird *Apriori*-Algorithmus genannt.

```
1 i = 0;
2 C_i = {{a} | a is a element }
3 While C_i is not empty do
4     Database pass:
5         For each set in C_i, test whether it is frequent
6         Let L_i be the set of frequent sets from C_i;
7     Candidate formation:
8         Let C_{i+1} be those sets of size i+1 whose all subsets are frequent
9 End
```

Listing 5.1: Ablauf des Apriori-Algorithmus nach [41]

Bei unserem Beispiel mit den Büchern können wir als ersten Schritt die Menge aller Bücher bestimmen, die häufig gekauft werden. Die Bildung der Kandidaten für die nächste Iteration C_2 kann beispielsweise erfolgen, indem Kombinationen aus der Menge L_1, also der Menge aller häufigen Bücher, erzeugt werden. Es gibt zahlreiche Variationen des grundlegenden *Apriori*-Algorithmus [41, 6, 7]. Typischerweise verfolgen diese Variationen folgende Ziele: die Anzahl der Durchläufe durch die Datenmenge zu minimieren, die Anzahl der Kandidaten, die auf Häufigkeit überprüft werden müssen, zu minimieren sowie die Minimierung der Zeit, die benötigt wird, um die Häufigkeit eines Kandidaten zu berechnen [41].

Adaptionen des *Apriori*-Algorithmus werden von einigen der in dieser Arbeit vorgestellten Ansätze zu *Workflow Pattern Mining* (siehe Abschnitt 5.6) verwendet.

Einige Ansätze zu Mining häufiger Muster in der Literatur konzentrieren sich auf das Auffinden häufiger Substrukturen von Graphen (vgl. [36, 51, 95, 57]). Diese Ansätze sind für

das Mining von *Workflow Patterns* sicherlich auch interessant. Da die verwendete Graphrepräsentation jedoch zu einfach ist und keine Kontrollflusskonstrukte, z.B. Joins und Splits, vorgesehen sind, sind diese Ansätze jedoch nur bedingt für diesen Zweck brauchbar.

Weiterhin gibt es auch Ansätze, die Parallelität berücksichtigen. Die Arbeit von Mannila et al. in [59] beschäftigt sich mit dem Auffinden häufiger Episoden. Eine Episode ist dabei eine Menge partiell geordneter Ereignisse. Der Ablauf entspricht im Wesentlichen dem *Apriori*-Algorithmus. Ein wesentlicher Unterschied liegt darin, dass in [59] auch parallele Episoden berücksichtigt werden. Für einen Überblick über die Thematik von *Mining Frequent Patterns* verweisen wir auf [41].

Die Ansätze zu *Workflow Pattern Mining*, die wir im Abschnitt 5.6 vorstellen, finden im wesentlichen eine geeignete Präsentation für den Workflow-Kontext, so dass auch domänenspezifisches Wissen ausgenutzt werden kann, und adaptieren bekannte Techniken.

5.5 Eigenschaften bestehender Ansätze

Nachdem die Randbedingungen von *Control Flow Mining* erörtert wurden, gehen wir nun darauf ein, welche grundsätzlichen Eigenschaften die bestehenden Ansätze aufweisen können.

5.5.1 Algorithmische, heuristische und hybride Ansätze

Bestehende Ansätze können grundsätzlich in drei Kategorien eingeteilt werden: algorithmische Verfahren, heuristische bzw. statistische Verfahren sowie hybride Verfahren.

Rein algorithmische Verfahren erzeugen das Prozessmodell basierend auf Ordnungsbeziehungen. Dabei können auch nachverarbeitende Schritte zum Tragen kommen, z.B. die Knoten zusammenfassen, um das resultierende Modell zu vereinfachen. Beispiele für rein algorithmische Verfahren sind der α und α^+-Algorithmus sowie der β-Algorithmus (siehe Abschnitt 5.7.3).

Rein heuristische Verfahren erzeugen das Prozessmodell auf Grundlage von Häufigkeiten oder Wahrscheinlichkeiten von Sequenzen. Der Vorteil von heuristischen bzw. statistischen Verfahren gegenüber rein algorithmischen Verfahren liegt darin, dass sie robuster gegenüber Rauschdaten sind. Der Ansatz LittleThumb (siehe Abschnitt 5.7.4) kann beispielsweise als rein heuristisch aufgefasst werden.

Hybride Verfahren vereinigen algorithmische und statistische Techniken. Typischerweise sind dies oftmals rein algorithmische Ansätze, die, um eine Robustheit gegenüber Rauschdaten zu ermöglichen, um zusätzliche Schwellenwerte als Parameter erweitert werden. Dies ist z.B. bei den Ansätzen von Datta und Cook und Wolf, die Techniken der FSM-Synthese für *Control Flow Mining* adaptierten, der Fall. Darüber hinaus ist der induktive Ansatz von Herbst (siehe

Abschnitt 5.7.7) ein gutes Beispiel für einen hybriden Ansatz. Er verwendet Techniken aus dem Bereich des maschinellen Lernens und setzt Heuristiken zur Steuerung des Mining-Prozesses ein.

5.5.2 Lokalität und Globalität

Ein wichtiger Aspekt ist die Lokalität bzw. Globalität eines Ansatzes. Lokale Strategien verwenden lokales Wissen, um den Prozess zu rekonstruieren. Ein Beispiel für einen lokalen Ansatz ist der α-Algorithmus (siehe Abschnitt 5.7.3). Einige Algorithmen, wie der erwähnte α-Algorithmus, berücksichtigen nur direkte Folgebeziehungen, also sehr lokales Wissen, um eine mögliche Kausalität zwischen Aktivitäten festzustellen. Eine lokale Strategie läuft Gefahr, kausale Beziehungen nicht aufdecken zu können, die sich nicht lokal bemerkbar machen. Dies betrifft beispielsweise die Non-Free-Choice-Konstrukte.

Globale Ansätze haben bessere Chancen, mit solchen Problemen umgehen zu können. Ein besonders globaler Ansatz ist der genetische Ansatz, der in Abschnitt 5.7.8 beschrieben wird. Statt Schritt für Schritt kausale Beziehungen abzuleiten, erzeugt der genetische Ansatz das Prozessmodell in einem Schritt und validiert es gegen die Verlaufsdaten. Globale Ansätze haben darüber hinaus im allgemeinen den Vorteil, robuster gegenüber Rauschdaten zu sein [87]. Der Nachteil ist, dass die Komplexität globaler Ansätze meist höher ist als die lokaler Ansätze.

5.5.3 Induktiver Bias

Die bestehenden Ansätze unterscheiden sich stark in den Voraussetzungen und Annahmen, die sie machen. Je mehr Annahmen ein Verfahren von vornherein trifft, desto höher ist der induktive Bias dieses Verfahrens. Wird das Problem von *Control Flow Mining* als eine Suche über den Suchraum aller möglichen Prozessmodelle betrachtet, wird der Suchraum mit den Anfangsannahmen eingegrenzt [87].

Der induktive Bias kann beispielsweise die Prozessmodellierung betreffen. Ein Beispiel für einen starken induktiven Bias bezüglich Prozessmodellierung ist bei dem Ansatz von Schimm (siehe Abschnitt 5.7.5) gegeben. Dieser Ansatz setzt blockstrukturierte Prozesse voraus. Weitere Annahmen werden von vielen Ansätzen von vornherein getroffen. So setzen die meisten Ansätze eine injektive Aktivitätszuordnungsfunktion voraus (siehe Abschnitt 5.3.3).

Annahmen können nur dann sinnvoll getroffen werden, wenn von vornherein beispielsweise klar ist, dass nur blockstrukturierte Prozesse auftreten können oder dass eine Aktivität nur einem Knoten im Prozessmodell zugeordnet sein kann. In der Regel ist dies jedoch unrealistisch, insbesondere wenn *Control Flow Mining* zur Aufdeckung unbekannter Prozesse dienen soll. Ansätze, die einen weniger starken induktiven Bias besitzen, können folglich auf ein breiteres Spektrum von Prozessen eingesetzt werden. Auf der anderen Seite schränken Annahmen die

Komplexität der Problemstellung ein.

5.6 Workflow Pattern Mining

Im Unterschied zu den Ansätzen, die wir in Abschnitt 5.7 vorstellen werden, geht es bei *Workflow Pattern Mining* nicht darum, ein vollständiges Modell abzuleiten. Vielmehr verfolgen die in diesem Abschnitt vorgestellten Ansätze das Ziel, häufig ausgeführte Prozessfragmente zu finden. Da den Fragmenten nicht notwendigerweise ein Prozessmodell zugrunde liegt, ist *Workflow Pattern Mining* insbesondere für die Untersuchung unstrukturierter Prozesse, z.B. Krankenhausabläufe, interessant (vgl. [87]).
Auch im Kontext von *Emergent Workflow* [10] und *Pattern-driven Process Design* [96] sind Ansätze zum Auffinden häufiger Prozessfragmente von großem Interesse. Häufig wiederkehrende Muster in den Verlaufsdaten können auf den Ablauf von Routinefällen hindeuten. Häufig auftretende Prozessfragmente können auf diese Weise gesammelt und nach Evaluationsprozessen gegebenenfalls zu Workflow-Modellen veredelt werden.
Im folgenden werden drei Ansätze zum Auffinden von Workflow Patterns vorgestellt, die sehr unterschiedliche Ziele verfolgen.

5.6.1 WorkflowMiner

In [33, 32] stellen Gaaloul et al. einen Ansatz zur Ableitung von Workflow Patterns aus Ereignisspuren vor. Dazu wird eine Kombination aus statistischen und algorithmischen Techniken eingesetzt. Der Mining-Prozess besteht aus drei Teilen. Zunächst wird eine Tabelle aufgestellt, die für jede Aktivität ihre Auftrittshäufigkeit im Log festhält. Darüber hinaus werden Relationen (Folgebeziehungen) zwischen einer Aktivität a und ihren Vorgängern festgehalten. Da eine Aktivität, z.B. aufgrund von parallelen Abläufen, auch in einer kausalen Beziehung mit indirekten Logvorgängern stehen kann, werden auch indirekte Folgebeziehungen berücksichtigt. Wie weit indirekte Vorgänger einer Aktivität berücksichtigt werden, wird vom Anwender festgelegt.

Im zweiten Schritt wenden Gaaloul et al. einen bekannten Algorithmus zur Auffindung häufiger Episoden an (siehe auch Abschnitt 5.4.2 oder [59]). Damit werden parallele und sequentielle Episoden[4] gefunden.

In einer dritten Verarbeitungsphase werden Regeln auf die gefundenen Episoden verwendet, um Kontrollstrukturen zu identifizieren.

[4]Da hier auf Ereignisspuren mit Start- und Endereignissen von Aktivitäten gearbeitet wird, können Aktivitäten anhand der Spur partiell geordnet werden. Darum kann der angewendete Algorithmus auch parallele Episoden finden.

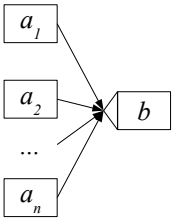

Abbildung 5.7: Ein XOR-Join

Auf eine sequentielle Episode kann beispielsweise die folgende Regel angewendet werden, um ein XOR-Join, wie in Abbildung 5.7 gezeigt, zu identifizieren:

$$(\sum_{i=0}^{n} P(b/a_i) = 1) \wedge (\exists 0 \leq i, j \leq n; P(a_i/a_j) = 0) \wedge (\sum_{i=0}^{n} (\sharp a_i) = \sharp b)$$

$P(b/a_i) = 1$ bedeutet dabei, dass vor b stets ein a_i ausgeführt wurde. $\sharp a_i$ und $\sharp b$ bezeichnen jeweils die Häufigkeit von a_i und b.

Insgesamt haben Gaaloul et al. Regeln für Sequenz, XOR-Split, XOR-Join, AND-Split, AND-Join sowie Multi-Choice-Split und Multi-Choice-Join definiert. Mit WorkflowMiner haben Gaaloul et al. auch eine Implementierung für ihren Ansatz vorgestellt. WorkflowMiner ist in Java implementiert und basiert auf dem Workflow-Management-System Bonita (siehe bonita.objectweb.org).

5.6.2 *TP-Graph*, *TP-Itemset* und *TP-Sequence*

In [47, 91] stellen Hwang et al. drei Algorithmen zum Auffinden häufiger Muster vor. Allerdings verfolgen Hwang et al. nicht das Ziel, kausale Beziehungen oder Kontrollflusskonstrukte abzuleiten. Vielmehr geht es ihnen um das Auffinden häufiger zeitlicher Beziehungen (*Temporal Relationships*) zwischen Aktivitäten. Darum berücksichtigen Hwang et al. in ihrer Arbeit die Ausführungszeit von Aktivitäten, markiert durch das Start- und das Endereignis einer Aktivität.

Temporäre Beziehungen können als *Temporal Graph* dargestellt werden. Abbildung 5.9 zeigt einen *Temporal Graph* für die Spur in Abbildung 5.8. Kanten zwischen zwei Aktivitäten stellen eine direkte Folgebeziehung dieser Aktivitäten bezüglich der Ereignisspuren dar. Gibt es keinen Pfad zwischen zwei Aktivitäten, so überlappen sie sich in ihrer Ausführungszeit.

Die von Hwang et al. vorgestellten Algorithmen *TP-Graph*, *TP-Itemset* und *TP-Sequence* finden häufig vorkommende maximale *Temporal Graphs* aus Verlaufsdaten. Maximal ist ein *Temporal Graph* dann, wenn er nicht Subgraph eines anderen *Temporal Graphs* ist.

Da sich die drei Algorithmen vom Ablauf her sehr ähneln und der wesentliche Unterschied nur in der Repräsentation der Daten und der Muster besteht, gehen wir im folgenden nur auf

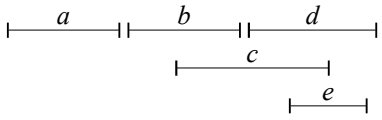

Abbildung 5.8: Zeitliche Beziehungen zwischen Aktivitäten

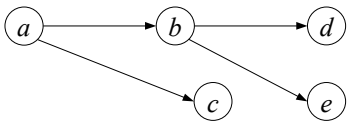

Abbildung 5.9: Temporal Graph für die zeitlichen Beziehungen der Aktivitäten aus Abbildung 5.8

den *TP-Graph*-Algorithmus ein. Der Ablauf der Algorithmen entspricht dabei im wesentlichen dem in Abschnitt 5.4.2 bereits vorgestellten *Apriori*-Algorithmus (vgl. Listing 5.1).

Der *TP-Graph*-Algorithmus verwendet *Temporal Graphs* als Repräsentation der Daten und Muster. Der Ablauf ist wie folgt:

- Für $k = 1$ ist C_k eine Menge von *Temporal Graphs* mit nur einer Aktivität.
- Für andere k wird die Menge der Kandidaten C_k gebildet, indem jeweils zwei häufige *Temporal Graphs* der Größe $k - 1$ zusammengefasst werden.
- Die Kandidaten C_k werden gegen das Log geprüft, um L_k zu erhalten.
- Im Unterschied zum *Apriori*-Algorithmus aus Listing 5.1 werden in jeder Iteration alle nicht maximalen Muster aus der Ergebnismenge gelöscht, da Hwang et al. maximale *Temporal Graphs* erhalten wollen.

Der *TP-Itemset*-Algorithmus verwendet zur Repräsentation eines *Temporal Graphs* eine Menge von Items (*Itemset*). Jede temporäre Beziehung zwischen den Aktivitäten wird als ein Item (z.B. $a \rightarrow b$ für die direkte Folgebeziehung) dargestellt.

Beim *TP-Sequence*-Algorithmus wird eine Quasi-Sequenz (*Quasi-Sequence*) als Repräsentation für den *Temporal Graph* verwendet. Dabei ist eine Quasi-Sequenz eine Sequenz von *Itemsets*.

5.6.3 *w-find* und *c-find*

In [37, 36, 38] stellen Greco et al. zwei Algorithmen, *w-find* und *c-find*, vor, um häufige Muster aus Verlaufsdaten zu extrahieren. Im Unterschied zu den Ansätzen von Hwang et al.

im vorhergehenden Abschnitt gehen Greco et al. von einem bereits bekannten Prozessmodell aus. Folglich kann dieses zusätzliche Wissen verwendet werden, um Muster abzuleiten.

Das Prozessmodell wird als gerichteter azyklischer Graph dargestellt. Kausale Beziehungen zwischen Aktivitäten werden durch Kanten repräsentiert. Die Semantik von Joins und Splits wird durch Funktionen der Aktivitäten, die Bedingungen an Ein- und Ausgangskanten darstellen, modelliert.

Patterns, die mit Hilfe der im folgenden vorgestellten Algorithmen gefunden werden sollen, sind Subgraphen des Worfklow-Schemas. Die Autoren konzentrieren sich auf das Auffinden von zusammenhängenden Patterns die *ws-closed* sind. *ws-closed* bedeutet dabei den deterministischen Abschluss eines Graphen. Die Idee, die hinter dem deterministischen Abschluss steckt, ist einfach. Wenn eine Aktivität a häufig im Log vorkommt, so müssen folglich alle Aktivitäten, die vor a ausgeführt werden müssen und die mit a über eine AND-Join-Semantik verbunden sind, ebenfalls ausgeführt worden sein. Daher müssen diese Aktivitäten mindestens so häufig im Log vorkommen wie Aktivität a selbst. Analoges gilt für AND-Splits.

Die von Greco et al. vorgestellten Algorithmen stellen eine Adaption des *Apriori*-Algorithmus (vgl. Listing 5.1) dar. Beim $w\text{-}find$-Algorithmus wird zunächst, basierend auf dem bereits bekannten Prozessmodell, eine Menge von elementaren *Weak Patterns* erzeugt, deren Häufigkeit im Workflow-Log den Schwellenwert überschreitet. Elementare *Weak Patterns* sind Teilgraphen des Schemas, die durch den deterministischen Abschluss einer einzelnen Aktivität transitiv gebildet werden. Dies ist möglich, da das Workflow-Schema bekannt ist. Diese Menge der häufigen elementaren Patterns entspricht L_1 aus dem *Apriori*-Algorithmus. Darüber hinaus wird eine Menge von häufigen Kanten initialisiert, die nicht mit Aktivitäten in einer AND-Join- oder AND-Split-Semantik verbunden sind.

In weiteren Schritten wird die Menge der Kandidaten (C_{k+1}) gebildet, indem die gefundenen Patterns erweitert werden. Eine Erweiterung der bestehenden Patterns erfolgt durch das Einfügen häufiger Kanten in das Pattern sowie durch Kombination gefundener Patterns mit einem Pattern aus der Menge der häufigen elementaren Patterns (L_1). Das Einfügen der Kanten erfolgt, da nur für Kanten die mit einer AND-Semantik assoziiert werden, sicher ist, dass sie aufgrund des transitiven Abschlusses im Muster enthalten sind.

Der $c\text{-}find$-Algorithmus läuft im wesentlichen analog zum $w\text{-}find$-Algorithmus ab. Der wesentliche Unterschied ist, dass Kandidaten durch das Verknüpfen zweier gefundener Patterns erzeugt werden. Da das Prozessmodell bekannt ist, kann die Information des Modells dazu verwendet werden, zu bestimmen, welche Patterns sich verknüpfen lassen.

In Experimenten auf synthetischen Daten stellten Greco et al. fest, dass der $c\text{-}find$-Algorithmus für größere Datenmengen eine bessere Performanz zeigt. Der Vorteil von $c\text{-}find$ ist, dass er i.d.R. weniger Kandidaten erzeugt als der $w\text{-}find$-Algorithmus.

5.6.4 Zusammenfassung und Diskussion

Die drei vorgestellten Ansätze haben gemeinsam, dass sie im wesentlichen bekannte Algorithmen verwenden. Diese werden an den Workflow-Kontext angepasst. Zum einen, indem passende Repräsentationsformen gefunden werden, zum anderen, indem domänenspezifisches Wissen verwendet wird. Allerdings verfolgen die vorgestellten Ansätze sehr unterschiedliche Ziele.

Während der Ansatz *WorkflowMiner* das Ziel hat, mit Workflow Patterns auch Kontrollflusskonstrukte aus Verlaufsdaten zu finden, geht es bei den Algorithmen *TP-Graph*, *TP-Itemset* und *TP-Sequence* darum, temporäre Beziehungen zu finden, jedoch ohne dabei Kontrollflusskonstrukte zu berücksichtigen. Für viele Einsatzszenarien ist letzteres ein großes Defizit.

Die Algorithmen $w\text{-}find$ und $c\text{-}find$ wiederum setzen ein bereits bekanntes Prozessmodell voraus. Da dieses direkt verwendet wird, um häufige Muster zu finden, sind diese Algorithmen für das Mining von unbekannten Prozessen ungeeignet. Als ein mögliches Anwendungsgebiet ihrer Ansätze nennen Greco et al. den Bereich der Prozess-Überwachung. Da $w\text{-}find$ und $c\text{-}find$ jedoch eher ungeeignet sind, um Abweichungen vom Prozessmodell festzustellen, wird die Menge der sinnvollen Anwendungsmöglichkeiten dieser Ansätze stark eingeschränkt.

Der Ansatz von Gaaloul et al. scheint vielversprechend zu sein. Allerdings fehlen noch häufig vorkommende Strukturen wie Zyklen. Um den Ansatz aber wirklich bewerten zu können, sind die Informationen in den Arbeiten [33, 32] zu knapp. Gaaloul et al. heben hervor, dass die in [33, 32] vorgestellten Ansätze erst der Anfang ihrer Arbeit darstellen, eine Weiterentwicklung dieser Ansätze ist daher zu erwarten.

5.7 Mining von Prozessmodellen

Im folgenden stellen wir Ansätze vor, um komplette Prozessmodelle aus den Verlaufsdaten abzuleiten. Im ersten Abschnitt werden Ansätze vorgestellt, die Techniken der FSM-Synthese (siehe Abschnitt 5.4.1) adaptieren. Ansätze, die einen gerichteten Graphen zur Darstellung des Prozessmodells verwenden, werden in Abschnitt 5.7.2 vorgestellt. In Abschnitt 5.7.3 gehen wir auf den α-, den α^+- sowie den β-Algorithmus ein. Diese drei Ansätze haben denselben prinzipiellen Ablauf und gehen sehr lokal vor, um ein Prozessmodell abzuleiten. LittleThumb, ein heuristischer Ansatz, wird in Abschnitt 5.7.4 vorgestellt. Im Anschluss daran wird in Abschnitt 5.7.5 ein Ansatz erläutert, der speziell auf blockstrukturierte Prozesse zugeschnitten ist. *Multi-Phase Process Mining*, ein sehr neuer Ansatz, der auf der Idee der Aggregation von Instanzgraphen basiert, wird in Abschnitt 5.7.6 vorgestellt. In Abschnitt 5.7.7 stellen wir einen induktiven Lernansatz vor. Ein genetisches Verfahren wird in Abschnitt 5.7.8 erläutert.

5.7.1 Ansätze aus der FSM-Synthese

In [18] stellen Cook und Wolf drei Ansätze, *Rnet*, *Ktail* und *Markov*, vor, um sequentielle Prozesse aus ereignisbasierten Daten abzuleiten. *Rnet* ist ein rein stochastischer Ansatz, der auf neuronalen Netzen basiert. Da dieser Ansatz allerdings von den Autoren selbst als zu unausgereift bewertet wird, verzichten wir auf genauere Erläuterungen zu *Rnet*.

Datta stellt in seiner Arbeit in [22] zwei Ansätze vor, die nahezu identisch sind mit *Ktail* und *Markov*. Darum gehen wir hier nur auf die Ansätze von Cook und Wolf genauer ein. Die Arbeit von Cook und Wolf entstand im Kontext von Software-Prozessen. Ihr Ziel ist es nicht, ein komplettes Prozessmodell abzuleiten, sondern vielmehr die Prozessausführung mit dem abgeleiteten Modell zu beschreiben.

5.7.1.1 Ktail

Da von sequentiellen Prozessen ausgegangen wird, können Ansätze aus der FSM-Synthese angepasst werden. Cook und Wolf modellieren den Prozess als endlichen Zustandsautomaten, wobei Aktivitäten durch Kanten zwischen den Zuständen dargestellt werden.

Ktail ist eine Adaption des Biermann-Feldmann-Algorithmus [18] (B-F-Algorithmus), einem Algorithmus aus der FSM-Synthese. Die Grundidee des B-F-Algorithmus ist, dass jeder Zustand in einem Prozess dadurch definiert wird, welches zukünftige Verhalten von diesem Zustand aus erzeugt werden kann. Ein endlicher Zustandsautomat wird erzeugt, indem Äquivalenzklassen gebildet werden. Eine Äquivalenzklasse bezeichnet eine Menge von *Histories* (Prefixen), die die gleiche k-elementige Zukunft haben, d.h., es wird um k Aktivitäten in die Zukunft geschaut. Die Menge der Äquivalenzklassen bildet die Menge der Zustände. Jede *History* h wird mit einer Aktivität q konkateniert (hq). Falls hq in einer Äquivalenzklasse C vorkommt, wird eine Kante zwischen der Äquivalenzklasse von h und der Äquivalenzklasse C erzeugt. Um mit Rauschdaten umgehen zu können, schlagen Cook und Wolf in [18] vor, Äquivalenzklassen mit wenigen Mitgliedern zu ignorieren.

5.7.1.2 Markov

Markov ist ein hybrider Ansatz, der stochastisches und algorithmisches Vorgehen vereinigt. Das Konzept von Markov-Ketten [18] wird verwendet, um die wahrscheinlichsten Ereignissequenzen festzustellen. Algorithmisches Vorgehen wird verwendet, um die Wahrscheinlichkeiten in einen Automaten zu transformieren.

Für eine Markov-Kette n-ter Ordnung gilt, dass die Wahrscheinlichkeit eines Zustands nur von den n vorangegangenen Zuständen abhängig ist. Für den *Markov*-Ansatz wird eine Abhängigkeitstabelle erstellt. Für Sequenzen der Länge n und kleiner wird in der Tabelle

festgehalten, wie wahrscheinlich Folgeaktivitäten sind. Für jede Aktivität wird ein Knoten in einem *Activity Graph* erzeugt. Kanten werden zwischen Aktivitäten einer Sequenz erzeugt, wenn die Wahrscheinlichkeit und die Häufigkeit dieser Sequenz vorgegebene Schwellenwerte überschreiten. Um zu verhindern, dass illegale Sequenzen durch die erzeugten Kanten möglich sind, werden Knoten gespalten. Im Anschluss daran kann anhand des *Activity Graphs* ein Zustandsautomat erzeugt werden. Die Kanten im *Activity Graph* stellen Knoten des Automaten dar, während die Knoten im *Activity Graph* die Kanten im resultierenden Automaten bilden.

In [16, 19] stellen Cook et al. vier Metriken vor, um mit ihren Ansätzen, insbesondere *Markov*, den sie als am aussichtsreichsten betrachten, auch parallele Abläufe erkennen zu können. Als Metriken werden Entropie, Häufigkeit, Periodizität sowie eine Kausalitätsmetrik eingeführt. Da die Erklärung jeder Metrik den Rahmen der Arbeit sprengen würde, gehen wir nur auf die Entropie als Beispiel ein. Die Idee dahinter ist, dass falls einer Aktivität a stets die Aktivität b folgt, so ist der Nachfolger von a deterministisch. Die Entropie von a, also der Informationsgehalt von a, ist dementsprechend 0. Um festzustellen ob eine Aktivität eine Verzweigung darstellt, wird daher die ermittelte Entropie mit maximalen Entropie-Werten von Verzweigungen verglichen. Die Entropie einer Verzweigung in zwei Pfaden bei fünf möglichen Aktivitäten beträgt beispielsweise 0,39. Es ist hervorzuheben, dass Cook und Wolf mit den in [16, 19] vorgestellten Metriken nicht unbedingt ein vollständiges Modell sondern vielmehr den groben Ablauf des Prozesses erfassen wollen, um ein besseres Verständnis vom Prozessablauf zu ermöglichen. Für detailliertere Informationen verweisen wir auf die Arbeiten von Cook und Wolf [18, 16] sowie die Arbeit von Datta [22].

5.7.2 Ableitung von gerichteten Graphen

Die drei in diesem Abschnitt vorgestellten Ansätze verwenden zur Modellierung des Prozessmodells gerichtete Graphen. Damit orientiert sich Prozessmodellierungssprache an IBM Flowmark[5].

Aktivitäten werden durch Knoten, kausale Beziehungen zwischen Aktivitäten durch Kanten modelliert. Eine boolesche Funktion, die den Kanten zugeordnet ist, bestimmt nach der Ausführung einer Aktivität, welche Folgeaktivitäten ausgeführt werden können. Aus dieser Modellierung des Prozessmodells ergibt sich, dass der Charakter von Splits und Joins (z.B. AND oder OR) nicht ermittelt werden muss.

Die Aufgabenstellung liegt daher darin, einen gerichteten Graphen abzuleiten, der konsistent mit den Verlaufsdaten ist, d.h. die Log-Daten lassen sich mit dem Graphen erzeugen.

Neben der Repräsentation des Prozessmodells sind sich die im folgenden vorgestellten Ansätze auch vom prinzipiellen Ablauf her sehr ähnlich. Es wird ein Graph, der den Prozess

[5]Der aktuelle Nachfolger des WfMS Flowmark von IBM ist MQWorkflow [50, 49].

modelliert, generiert, indem Graphen oder auch Kanten[6] für die einzelnen Spuren erzeugt und nach bestimmten Kriterien, um voneinander unabhängige Aktivitäten zu erkennen, zusammengefasst werden.[7].

5.7.2.1 Ansatz nach Agrawal et al.

In [4, 5] stellen Agrawal et al. einen Ansatz vor, der sich zunächst auf zyklenfreie Graphen beschränkt. Ein gerichteter Graph $G =< V, E >$, wobei V der Menge aller Aktivitäten im Log entspricht und E eine Menge von Kanten darstellt, wird generiert, indem für jede Folgebeziehung zweier Aktivitäten bezüglich der Ereignisspur eine Kante in E erzeugt wird. Eine Aktivität b folgt einer Aktivität a, falls b beginnt, nachdem a beendet ist, oder b einer Aktivität c folgt, welche auf a folgt. Die so erzeugten Kanten stellen mögliche kausale Beziehungen und damit Kanten im resultierenden Prozessgraphen dar.

Da bisher nur jede Spur und damit jede Instanz isoliert betrachtet wurde, können in G Kanten zwischen Aktivitäten enthalten sein, die voneinander unabhängig sind. Um diese Kanten zu entfernen werden aus E alle Kanten entfernt, die in beiden Richtungen vorkommen oder zu einem stark zusammenhängenden Teilgraphen[8] gehören. Da das dem Log zugrundeliegende Modell zyklenfrei sein soll, sind diese Zyklen ein Zeichen dafür, dass die Aktivitäten unabhängig voneinander sind. Aus E werden im Anschluss Kanten entfernt, die für die Ausführung der Spuren nicht notwendig sind. Dies erfolgt, indem für jede Spur der induzierte Teilgraph in G gefunden wird und eine transitive Reduktion des Teilgraphen durchgeführt wird. Alle Kanten, die nicht in den transitiven Reduktionen der Spurgraphen enthalten sind, werden anschließend entfernt. Dies ist nachzuvollziehen, da durch die transitive Definition der Folgebeziehung auch für indirekte Folgebeziehungen Kanten erzeugt wurden. Die folgenden beiden Abbildungen veranschaulichen das Prinzip für das Log $\{< a, b, c, d, e, f >, < a, c, d, g, f >\}$. Abbildung 5.10 zeigt den zugehörigen Abhängigkeitsgraphen, und Abbildung 5.11 zeigt den resultierenden Graphen nach der transitiven Reduktion.

Um mit Zyklen umzugehen gehen Agrawal et al. vor wie in Abschnitt 5.3.4 bereits erwähnt. Mehrfache Vorkommen einer Aktivität werden in einem vorgelagerten Schritt durchnummeriert und wie unterschiedliche Aktivitäten behandelt. Nach der Ausführung des Verfahrens werden die mehrfachen Vorkommen der Aktivitäten wieder auf eine Aktivität abgebildet. Der Graph wird somit gefaltet. Diese Art mit Zyklen umzugehen ist jedoch problematisch. Bei Zy-

[6] Es müssen nicht unbedingt komplette Graphen für einzelne Spuren erzeugt werden. Es werden jedoch Folge- bzw. Abhängigkeitsbeziehungen für jede Spur erzeugt und zusammengefasst.

[7] Wir bitten hier, zu beachten, dass die in diesem Abschnitt vorgestellten Ansätze nicht die einzigen sind, die nach diesem Prinzip ablaufen. Da sie jedoch denselben Repräsentationsformalismus verwenden, fassen wir sie in einem Abschnitt zusammen. Auf weitere Ansätze, die ähnlich ablaufen gehen wir in Abschnitt 5.7.5 sowie Abschnitt 5.7.6 ein.

[8] Ein Graph ist stark zusammenhängend, wenn für je zwei Knoten i und j sowohl i von j als auch j von i aus erreichbar sind.

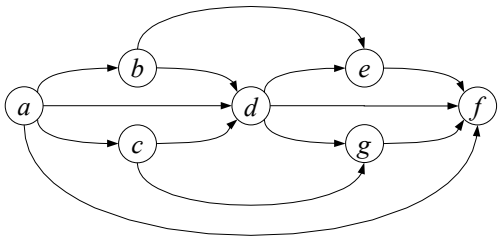

Abbildung 5.10: Abhängigkeitsgraph für das Beispiellog

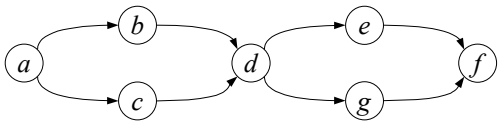

Abbildung 5.11: Resultierender Graph nach der transitiven Reduktion

klen, die in Kombination mit Verzweigungen auftreten funktioniert dies u.U. nicht. Dadurch kann der abgeleitete Graph falsche Kanten enthalten [48].

Anders als beim Ansatz von Agrawal et al. verwenden die beiden im folgenden vorgestellten Ansätze von Hwang und Yang [48] sowie Golani und Pinter [34, 35] Informationen über Ausführungsintervalle von Aktivitäten zur Aufdeckung von Parallelität bzw. von unabhängigen Aktivitäten.

5.7.2.2 Ansätze nach Hwang und Yang sowie Golani und Pinter

Da sich die Ansätze von Hwang und Yang sowie von Golani und Pinter sehr stark ähneln, beschränken wir uns darauf, den Ansatz von Hwang und Yang näher zu erläutern und wesentliche Unterschiede zum Ansatz von Golani und Pinter hervorzuheben. Der Ansatz von Hwang und Yang [48] ist vom Ablauf her ähnlich wie der Ansatz von Agrawal et al. Ein gerichteter Graph, der alle Spuren des Logs modelliert, wird abgeleitet, indem zunächst für jede Ereignisspur die Menge von Paaren, die in einer direkten Folgebeziehung bezüglich dieser Spur stehen (siehe Abschnitt 4), ermittelt werden. Dieser Schritt kommt einer Generierung von einzelnen Graphen für jede einzelne Spur gleich. Die Menge der direkten Folgebeziehungen in den Spuren stellen potentielle Kanten im resultierenden Prozessgraphen dar. Der resultierende Graph enthält daher alle Kanten, die durch die Folgebeziehungen der Spuren entstehen. Es werden anschließend alle Kanten zwischen Aktivitäten entfernt, die in mindestens einer Spur überlappt auftreten.

Der Ansatz von Golani und Pinter [34, 35] kann als eine Mischung der beiden oben vorgestellten Ansätze betrachtet werden. Das Vorgehen des Ansatzes entspricht im wesentlichen

dem Vorgehen von Hwang et al. Der wesentliche Unterschied besteht darin, dass Golani und Pinter in ihrer Arbeit zunächst nur die Ableitung von azyklischen Graphen vorsehen. Daher können sie sowohl zeitliche Überlappung von Aktivitäten als auch Verschränkung von Aktivitäten als Zeichen von Parallelität deuten, da eine Verschränkung von Aktivitäten zu einem Zyklus im resultierenden Modell führen würde. Nachdem ein Graph in ähnlicher Weise wie bei dem Ansatz von Hwang und Yang erzeugt wird, folgt bei dem Ansatz von Golani und Pinter daher ein Schritt, bei dem stark zusammenhängende Teile des Graphen entfernt werden. Ein wesentlicher Unterschied zwischen dem Ansatz von Agrawal et al. und dem von Golani und Pinter sowie Hwang und Yang ist, dass beim Ansatz von Agrawal et al. zunächst auch indirekte Folgebeziehungen berücksichtigt werden. Für Details verweisen wir auf die Arbeiten von Hwang und Yang [48] sowie Golani und Pinter [35, 34].

Für den Umgang mit Rauschdaten verwenden alle drei vorgestellten Ansätze Schwellenwerte, um die Relevanz von Folgebeziehungen zu bewerten.

5.7.3 Der α- , α^+- und β-Algorithmus

5.7.3.1 Der α-Algorithmus

Der α-Algorithmus [25] von van der Aalst et al. stellt eher eine formale Herangehensweise dar. Ziel von van der Aalst et al. war es, festzustellen, welche Klassen von Netzen korrekt rekonstruiert werden können, so dass das abgeleitete Prozessmodell dem ursprünglichen Prozessmodell entspricht.

Der α-Algorithmus geht rein algorithmisch vor und verwendet Petri-Netze zur Repräsentation des Prozessmodells. Der Algorithmus arbeitet auf Aktivitätsspuren und verwendet daher keine Ereignistypen, um auf Parallelität zu schließen. Er basiert auf vier verlaufsdatenbasierten Ordnungskriterien (*Log-based ordering relations)*. Die kausale Beziehung zwischen Aktivitäten ($a \rightarrow_W b$) wird geschlossen, wenn zwei Aktivitäten a und b mindestens einmal in der Sequenz ab im Log vorkommen, und die Sequenz ba nie auftritt.
Auf parallele Ausführung zweier Aktivitäten ($a \|_W b$) wird durch die Verschränkung dieser Aktivitäten geschlossen (siehe Abschnitt 5.3.2).
Die alternative Ausführung oder auch die Unabhängigkeit zweier Aktivitäten a und b ($a \#_W b$) wird geschlossen, wenn weder die Sequenz ab noch die Sequenz ba im Log vorkommt.

Auf Grundlage der genannten Ordnungskriterien generiert der α-Algorithmus ein S/T-Netz als Ausgabe. Da jede Aktivität, die im Log vorkommt, auch im Workflow-Schema auftauchen muss, wird im ersten Schritt für jede Aktivität, die nicht Start- oder Endaktivität ist, eine Transition in der Menge der Transitionen T_W erzeugt. Die Menge der Start und End-Aktivitäten der Spuren wird separat erfasst.
Stellen, die Beziehungen zwischen Aktivitäten darstellen, werden nur aufgrund der kausalen

Beziehung ($\rightarrow_W$) erzeugt. Um festzustellen, wie viele Stellen generiert werden müssen, wird die Menge X_W erzeugt, so dass für X_W folgendes gilt:

Die Transitionen in A (bzw. B) stehen jeweils in keiner kausalen Beziehung zueinander. Demzufolge gibt es auch keine Stelle, die diese Transitionen verbinden.

Mit Y_W, werden die größten Mengen A und B ausgewählt, da dadurch die Anzahl der Stellen korrekt bestimmt wird. Für zwei Mengen A und B wird eine Stelle erzeugt. Abbildung 5.12 verdeutlicht das Prinzip. Da in A nur eine Aktivität enthalten ist, während in B drei Aktivitäten enthalten sind, stellt das in Abbildung 5.12 dargestellte Konstrukt ein XOR-Split dar.

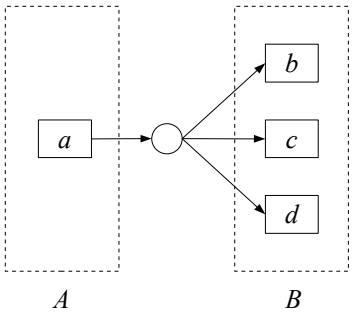

Abbildung 5.12: Generierung eines XOR-Splits mit den Mengen A und B

Für die Menge Y_W sowie die Start- und Endaktivitäten werden im Anschluss entsprechende Stellen generiert und durch Kanten verbunden. Van der Aalst et al. konnten beweisen, dass der α-Algorithmus in der Lage ist, alle SWF-Netze zu rekonstruieren, die keine Zyklen der Länge zwei oder eins beinhalten, wenn ein vollständiges Log bezüglich der direkten Folgebeziehung der Aktivitäten vorliegt. SWF-Netze sind eine Teilmenge von WF-Netzen[9], die keine Non-free-choice-Konstrukte sowie keine Synchronisation von OR-Joins beinhalten. Für weitere Informationen zu SWF-Netzen verweisen wir auf die Arbeit von van der Aalst et al. [88].

5.7.3.2 Der α^+-Algorithmus

Der α^+-Algorithmus ([26, 27]) ist eine Modifikation des α-Algorithmus. Er ist in der Lage, alle SWF-Netze, einschließlich jenen mit Zyklen der Länge eins oder zwei, korrekt abzuleiten. Da der α^+-Algorithmus vom Ablauf her dem α-Algorithmus entspricht, verzichten wir an dieser Stelle auf genauere Erläuterungen. Anzumerken ist, dass der α^+-Algorithmus die Ordnungskriterien modifiziert und ein anderes Verständnis von der Vollständigkeit des Logs erfordert.

[9]WF-Netze, auch Workflow-Netze genannt, sind eine Unterklasse der S/T-Netze. Sie wurden definiert, um Prozesse zu modellieren. Für Details verweisen wir auf die Arbeit von van der Aalst et al. [88]

Für detailliertere Informationen über den α- und den α^+-Algorithmus verweisen wir auf die Arbeiten [86, 25, 87, 88].

Da beide Ansätze rein algorithmisch vorgehen, sind sie dementsprechend sehr empfindlich gegenüber Rauschdaten. Darüber hinaus werden bei diesen Ansätzen sehr lokale Informationen verwendet. Daher können grundsätzlich keine kausalen Beziehungen zwischen zwei im Log nicht direkt benachbarten Aktivitäten, wie es z.B. bei Non-Free-Choice-Konstrukten der Fall sein kann, aufgedeckt werden.

Der α- und α^+-Algorithmus sind in der Anwendung EMiT implementiert. EMiT akzeptiert als Eingabe Log-Daten in einem XML-Format. Indem das Log durchgespielt wird, ist EMiT auch in der Lage, das Prozessmodell um Performanzwerte, wie z.B. Durchlaufzeiten, zu ergänzen. Darüber hinaus wurden die Algorithmen auch in dem Framework ProM integriert. ProM ist eine Anwendung, die von der Arbeitsgruppe um Prof. van der Aalst entwickelt wurde, um ein einheitliches Framework für die von ihnen entwickelten Ansätze bereitzustellen.

5.7.3.3 Der β-Algorithmus

Der β-Algorithmus von Wen et al. (vgl. [92]) kann als eine Modifikation des α-Algorithmus betrachtet werden. Als Ausgabe generiert der β -Algorithmus ebenfalls ein S/T-Netz. Im Unterschied zum α- bzw. α^+-Algorithmus, die auf einer Spur von Aktivitäten arbeiten, arbeitet der β-Algorithmus auf Ereignisspuren und berücksichtigt Start- und Endereignisse von Aktivitäten. Damit ist es ihm möglich, parallele Abläufe aufgrund von zeitlicher Überlappung direkt zu erkennen (siehe auch Abschnitt 5.3.2). Der β-Algorithmus basiert auf Ordnungskriterien, die ähnlich zu denen vom α- bzw. α^+-Algorithmus sind. Der entscheidende Unterschied liegt darin, dass parallele Abläufe nun nicht anhand von Verschränkungen der Aktivitäten sondern über ihre zeitliche Überlappung erkannt werden. Der Ablauf des β-Algorithmus ist fast identisch mit dem Ablauf der im letzten Abschnitt beschriebenen Algorithmen. Daher gehen wir an dieser Stelle nicht weiter darauf ein. Für detailliertere Informationen verweisen wir auf die Arbeit von Wen et al. [92]. Eine Implementierung des β-Algorithmus als Plug-In ist in dem Framework ProM integriert.

Durch die Aufdeckung von Parallelität durch zeitliche Überlappung hat der β-Algorithmus größere Chancen, mit Situationen von parallelen Pfaden mit Aktivitäten unterschiedlicher Ausführungszeiten, wie in Abschnitt 5.3.2 beschrieben, umzugehen. Auf der anderen Seite besteht der Nachteil des β-Algorithmus gegenüber dem α- bzw. dem α^+-Algorithmus darin, dass er ganz auf entsprechende Start- und Endereignisse von Aktivitäten angewiesen ist. In Umgebungen, in denen nur das Endereignis einer Aktivität protokolliert wird, kann der β-Algorithmus daher nicht angewendet werden. Wie der α- und der α^+-Algorithmus ist auch der β-Algorithmus nicht robust gegenüber Rauschdaten, da jedes Vorkommen einer Aktivität ausschlaggebend ist. Für einen Umgang mit Rauschdaten verweisen sowohl van der Aalst et

al. als auch Wen et al. daher auf den im nächsten Abschnitt vorgestellten Ansatz LittleThumb.

5.7.4 LittleThumb - Ein heuristischer Ansatz

Der im folgenden vorgestellte Ansatz LittleThumb von Weijters et al. ([3, 1, 2]) geht heuristisch vor. Der Ablauf erfolgt in drei Schritten: Konstruktion einer *Dependency/Frequency Table*, Generierung eines *Dependency/Frequency Graphs* (D/F-Graph) und Konstruktion eines WF-Netzes aus dem D/F-Graph.

Die *Dependency/Frequency Table* wird im ersten Schritt erzeugt, indem für jede Aktivität a neben ihrer eigenen Häufigkeit und der Häufigkeit mit der a direkt vor und direkt nach einer Aktivität b auftritt eine lokale und eine globale Metrik verzeichnet wird. Die lokale Metrik bezieht sich auf direkte Nachfolgebeziehungen, während die globale Metrik auch indirekte Nachfolgebeziehungen berücksichtigt. Allerdings ist ein Zerfallsfaktor in der globalen Metrik integriert, so dass die Bedeutung der indirekten Folgebeziehung mit zunehmender Distanz zwischen zwei Aktivitäten abnimmt. Zweck dieser Metriken ist es, ein Maß darzustellen, um eine Bewertung zu ermöglichen, wie stark zwei Aktivitäten von einander abhängen.

Der D/F-Graph wird im zweiten Schritt erzeugt, indem einfache Regeln angewendet werden. Ein *Dependency Score*, der sich aus der globalen und der lokalen Metrik berechnen lässt, wird dazu eingesetzt.
Da alle Aktivitäten, ausgenommen die Start- und die Endaktivität, mindestens eine Vorgänger- und eine Nachfolgeraktivität haben müssen, ist die Idee, kausale Beziehungen zwischen Paaren von Aktivitäten zu schließen, zwischen denen der *Dependency Score* am höchsten ausfällt. Durch die Anwendung dieser Regel kann bereits ein Kontrollflussgraph abgeleitet werden. Allerdings ist diese Regel noch zu einfach, um auch komplexere Kontrollflusskonstrukte sowie kurze Zyklen, aufzudecken. Der prinzipielle Gedanke ist jedoch anhand dieser Regel gut nachvollziehbar. Für detaillierte Informationen zu den Regeln, insbesondere auch Modifikationen der vorgestellten Regel, die auf Erfahrungswerte basieren, verweisen wir auf die Arbeit [3].

Die Ableitung der Semantik von Joins und Splits erfolgt im dritten Schritt. Dazu können die Werte in der D/F-Tabelle verwendet werden, ebenso wie die Häufigkeiten von Aktivitäten. Der α-Algorithmus kann anschließend verwendet werden, um ein WF-Netz zu generieren.

Der heuristische Ansatz der Arbeitsgruppe um Prof. van der Aalst ist in der Anwendung namens LittleThumb implementiert. LittleThumb unterscheidet zwischen AND-/OR-Joins und AND-/OR-Splits mit Hilfe der D/F-Tabelle. Darüber hinaus verwendet LittleThumb die Häufigkeitsinformation von Aktivitäten, um das erzeugte Modell zu überprüfen.

LittleThumb ist, aufgrund des heuristischen Vorgehens, weniger anfällig gegenüber Rauschdaten. In Experimenten mit synthetischen Daten lieferte LittleThumb gute Ergebnisse. Auch in Experimenten mit realitätsnahen Prozessen, konnten gute Ergebnisse erzielt werden. Die

Autoren weisen jedoch darauf hin, dass komplexe Kontrollflussstrukturen in Verbindung mit kurzen Zyklen noch eine größere Fehlerquelle darstellen und ihre Regeln noch weiterer Verbesserungen bedürfen.

5.7.5 ProcessMiner - Ein Ansatz für blockstrukturierte Prozesse

Der in diesem Abschnitt vorgestellte Ansatz ist den Ansätzen aus Abschnitt 5.7.2.1 vom Ablauf her ähnlich. Ein Modell wird generiert, indem Modelle für Instanzen zu Cluster zusammenfasst werden. Da der Ansatz von Schimm [77, 75, 76, 74], im Unterschied zu den in Abschnitt 5.7.2.1 vorgestellten Ansätzen, anstelle von gerichteten Graphen blockstrukturierte Workflow-Modelle verwendet, gehen wir auf den Ansatz von Schimm im folgenden separat ein.

Zunächst werden diejenigen Spuren mit derselben Menge an Aktivitäten und derselben Menge als Folgebeziehungen bezüglich der Ereignisspuren zusammengefasst.

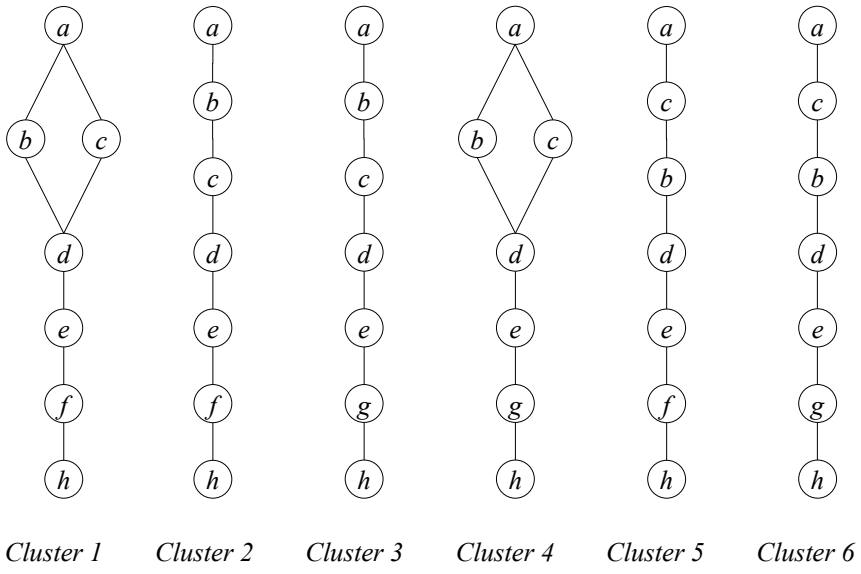

Abbildung 5.13: Sechs Cluster eines Beispiellogs

Abbildung 5.13 zeigt sechs Cluster eines Beispiellogs. Das Beispiel, das wir hier verwenden, ist der Arbeit von Schimm [77] entnommen. Transitive Kanten wurden in der Abbildung ausgelassen.

Um auf Parallelität, also die Unabhängigkeit von Aktivitäten, zu schließen, werden Cluster zusammengefasst. Cluster 1 und Cluster 2 haben dieselbe Menge an Aktivitäten und unterscheiden sich lediglich in der Folgebeziehung zwischen b und c. Da b und c in Cluster 1 unabhängig sind, d.h. sie überlappen sich in ihrer Ausführungszeit, ist die Abhängigkeit

zwischen b und c in Cluster 2 keine echte Abhängigkeit. Folglich werden Cluster 1 und Cluster 2 zusammengefasst. Analog werden auch Cluster 1 und Cluster 5 zusammengefasst sowie Cluster 3, Cluster 4 und Cluster 6.
An dieser Stelle ist noch anzumerken, dass die Unabhängigkeit zweier Aktivitäten über ihre Überlappung festgestellt wird. Ohne Cluster 1 würden Cluster 2 und Cluster 5 nicht zusammengefasst werden, obwohl b und c in den Clustern 2 und 5 verschränkt vorkommen.

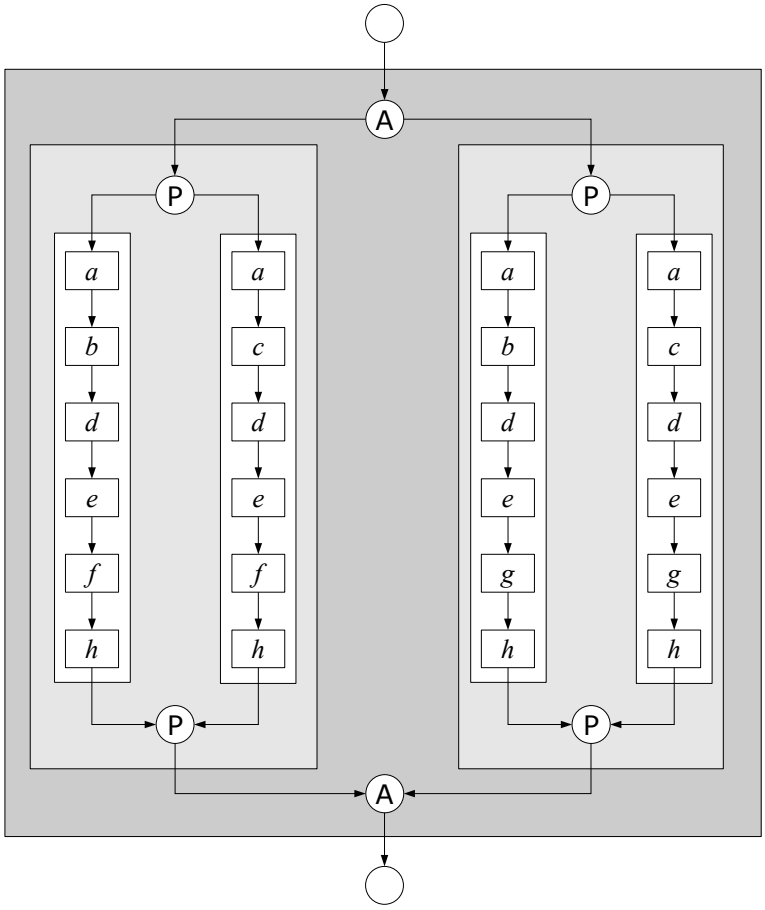

Abbildung 5.14: Ein disjunktives Prozessmodell

Um ein Modell zu erhalten, welches das Log-Verhalten modelliert, wird ein disjunktives Modell erzeugt. Das disjunktive Modell fasst Modelle für jeden Pfad eines Clusters in einer alternativen Verzweigung zusammen. Abbildung 5.14 veranschaulicht dies für die Beispielcluster aus Abbildung 5.13. Für jeden möglichen Ausführungspfad im Log ist ein Pfad im disjunktiven Modell enthalten. Die Knoten werden als Zeiger auf Aktivitäten interpretiert.

Da nur blockstrukturierte Prozesse zugelassen werden, können diese auch als Term dargestellt werden. Die Termdarstellung für das disjunktive Modell aus Abbildung 5.14 sieht

wie folgt aus: $A(P(S(a,b,d,e,f,h),S(a,c,d,e,f,h)),P(S(a,b,d,e,g,h),S(a,c,d,e,g,h)))$ A steht dabei für eine Alternative Ausführung, P für eine parallele und S für eine Sequenz. Auf diese Termdarstellung des Modells werden im Anschluss Termumformungsregeln (*Term Rewriting Rules*) angewendet. Mit Hilfe der Umformungsregeln, die auch Workflow-Algebra genannt werden, können alternative und parallele Verzweigung nach innen gebracht werden. Eine Beispielregel lautet wie folgt:

$$P(S(b_t,b_1),...,S(b_t,b_n)) \rightarrow S(b_t,P(b_t,...,b_n))$$

Diese Regel kann beispielsweise auf die linke oder rechte parallele Verzweigung angewendet werden, um diese nach innen zu bringen.

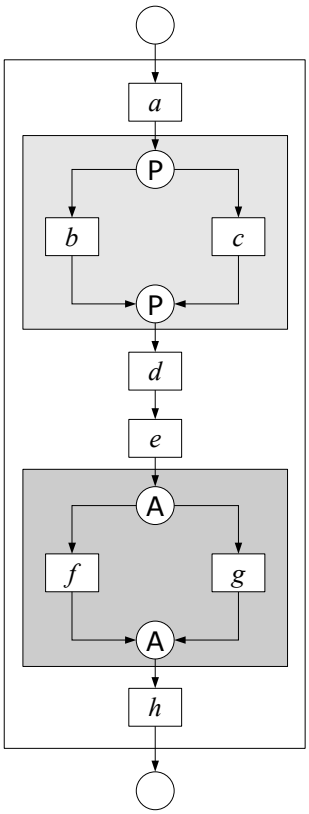

Abbildung 5.15: Das resultierende Prozessmodell

Zu bemerken ist, dass die Anwendung der Termumformungsregeln semantikerhaltend ist. Um mit Zyklen umzugehen, werden diese in einem ersten Schritt herausgefiltert und durch Start- und Endereignisse für Subworkflows ersetzt. So können die Zyklen wie normale Prozesse behandelt und später wieder eingefügt werden. Bei dem Ansatz von Schimm sind allerdings keine Mechanismen vorgesehen, um mit Rauschdaten umzugehen. Ziel dieses Ansatzes ist es, mit dem abgeleiteten Modell exakt die Menge der Spuren im Log abzudecken.

Der blockstrukturbasierte Ansatz wurde in der Anwendung *Process Mining Workbench* implementiert. Das von *Process Mining Workbench* erzeugte Prozessmodell kann in WPD-L/XPDL exportiert werden [77].
Darüber hinaus ist auch die Anwendung QuaxMap verfügbar, die Log-Daten von IBM MQ Series WfMS in ein passendes XML-Format als Eingabe für *Process Mining Workbench* abbilden kann.

5.7.6 Multi-Phase Process Mining

In [89] stellen van Dongen und van der Aalst einen Ansatz vor, der auf der Idee basiert, ein Prozessmodell zu erzeugen, indem Modelle für einzelne Instanzen (*Instance Graphs*) abgeleitet und aggregiert werden. Damit weist dieser Ansatz Ähnlichkeiten zum Ansatz von Schimm auf. Da zu der Bearbeitungszeit dieser Diplomarbeit erst der erste Schritt des Mining-Prozesses, also die Generierung der Instanzgraphen, publiziert wurde, befassen wir uns im folgenden ausschließlich mit diesem Aspekt.

Die grundlegende Idee ist, die gesamten Verlaufsdaten zu verwenden, um kausale Beziehungen ($\rightarrow_W$) zwischen den Aktivitäten zu finden. Die Beziehung $\rightarrow_W$ ist dabei ähnlich definiert wie die für den α^+-Algorithmus (siehe Abschnitt 5.7.3), weshalb wir darauf nicht weiter eingehen werden. Im Gegensatz zu Prozessmodellen enthalten Instanzgraphen keine Oder-Verzweigungen, da sie jeweils nur eine einzelne Instanz modellieren. Auch Schleifen sind in Instanzgraphen nur in abgerollter Form enthalten.
Da Aktivitäten mehrfach vorkommen können, wird für die Erzeugung der Instanzgraphen nicht direkt auf den Aktivitäten gearbeitet. Stattdessen werden Abbildungen der Indize der einzelnen Spuren auf die Aktivitäten verwendet. Mit σ_i wird die i-te Aktivität der Spur σ bezeichnet. Die kausalen Beziehungen werden, so weit möglich, mit Hilfe der Abbildung σ_i auf eine Aktivitätsspur angewandt, um einen Instanzgraphen zu erzeugen.

Ein Ziel, das van Dongen und van der Aalst mit diesem Ansatz verfolgen, ist die Umwandlung der Instanzgraphen in Ereignis-Prozess-Ketten, um eine weitere Analyse im *Aris Process Performance Monitor* (Aris PPM) zu ermöglichen. Mit Aris PPM soll es auch möglich sein, die Instanzgraphen zu aggregieren [89].

5.7.7 InWoLvE - Ein induktiver Ansatz

In [42] stellt Herbst einen Ansatz vor, der auf Techniken aus dem Bereich des Maschinellen Lernens basiert.
Die Ableitung eines Prozessmodells erfolgt dabei in zwei Schritten. Im ersten Schritt, dem Induktionsschritt, wird ein Graph induziert. Im zweiten Schritt, dem Transformationsschritt, wird der Graph in ein ADONIS Workflow-Modell umgewandelt, welches mit der Prozessmo-

dellierungssprache ADL (*ADONIS Description Language*) von ADONIS beschrieben ist. Da der Transformationsschritt aus der Sicht von *Process Mining* weniger interessant ist, beschränken wir uns im folgenden auf die Erläuterung des Induktionsschritts. Der Induktionsalgorithmus entspricht im wesentlichen einem Graphgenerierungsalgorithmus, welcher in einer Suchfunktion[10] eingebettet ist [42, 44].
Zur Repräsentation des Prozessmodells im Induktionsschritt verwendet Herbst einen stochastischen Aktivitätsgraphen (*Stochastic Activity Graph*), kurz *SAG*. Den Kanten eines SAG ist jeweils eine Übergangswahrscheinlichkeit zugeordnet.

Da der Ansatz von Herbst das Ziel verfolgt, mit einer nicht-injektiven Aktivitätszuordnungsfunktion sowie mit zyklischen Abläufen umgehen zu können, können Aktivitäten mehrfach in einer Spur auftreten. Darüber hinaus können unterschiedliche Knoten des Prozessmodells derselben Aktivität zugeordnet sein.

Um die Aktivitätszuordnungsfunktion zu lernen, wird Suche verwendet. Der Suchraum ist durch die Abbildungen zwischen Aktivitäten der Instanzen des Logs und den Knoten im Modell gegeben. Der Suchalgorithmus beginnt mit dem allgemeinsten Modell, d.h. alle gleichnamigen Instanzen werden auf die gleiche Aktivität abgebildet, und sucht top-down nach einer passenden Abbildung.

Die heuristische Suche (*Beam Search*), die eingesetzt wird, verwendet zur Steuerung durch den Suchraum eine Approximation des Log-Likelihoods. Dieser wird mit Hilfe der Übergangswahrscheinlichkeiten approximiert und kann als Parameter vom Benutzer eingestellt werden.

Das Modell wird spezialisiert, indem ein Split-Operator angewendet wird. Dieser führt zu einer Aufteilung der Aktivitäten in den Instanzen im Log, die einem Knoten im Modell zugeordnet werden, indem diese in den Instanzen umbenannt werden.
Der Graphgenerierungsalgorithmus, der dem Algorithmus von Agrawal (siehe Abschnitt 5.7.2.1) im wesentlichen sehr ähnelt, wird dann aufgerufen und erzeugt für die Instanzen einen entsprechenden Aktivitätsgraphen.

Auch für den Umgang mit Rauschdaten sieht der Ansatz von Herbst Mechanismen vor. Im besonderen handelt es sich dabei um einen speziellen Split-Operator. Damit werden Knoten, die Ausnahmefälle widerspiegeln, von den übrigen Knoten separiert [42]. So wird eine künstliche Unterscheidung der Knoten herbeigeführt. Das von Herbst abgeleitete Modell ist damit im Gegensatz zu dem Ansatz von Agrawal in der Lage, den Ausnahmefall im abgeleiteten Prozessmodell zu erfassen und auch als einen Ausnahmefall kenntlich zu machen, da die Übergangswahrscheinlichkeiten im SAG erfasst werden.
Für Details zum Induktionsschritt sowie zur Transformation des SAG in ein ADONIS-Modell,

[10] Für einen Überblick über Suchstrategien und Suchverfahren verweisen wir auf eine Arbeit von Russel und Norvig [66].

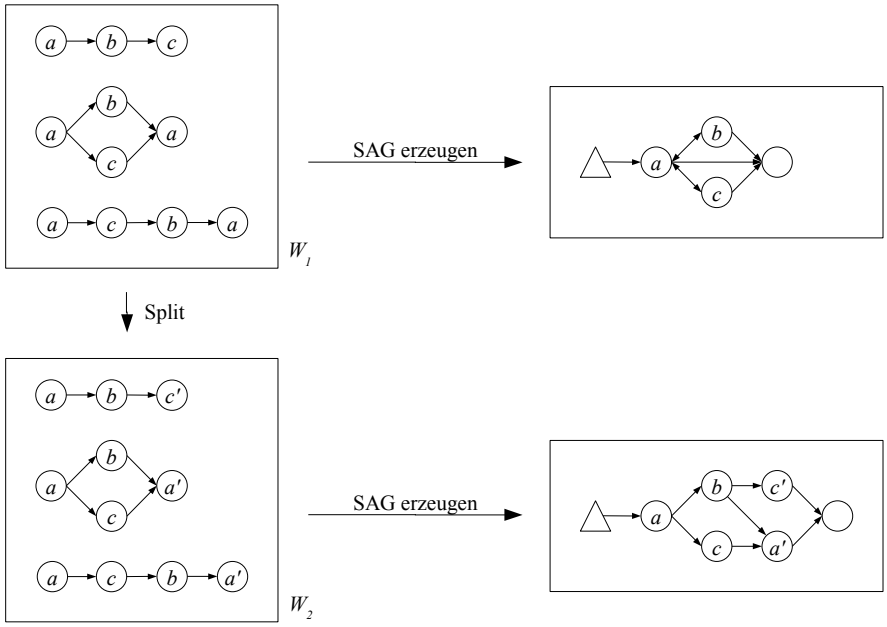

Abbildung 5.16: Die Anwendung des Split-Operators

verweisen wir auf die Dissertation von Herbst [42].

Eine prototypische Implementierung des induktiven Ansatzes liefert Herbst mit InWoLvE [43].
Hammori, der sich im Rahmen seiner Diplomarbeit mit dem Thema *Interactive Workflow Mining* beschäftigte, leitete aufgrund von Erfahrungen in der Anwendung von InWoLvE Anforderungen für ein neues System ab.
Es wurde festgestellt, dass der Mining-Prozess mit InWoLvE in hohem Maße interaktiv ist [39, 40]. Die Arbeit mit InWoLvE entspricht einem iterativen Prozess. In jeder Iteration erfolgen mehrere Schritte: die Anpassung der Parameter, die Visualisierung der Ergebnisse in ADONIS sowie die Evaluierung durch den Prozessexperten. Da InWoLvE für eine interaktive Arbeit in diesem Maße jedoch nicht zugeschnitten und daher ungeeignet ist, wurde die Anwendung ProTo entwickelt.
ProTo verwendet einen InWoLvE-Kern und bietet Funktionen zur Unterstützung des Prozessexperten bei der Evaluation des abgeleiteten Prozessmodells. Insbesondere werden Data-Mining-Techniken verwendet, um zu überprüfen, wie robust das abgeleitete Modell gegenüber zukünftige Verlaufsdaten ist. Darüber hinaus ist in ProTo auch eine Visualisierungskomponente integriert, so dass ADONIS nicht in jeder Iteration zur Darstellung des Prozessmodells geöffnet werden muss. Die Visualisierungskomponente versucht das abgeleitete Prozessmodell in einer neuen Iteration so darzustellen, dass Änderungen gegenüber dem Modell der letzten Iteration besser deutlich werden. Kleinere Veränderungen führen nicht zu einer komplett anderen Darstellung des Modells. Dadurch kann der Prozessexperte ein „mentales Modell“ des

Prozessmodells entwickeln und die wesentlich Züge besser erfassen.

5.7.8 Genetisches Control Flow Mining

In [83, 28] stellen van der Aalst, de Medeiros und Weijters einen genetischen Ansatz zur Ableitung eines Prozessmodells vor.
Genetische Algorithmen sind an die Evolutionstheorie angelehnt. Demzufolge setzen sich in einer Population immer die stärksten Individuen durch. Die Population ändert sich durch genetische Operationen, z.B. Mutationen oder Kreuzungen von Individuen. Im allgemeinen lässt sich der Ablauf eines genetischen Algorithmus wie folgt beschreiben.
Eine anfängliche Menge von Individuen wird erzeugt. Genetische Operationen werden darauf angewandt, bis in der Menge ein Individuum mit gewünschten Eigenschaften gefunden werden kann.

Im genetischen Ansatz von van der Aalst et al. entspricht jedes Individuum einem Prozessmodell, welches mehr oder weniger konsistent zu den Instanzen des Logs ist. Die Tauglichkeit (*Fitness*) eines Prozessmodells ist durch die Größe der Übereinstimmung zum Log gegeben. Als genetische Operationen werden Elitismus, Kreuzung und Mutation verwendet.

Eine Kernfrage bei der Verwendung eines genetischen Algorithmus für *Control Flow Mining* ist die Repräsentation des Prozessmodells. Van der Aalst et al. führen dazu eine *Causal Matrix* (CM) ein. Die CM lässt eine Matrixdarstellung kausaler Zusammenhänge zwischen Aktivitäten zu und erlaubt eine einfache boolesche Darstellung von AND/OR Semantik.
Wie eine CM in ein WF-Netz transformiert werden kann, erläutern van der Aalst et al. in [83, 28]. Wir wollen an dieser Stelle nicht genauer darauf eingehen. Wichtiger ist für uns das Verständnis, wie neue Prozessmodelle mittels genetischer Operationen erzeugt werden. Im folgenden gehen wir genauer darauf ein. Um die Auswirkungen der genetischen Operationen zu erklären, müssen wir zunächst die *Causal Matrix* erläutern.

In der Matrix wird der Index von Zeilen und Spalten jeweils auf die Aktivitäten des gesamten Logs abgebildet. Besteht ein kausaler Zusammenhang ($\rightarrow$) zwischen Aktivität a und Aktivität b, so wird in der Matrix in der entsprechenden Zelle eine 1 eingetragen. In Tabelle 5.1 bestehen folgende kausale Zusammenhänge: $a \rightarrow b$ und $a \rightarrow c$. Durch Joins bzw. Splits ergeben sich Zeilen bzw. Spalten mit mehr als einer 1.

Die Semantik von Joins und Splits, also ob es sich dabei z.B. um ein AND-Split handelt, wird durch boolesche Ausdrücke[11] in konjunktiver Normalform (KNF) beschrieben. Die Semantik von Splits bzw. Joins wird im Feld *Outputs* bzw. *Inputs* einer Aktivität festgehalten. *Outputs* bestimmt das Ausgangsverhalten einer Aktivität (Splits), *Inputs* bestimmt ihr Eingangsverhalten (Join). Abbildung 5.17 zeigt die Abbildung der CM auf ein Prozessmodell.

[11] Die booleschen Ausdrücke dürfen keine Negationen enthalten, da diese für den Prozesskontext auch nicht sinnvoll sind.

→	a	b	c	...	Outputs
a	0	1	1		$b \wedge c$
b	0	0	0		...
c	0	0	0		...
...	...	...	...	...	...
INPUTS	*true*	a	a	...	

Tabelle 5.1: Eine Causal Matrix

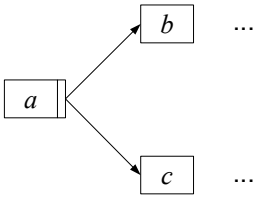

Abbildung 5.17: Das Prozessmodell für die Causal Matrix in Tabelle 5.1

Welche Aktivitäten in den booleschen Ausdrücken vorkommen dürfen, wird durch die kausalen Zusammenhänge bestimmt. Bei Änderungen müssen die booleschen Ausdrücke und die kausalen Zusammenhänge konsistent gehalten werden.

Für die Erzeugung der Anfangspopulation werden an LittleThumb (siehe Abschnitt 5.7.4) angelehnte Heuristiken im Zusammenhang mit einem zufällig bestimmten Schwellenwert verwendet, um kausale Beziehungen zwischen Aktivitäten festzulegen. Die Heuristiken verwenden zur Berechnung eines möglichen kausalen Zusammenhangs zweier Aktivitäten nur die Häufigkeit, wie oft diese Aktivitäten direkt benachbart im Log vorkommen. Damit weisen die Heuristiken einen starken lokalen Charakter auf. Für zwei Aktivitäten, die nie direkt benachbart im Log registriert wurden, ergibt sich ein Heuristikwert von 0.
Beziehungen zwischen Aktivitäten, die sich durch einen höheren Wert der Heuristik ausdrücken, haben grundsätzlich größere Chancen in der Anfangspopulation berücksichtigt zu werden. Die booleschen Ausdrücke in der Anfangspopulation werden rein zufällig generiert.

Zur Berechnung der Tauglichkeit (*Fitness*) eines Individuums wird jede Spur durchgespielt (*Token Game*). Die Tauglichkeit, eine Zahl zwischen 0 und 1, berechnet sich aus den Verhältnissen zwischen den erfolgreich geparsten und den gesamten Aktivitäten sowie zwischen erfolgreich durchgespielten Spuren und allen Spuren des Logs.

Der eigentliche Kern des genetischen Algorithmus wird durch die genetischen Operationen gebildet. Bei jeder Iteration wird auf Basis der alten Population eine neue generiert. Durch Elitismus werden die tauglichsten Individuen ausgewählt und in die neue Population übertragen. Kreuzungen dienen dazu, durch die Kombination von Eltern-Individuen aus der alten Population zwei neue Abkömmlinge zu erzeugen, mit der Hoffnung die tauglichen Teile

von beiden Elternteilen zu verschmelzen. Zusätzlich können durch Mutationen Änderungen an den Individuen durchgeführt werden, wodurch neue Aspekte erhalten werden. Darüber hinaus können Turniere verwendet werden: Aus der Menge der Individuen wird eine Untermenge gebildet, das tauglichste Individuum dieser Untermenge wird ausgewählt. Im folgenden gehen wir genauer auf die genetischen Operationen ein.

Bei Kreuzungen zweier Individuen werden die *Inputs* bzw. die *Outputs* einer einzelnen zufällig gewählten Aktivität vertauscht. Ergebnis der Kreuzung sind zwei Abkömmlinge, wobei für jeden Abkömmling ein Elternteil als Vorlage dient.
In der Menge der Konjunktionsglieder der booleschen Formel wird zufällig eine Stelle gewählt. Von dieser Stelle an werden die Konjunktionsglieder getauscht. Die Matrix muss evt. angepasst werden, damit die kausalen Zusammenhänge und die *Inputs* und *Outputs* konsistent sind.

Mutationen werden zufällig ausgeführt und wirken sich ausschließlich auf die Form der booleschen Ausdrücke aus, also die Semantik von Joins und Splits. Die kausalen Zusammenhänge werden dadurch nicht beeinflusst. Eine Mutation bewirkt, dass der boolesche Ausdruck auf zufälliger Basis neu gebildet wird.

Genetic Process Mining verspricht ein globaleres Vorgehen als die meisten in dieser Arbeit vorgestellten Ansätze, da Prozessmodelle immer gegen das vollständige Log geprüft werden. Allerdings wird für die Generierung der Anfangspopulation eine sehr lokale Heuristik verwendet, die nur direkt benachbarte Aktivitäten berücksichtigt. Daher erscheint es uns fraglich, ob eine kausale Beziehung zwischen zwei nicht direkt benachbarten Aktivitäten, wie es bei Non-Free-Choice-Konstrukten (siehe Abschnitt 5.3.5) der Fall ist, überhaupt in die Anfangspopulation übernommen wird.

In Experimenten wurde ermittelt, dass der genetische Ansatz mit Rauschdaten umgehen kann. Dies scheint sich jedoch hauptsächlich auf die Rauschdaten-Typen *Missing Head*, *Missing Tail* und bei kleineren Prozessen auch auf *Exchanged Activities* zu beziehen. Bei den wichtigen Rauschdaten-Typen *Missing Body*, *Missing Activity*, *Exchanged Activities* bei Versuchs-Prozessen ab 22 Aktivitäten und einer Mischung aus allen Typen sind die Ergebnisse leider enttäuschend. Dies führen die Autoren auf die zur Generierung der Anfangspopulation verwendete Heuristik zurück.

Eine weitere interessante Eigenschaft ist, dass sich die Ergebnisse bei zunehmender Prozessgröße verschlechtern. Vermutlich wird dies in Zusammenhang mit der möglichen Anzahl von Matrizen stehen, welche exponentiell zur Prozessgröße wächst. Insbesondere wenn Rauschen hinzukommt, lassen die Aussichten, einen großen Prozess vollständig zu erkennen, schnell nach.

Es ist anzunehmen, dass die Laufzeit sehr hoch ist, da beispielsweise jedes Individuum gegen alle Spuren geprüft wird. Die Verwendung eines stochastischen Verfahrens um die Tauglichkeit

zu approximieren, z.B. ähnlich wie bei dem induktiven Ansatz von Herbst, könnte hier einen Vorteil bringen.

Die vorgestellten genetischen Operationen haben einen recht geringen Einfluss auf ein Individuum. Auch hier wären weitere Alternativen interessant.

Der genetische Ansatz ist in der Anwendung ProM als Plug-In integriert. Für Details zu dem Ansatz, insbesondere auch zur Transformation der *Causal Matrix* in ein WF-Netz, verweisen wir auf die Arbeiten [83, 28]. Da erste Publikationen zu dem genetischen Ansatz erst vor kurzem veröffentlicht wurden, sind weitere Entwicklungen zu erwarten.

5.7.9 Zusammenfassung und Diskussion

Die meisten der vorgestellten Ansätze, um ein komplettes Prozessmodell aus den Verlaufsdaten abzuleiten, sind sich von der darunter liegenden Idee her ähnlich. Die Folgebeziehungen der Aktivitäten in den Spuren stellen potentielle kausale Beziehungen dar. Um festzulegen, welche davon echte Kausalitäten darstellen, werden voneinander unabhängige Aktivitäten bestimmt, sei es durch Verschränkung oder durch Überlappung der Aktivitäten. Ein wesentlich anderes Vorgehen verwendet lediglich der genetische Ansatz von de Medeiros und van der Aalst.

Auch hinsichtlich des Umgangs mit Rauschdaten sind sich die meisten Ansätze sehr ähnlich. Entweder werden keine Mechanismen zur Behandlung von Rauschdaten verwendet, oder Schwellenwerte werden eingesetzt, um die Relevanz von Beziehungen zwischen Aktivitäten zu bestimmen. Insbesondere werden Beziehungen, die den Schwellenwert nicht erreichen, ignoriert. Einzig InWoLvE, der Ansatz von Herbst, integriert niedrig-frequente Beziehungen in das resultierende Modell, macht diese jedoch durch Aufspaltung von Knoten und Angabe von Übergangswahrscheinlichkeiten deutlich. Vor allem wenn vom Prozessschema abweichende Ausführungen mit *Control Flow Mining* festgestellt werden sollen, erscheint diese Strategie sinnvoller als selten ausgeführte Pfade gänzlich zu ignorieren.

Ähnlich sind sich die meisten Ansätze auch darin, dass sie starke Annahmen bezüglich des dem Log zugrundeliegenden Prozessmodells machen oder bestimmte Aspekte nicht zulassen oder nicht berücksichtigen. Als sehr schwierig erweisen sich kausale Beziehungen zwischen nicht direkt benachbarten Aktivitäten sowie der Umgang mit einer nicht-injektiven Aktivitätszuordnungsfunktion. Einzig InWoLvE ist für einen Umgang mit letzterem gedacht.

Ansätze, die auf explizite Start- und Endereignisse angewiesen sind, um parallele Abläufe aufzudecken (z.B. der β-Algorithmus), können nicht eingesetzt werden, wenn nur Aktivitätsspuren vorliegen. Nicht zuletzt deshalb sind nicht alle Ansätze für alle Einsatzszenarien anwendbar. Für einen praktischen Einsatz empfiehlt es sich im allgemeinen, alternative Ansätze einzusetzen und die Ergebnisse zu vergleichen.

5.8 Mining von Transitionsbedingungen

Agrawal et al. haben in [4, 5] das Problem, Transitionsbedingungen des Prozessmodells abzuleiten, unter dem Begriff *Condition Mining* definiert. Dies spielt bei Agrawal et al. in sofern eine besondere Rolle, da sie zur Repräsentation des Prozessmodells gerichtete Graphen verwenden. Dadurch ist auch eine einfache Unterscheidung zwischen AND-Split und OR-Split nicht möglich.

In wiefern Transitionsbedingungen untersucht werden können, hängt von den im Log zur Verfügung stehenden Daten ab. Insbesondere müssen Werteverläufe der entscheidungsrelevanten Variablen bekannt sein.

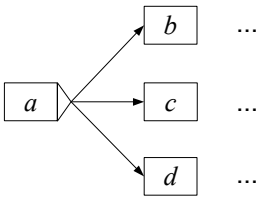

Abbildung 5.18: Ausschnitt eines Prozessmodells mit einem Split

Abbildung 5.18 zeigt den Ausschnitt eines Prozessmodells mit einem OR-Split bei Aktivität a. Tabelle 5.2 zeigt Beispieldaten für das Prozessfragment aus Abbildung 5.18. Mit $v_1(a)$ und $v_2(a)$ bezeichnen wir die Werte der zwei entscheidungsrelevanten Variablen v_1 und v_2 in der jeweiligen Instanz bei der Aktivität a.

Instanz	$v_1(a)$	$v_2(a)$	Ausgeführte Folgeaktivität
e_1	$true$	$false$	b
e_2	$false$	$true$	c
e_3	$false$	$false$	d
e_4	$true$	$true$	b
e_5	$false$	$true$	c

Tabelle 5.2: Beispieldaten für den in Abbildung 5.18 dargestellten Prozessausschnitt

Liegen ausreichend Daten vor, können verschiedene Klassifikationsverfahren angewendet werden, um Transitionsbedingungen abzuleiten. Agrawal et al. [4, 5] und Herbst [42] schlagen die Verwendung der Entscheidungsbauminduktion[12] vor.
Auf unsere Beispieldaten bezogen stellt jede Folgeaktivität von a eine Klasse dar. Ziel ist es, die Klassen anhand der Variablenwerte, die v_1 und v_2 in den Instanzen annehmen, zu separieren.

[12]Für weitere Informationen über die Entscheidungsbauminduktion verweisen wir auf Abschnitt 7.8.3, da wir die Entscheidungsbauminduktion später in der Arbeit ebenfalls einsetzen werden.

Dazu wird ein Entscheidungsbaum erzeugt, der die Instanzen anhand der Variablenwerte durch den Baum sortiert. Die abgeleiteten Regeln können wie folgt aussehen:

IF $(v_1(a) = true)$
THEN do b

IF $(v_1(a) = false$ AND $v_2(a) = true)$
THEN do c

IF $(v_1(a) = false$ AND $v_2(a) = false)$
THEN do d

Mit Zusatzwissen kann die Menge der entscheidungsrelevanten Variablen eingeschränkt werden. Dies ist beispielsweise der Fall, wenn die Auswahl zwischen b, c und d nur von Variablen abhängt, die von a geschrieben werden. Dadurch könnte die Rechenzeit verringert werden. Auch die Wahrscheinlichkeit, sinnvolle Regeln abzuleiten, steigt mit der Einschränkung der Variablenmenge auf die wirklich relevanten Variablen.

Neben der oben erläuterten Möglichkeit, einen Entscheidungsbaum für eine Verzweigung zu induzieren, ist es möglich, für jede Aktivität, die auf a folgen kann, einen Entscheidungsbaum zu erstellen (vgl. [42]). Für unser Beispiel gäbe es dann drei Bäume mit jeweils zwei Klassen, z.B. einen Baum für Aktivität b und die Klassen $true$ und $false$. Ziel ist es dann, Regeln abzuleiten, die ausdrücken, unter welchen Variablenwerten b ausgeführt wird.

Neben der Entscheidungsbauminduktion können noch weitere Verfahren eingesetzt werden, um Transitionsbedingungen abzuleiten. So verwenden Hwang und Yang in [48] beispielsweise das *Rule Induction System* CN2 [14].

5.9 Unterstützung der Evaluation von Prozessmodellen

Die Evaluation des abgeleiteten Prozessmodells durch Prozessexperten ist ein unerlässlicher Teilschritt im Mining-Prozess. Darüber hinaus können mit den meisten der in dieser Arbeit vorgestellten Ansätze durch Veränderung von Parametern, z.B. Schwellenwerten, unterschiedliche Prozessmodelle abgeleitet werden. Auch der Vergleich und die Evaluation dieser alternativen Modelle gehört zu den Aufgaben des Prozessexperten.

In [39, 40] schlägt Hammori die Verwendung von Data-Mining-Techniken zur Validierung des abgeleiteten Prozessmodells vor. Dem Prozessexperten soll damit ein Maß gegeben werden, um zu beurteilen, wie robust das abgeleitete Modell gegenüber zukünftigen Veränderungen der Verlaufsdaten ist. Der Grundgedanke dabei ist, die *Predictive Accuracy* [64, 39] des Modells zu überprüfen. Wenn die vorliegenden Verlaufsdaten z.B. nur einen kleinen Teil des Verhaltens

des ursprünglichen Prozessmodells abdecken, wird das abgeleitete Modell evt. nicht konsistent mit zukünftigen Verlaufsdaten, also ungesehenen Spuren, sein.
Als Evaluierungstechnik kann die *k-fold Cross Validation* (vgl. [64, 39]), eine Data-Mining-Technik, eingesetzt werden. Dabei wird die Menge der Spuren in k gleich große Teile geteilt. In k Schritten wird dann jeweils eine Menge als Testmenge ausgewählt. Die Spuren aus der Testmenge werden verwendet, um zu überprüfen, ob sie von dem abgeleiteten Prozessmodell generiert werden können. Die übrigen $k - 1$ Mengen werden als Trainingsmengen, also zur Ableitung des Prozessmodells, verwendet.

Cook und Wolf stellen in [20] zwei Metriken vor, um die Übereinstimmung zwischen Log-Daten und einem Modell zu evaluieren. Dabei wird im wesentlichen die Distanz zwischen der Ereignisspur aus dem Log und einer Spur des Modells berechnet, die der Ereignisspur am nächsten kommt. Als Kriterium, wie weit die Spuren voneinander entfernt sind, können Aspekte, wie die Größe der nicht übereinstimmenden Spurblöcke, einbezogen werden.

5.10 Praktischer Einsatz von Control Flow Mining

Die meisten der in diesem Kapitel vorgestellten Ansätze wurden bereits erfolgreich auf synthetischen Daten getestet. Studien über den Einsatz von *Control Flow Mining* in der Praxis sind jedoch rar. Im folgenden fassen wir Ergebnisse von einigen Studien zusammen.

Cook, Votta und Wolf stellen in [17] Ergebnisse einer Studie über die Anwendung ihrer Ansätze (siehe Abschnitt 5.7.1) in einem Industrieprojekt vor. Bei der Studie wurde der Software-Update-Prozess einer Entwicklungsabteilung untersucht. Ziel war es u.a. herauszufinden, welche Zusammenhänge zwischen einer erfolglosen Prozessinstanz und ihrer Ausführung bestehen. Unter Verwendung der Algorithmen von Cook und Wolf wurde aus Log-Daten von 159 Prozessinstanzen ein Prozessmodell generiert. Cook et al. versuchten, den Erfolg bzw. das Scheitern des Prozesses mit Merkmalen der Prozessausführung, z.B. Ausführungszeit oder interne Verzögerungen, in Beziehung zu setzen. Diese Studie geht daher über reines *Control Flow Mining* weit hinaus. Es konnte festgestellt werden, dass die tatsächliche Ausführung des Prozesses stark von dem vorliegenden Prozessmodell abweicht. Darüber hinaus wurden Zusammenhänge zwischen internen Verzögerungen bei der Prozessausführung und der Akzeptanz des Updates festgestellt. So stieg die Wahrscheinlichkeit, dass ein Update vom Kunden nicht akzeptiert wird, mit der Verzögerungszeit zwischen Anfrage und Lieferung.
Insgesamt betrachten die Autoren ihre Studie als erfolgreich. Insbesondere heben sie die geringen Kosten für die Untersuchung positiv hervor. Als Ergebnis der Untersuchung konnte die Entwicklungsabteilung ihren Prozess entscheidend verbessern, indem sie etwa Alarmfunktionen für interne Verzögerungen einführten. Ein interessanter Aspekt der Studie von Cook et al. liegt sicherlich auch darin, dass die Log-Daten sowohl automatisch als auch manuell

erzeugt wurden.

Herbst und Kleiner stellen in [44] Ergebnisse der Anwendung von InWoLvE (siehe Abschnitt 5.7.7) in einem Workflow-Projekt vor. Das Ziel dieses Projekts war die Verbesserung eines bestehenden Änderungsmanagement-Prozesses für elektronische Kontrolleinheiten sowie die Entwicklung einer neuen Workflow-Anwendung[13] zur Unterstützung des neuen Prozesses. Ein erster Entwurf des neuen Prozesses wurde bereits durch eine Gruppe von Experten und Anwendern des bestehenden Systems vorgelegt. Die von dem alten System mitprotokollierten Daten beschränkten sich auf einen Status (z.B. *geprüft*), das Datum der Statusänderung sowie eine Bemerkung.
Die Anwendung von InWoLvE auf die Log-Daten ermöglichte die Ableitung des Prozesses. Es wurde festgestellt, dass einige Aktivitäten, z.B. das Testen der Software, nur sehr selten ausgeführt wurden. Die Anwendung von InWoLvE half dabei, diese Aktivitäten, die aus gutem Grund bereits aus dem Workflow-Modell entfernt wurden, zu erkennen. So konnten sie auch aus dem neuen Entwurf des Prozesses entfernt werden. Allerdings ist anzumerken, dass der untersuchte Prozess sehr einfacher Natur zu sein scheint. Darum ist diese Studie nicht als eine Untersuchung zu sehen, die die Grenzen der Anwendbarkeit von *Control Flow Mining* untersucht, sondern vielmehr als ein Beleg für eine sinnvolle Anwendungsmöglichkeit von *Control Flow Mining* zu betrachten.

In [82] berichtet van der Aalst von zwei weiteren Studien. *Control-Flow-Mining*-Techniken der Arbeitsgruppe von Prof. van der Aalst wurden zur Ableitung eines Prozesses zur Einforderung von Bußgelder einer niederländischen Regierungseinrichtung eingesetzt. Jedes Bußgeldverfahren stellt eine Prozessinstanz dar. Der untersuchte Prozess enthält 99 Aktivitäten und ist damit wesentlich umfangreicher als die von Cook et al. und Herbst und Kleiner untersuchten Prozesse. Die Verlaufsdaten stammen von einem proprietären System. Das abgeleitete Modell konnte wesentliche Aspekte des Prozesses abdecken. Diese Untersuchung ist auch in der Dissertation von Maruster [61] zu finden.

In [82] wird darüber hinaus auch über die Untersuchung des Flusses von bestimmten Patienten (*Multi-disciplinary Patients*) in einer Krankenhausumgebung berichtet. Besuche der Patienten bei Spezialisten stellen die Aktivitäten dar. Grundtenor des Berichts ist, dass sich der Einsatz von *Control Flow Mining* bei derart unstrukturierten Prozessen (*"spaghetti-like"*) als schwierig erweist. Daher ist es notwendig zu filtern, z.B. nur häufige Aktivitäten zu berücksichtigen.

Abschließend ist zu sagen, dass die bisherigen Studien weniger die Untersuchung der Grenzen der Anwendbarkeit von *Control Flow Mining* darstellen. Lediglich die von van der Aalst in [82] vorgestellte Studie geht in diese Richtung. Es scheint, dass der Einsatz von *Control Flow Mining*, insbesondere Versuche einen vollständigen Prozess abzuleiten, bei unstruktu-

[13] Herbst und Kleiner verstehen unter dem Begriff Workflow-Anwendung alle Anwendungen, die einen Workflow implementieren, unabhängig davon, ob ein WfMS verwendet wird oder nicht.

rierten Prozessen weniger erfolgreich ist. Leider sind keine ausführlichen Informationen über diese Studie vorhanden. Da momentan noch zu wenig Studien und Erfahrungen darüber vorliegen, sind weitere Ergebnisse abzuwarten.
Als ein positiver Aspekt zeigt sich allerdings, dass eine feingranulare Log-Struktur, wie es z.B. bei Workflow-Management-Systemen der Fall ist, für eine sinnvolle Anwendung von *Control Flow Mining* nicht unbedingt erforderlich ist. Vielmehr ist es notwendig, die zur Verfügung stehenden Log-Daten sinnvoll zu verwenden und zu interpretieren oder gegebenenfalls auf den Prozess zugeschnittene Ereignisse zu definieren, wie z.B. bei Cook et al.

5.11 Zusammenfassung und Ausblick

In diesem Kapitel haben wir das Thema *Control Flow Mining*, die am genausten untersuchte Fragestellung von *Process Mining*, erörtert und bestehende Ansätze vorgestellt.
Wie wir gezeigt haben, ist *Control Flow Mining* keine triviale Fragestellung. Ein perfektes Modell aus den Verlaufsdaten abzuleiten erscheint unmöglich, insbesondere im Hinblick auf Rauschdaten und unvollständigen Log-Daten. Nicht nur deshalb ist es offensichtlich, dass *Control Flow Mining* nicht zur voll-automatischen Konstruktion von Prozessmodellen dienen kann. Vielmehr stellen die in dieser Arbeit vorgestellten Ansätze eine Unterstützung für den Prozessexperten dar. Eine Evaluation der Ergebnisse ist unabdingbar.

Einige Aspekte der Problemstellung selbst stellen ein größeres Problem für viele Ansätze dar. Allerdings können aus diesen Aspekten, die in Abschnitt 5.3 erläutert wurden, Anforderungen für Log-Mechanismen abgeleitet werden. So würde es den Mining-Prozess erheblich erleichtern, wenn Iterationen eines Zyklus jeweils bei der Protokollierung berücksichtigt würden. Das WfMS ADEPT [73, 70, 71] weist bereits einen solchen Log-Mechanismus auf. Hilfreich bei der Problematik im Umgang mit nicht-injektiven Aktivitätszuordnungsfunktionen wäre, wenn neben dem Namen der Aktivität auch die Knotennummer vermerkt wäre. Dadurch ließe sich die Menge der gleichnamigen Aktivitäten eindeutig unterschiedlichen Knoten zuordnen. Abbildung 5.19 zeigt einen Prozess, bei dem eine Aktivität mehreren Knoten zugeordnet ist. Die Knoten verfügen jeweils über eine Knotennummer. Zwei Aktivitätsspuren dieses Prozesses könnten daher wie folgt aussehen:

$< (a,1), (b,2), (d,4), (f,6), (h,10) >$
$< (a,1), (c,3), (d,7), (e,8), (g,9), (h,10) >$

In vielen Umgebungen sind die erwähnten Log-Mechanismen sicherlich nicht möglich, insbesondere wenn bereits Log-Daten vorliegen, mit denen gearbeitet werden muss. Im Kontext von Workflow-Management-Systemen ist die Realisierung dieser Mechanismen jedoch oft ohne

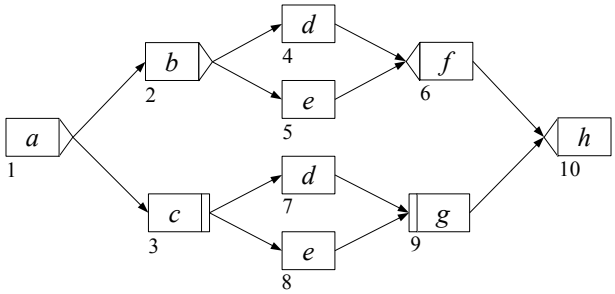

Abbildung 5.19: Ein Prozess mit einer nicht-injektiven Aktivitätszuordnungsfunktion sowie eindeutigen Knotennummern

weiteres möglich.

Wie auch durch die Studien deutlich wurde, sprechen besonders die vergleichsweise geringen Kosten für den Einsatz von *Control Flow Mining*. Insbesondere wenn eine Infrastruktur zur Erfassung von Log-Daten bereits gegeben ist, ist die Durchführung von *Control Flow Mining* zur Untersuchung von Prozessen mit nur wenig Aufwand verbunden.

Die Objektivität der Modelle, die mittels *Control Flow Mining*-Techniken abgeleitet werden, ist ein weiterer Vorteil. Alle traditionellen Techniken zur Erfassung von Prozesswissen, z.B. Interviews oder Fragebögen, sind sehr subjektiver Natur. So erscheint es nur sinnvoll, *Control Flow Mining* als ergänzende Technik zur Wissensakquisition einzusetzen.

Control Flow Mining ist ein aktuelles Thema, so sind die meisten Arbeiten dazu jüngeren Datums. Aktuell findet viel Forschungsarbeit in diesem Gebiet statt. Weitere Entwicklungen, insbesondere neue Ansätze und Verbesserungen von bestehenden Ansätzen, sind daher in nächster Zeit zu erwarten. Interessant wären auch weitere Studien über den praktischen Einsatz von *Control Flow Mining*. Insbesondere jene, die die Grenzen der Anwendbarkeit aufweisen, da dies anhand der wenigen bestehenden Studien schwer zu beurteilen ist.

Kapitel 6

Mining organisatorischer Aspekte

6.1 Einleitung

Bisherige Arbeiten zu *Process Mining* konzentrieren sich hauptsächlich auf *Control Flow Mining*. Organisatorische Aspekte wurden kaum berücksichtigt. Ein Grund dafür könnte darin liegen, dass die Möglichkeiten, um organisatorische Aspekte aus Verlaufsdaten abzuleiten, sehr begrenzt sind. Ein vollständiges Organisationsmodell aus den Verlaufsdaten abzuleiten, wäre eine unrealistische Aufgabe. Zum einen bieten die Verlaufsdaten typischerweise nicht genügend Informationen für eine solche Aufgabe, zum anderen ist es fraglich, ob das Organisationsmodell aus der Sicht eines Prozesses heraus konstruiert werden sollte. Bisherige Bemühungen, auch organisatorische Aspekte zu integrieren, können in zwei Kategorien eingeteilt werden:

- Beziehungen zwischen Agenten
- Beziehungen zwischen Aufbauorganisation und dem Prozess

Die erste Kategorie konzentriert sich auf Beziehungen zwischen den Agenten. Bei der zweiten Kategorie geht es um die Verbindung zwischen der Aufbauorganisation und dem Prozess.
Zur ersten Kategorie stellen van der Aalst und Song in [85, 84] unter dem Oberbegriff *Mining Social Networks* einen Ansatz vor, um soziale Netze zwischen den Agenten zu analysieren. Auf diesen Ansatz gehen wir im Abschnitt 6.2 genauer ein.
Die zweite Kategorie wurde bis dato lediglich angedacht. Konkrete Forschungsarbeiten dazu sind noch nicht vorhanden. Wir haben uns diesem Thema angenommen und stellen im Kapitel 7.1 einen Ansatz vor, um Bearbeiterzuordnungsregeln aus Verlaufsdaten zu extrahieren.

6.2 Mining Social Networks

In [85, 84] stellen van der Aalst und Song ihre Arbeit vor. Unter dem Begriff *Mining Social Networks* fassen sie die Aufgabenstellung zusammen, soziale Netze aus Verlaufsdaten abzuleiten und zu analysieren.
Einträge aus Verlaufsdaten enthalten typischerweise, neben Informationen, die für *Control Flow Mining* von Bedeutung sind, auch Bearbeiterinformationen. So ist den Ereignissen im Log meist auch der Benutzername des Agentens zugeordnet, der mit diesem Ereignis assoziiert wird. Van der Aalst und Song verwenden diese Bearbeiterinformationen, um soziale Netze im Kontext von Prozessen abzuleiten. Unter Verwendung von bestimmten Metriken, die die Beziehung der Bearbeiter zueinander im Kontext des Prozesses verdeutlichen, werden sog. Soziogramme (*Sociograms*) erstellt. Diese können gängigen Anwendungen zur Analyse von sozialen Netzen als Eingabe dienen und somit zur weiteren Analyse des sozialen Netzwerks verwendet werden.
Im folgenden wird zuerst ein Einblick in die Netzwerkanalyse gegeben. Danach wird auf die von van der Aalst und Song entwickelten Metriken eingegangen.

6.2.1 Social Network Analysis

Soziometrie (*Sociometry*), abgeleitet von den lateinischen Wörtern „*socius*“ für „sozial“ und „*metrum*“ für „messen“, fasst Methoden und Techniken zusammen, um Relationen zwischen Individuen messbar zu machen.
Die Netzwerkanalyse (*Social Network Analysis*), kurz SNA, hervorgegangen aus der Soziometrie, beschäftigt sich mit der Analyse von sozialen Netzen [29]. Als Ausgangspunkt der Analyse dienen sogenannte Soziogramme oder Soziomatrizen (*Sociomatrices*). Diese stellen das zu analysierende soziale Netz dar.
Ein Soziogramm, erfunden von Jacob Levy Moreno, der als Begründer der Soziometrie gilt, ist ein gerichteter oder ungerichteter Graph. Die Knoten des Graphs stellen Individuen, die Kanten die Beziehung zwischen den Individuen dar. Das soziale Netzwerk wird auf Grundlage dieser Beziehung analysiert. Die Kanten können gewichtet sein. Dies ermöglicht eine Qualifizierung der Beziehung. Eine Soziomatrix ist eine Matrix-Darstellung des Soziogramms.

Die formale Netzwerkanalyse unterscheidet zwischen drei Aspekten: Analysen zur Netzwerkstruktur, z.B. Netzdichte und Gliederung in Teilnetze, Analysen von Knoten, z.B. wie zentral ein Individuum ist, sowie Analysen zu Art und Eigenschaften der Beziehung, z.B. Symmetrie und Transitivität [29]. Für diese Analysen wurden zahlreiche Metriken, z.B. Bavelas-Leavitt als Index für die Zentralität eines Knotens, entwickelt. Für weitere Metriken aus der Netzwerkanalyse wird auf die Arbeit von van der Aalst und Song verwiesen [85]. Eine Einführung in die Netzwerkanalyse findet sich in [29].
Viele Anwendungen für die Netzwerkanalyse, z.B. Agna, sind verfügbar. Als Eingabe dient

den gängigen Anwendungen ein Soziogramm.
Die notwendigen Informationen für die Erstellung von Soziogrammen werden meist mittels traditioneller Methoden, insbesondere Befragungen, erhoben. Van der Aalst und Song wenden die Netzwerkanalyse jedoch im Kontext von Prozessen an und können daher auf die Verlaufsdaten zurückgreifen.

6.2.2 Ableitung von Soziogrammen aus Verlaufsdaten

Ziel der Arbeit von van der Aalst und Song ist die Generierung von Soziogrammen aus den Verlaufsdaten als Eingabe für weitere formale Netzwerkanalysen.
Da van der Aalst und Song die Netzwerkanalyse in den Prozesskontext verlagern, ist es notwendig, Metriken zu definieren, die für diesen Kontext Beziehungen zwischen Agenten ausdrücken können. Dazu haben sie vier Kategorien von Metriken definiert:

1. Metriken basierend auf (möglicher) Kausalität
2. Metriken basierend auf gemeinsamen Prozessinstanzen
3. Metriken basierend auf gemeinsamen Aktivitäten
4. Metriken basierend auf besonderen Ereignistypen

Prozessinstanz	*Aktivität*	*Bearbeiter*
Instanz 1	A	M1
Instanz 1	B	M2
Instanz 1	C	M4
Instanz 1	D	M6
Instanz 2	A	M5
Instanz 2	B	M1
Instanz 2	C	M2
Instanz 2	D	M1
Instanz 3	A	M3
Instanz 3	C	M5
Instanz 3	B	M4
Instanz 3	D	M1
Instanz 4	A	M1
Instanz 4	B	M2
Instanz 4	C	M6
Instanz 4	D	M4

Tabelle 6.1: Beispiel für Verlaufsdaten mit Bearbeiterinformationen

Für die erste Kategorie wurden *Handover of work metrics* und *In-between metrics* definiert. *Handover of work* beschreibt die Reihenfolge, in der Bearbeiter Aktivitäten ausführen. Mitarbeiter M1 aus Tabelle 6.1 bearbeitet in der Instanz 1 die Aktivität A, bevor M2 Aktivität B ausführt. Daher findet ein *Handover of work* von M1 zu M2 statt. Abbildung 6.1 zeigt ein Soziogramm für die Verlaufsdaten aus Tabelle 6.1. Die Kanten in dem Soziogramm stehen für eine direkte Folgebeziehung zwischen den Aktivitäten der Agenten.

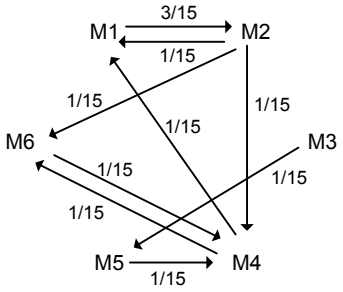

Abbildung 6.1: Soziogramm mit *Handover of work metrics* für Tabelle 6.1

In-between metrics berücksichtigen den Fall, dass ein Agent eine Aktivität zwischen zwei von einem anderen Agenten bearbeiteten Aktivitäten ausführt. Diese Situation ist für M1 und M2 in der Instanz 2 der Tabelle 6.1 gegeben. Die Annahme dabei ist, dass diese Situation ein Hinweis auf eine Art *Subcontracting*-Beziehung zwischen M1 und M2 sein könnte. Weitere Verfeinerungsmöglichkeiten der Metriken sind möglich. So können zum Beispiel auch indirekte Folgebeziehungen miteinbezogen werden. Unter Verwendung von Wissen über das Prozessschema kann auch festgestellt werden, ob die Folgebeziehungen aufgrund von kausalen Zusammenhängen zustande gekommen sind.

Bei Metriken basierend auf gemeinsamen Prozessinstanzen (*Working together metrics*) besteht die Annahme, dass eine stärkere Beziehung zwischen Agenten besteht, die an der Ausführung derselben Prozessinstanz beteiligt sind. Dabei wird im wesentlichen gezählt, wie oft zwei Agenten Aufgaben in derselben Prozessinstanz übernehmen und dies in Verhältnis dazu gesetzt, an wie vielen Instanzen die Agenten beteiligt sind. Für M1 und M2 aus Tabelle 6.1 wäre dies: $\mathrm{M1} \bowtie \mathrm{M2} = \frac{3}{4}$ und $\mathrm{M2} \bowtie \mathrm{M1} = \frac{4}{4}$ ($\bowtie$ ist das Symbol für diese Beziehung). Auch hier sind Verfeinerungen möglich. So kann beispielsweise die Distanz zwischen den Aktivitäten berücksichtigt werden.

Metriken der dritten Kategorie, *Metrics based on joint activities*, liegt die Annahme zugrunde, dass Agenten, die ähnliche Aktivitäten ausführen, eine stärkere Beziehung zueinander haben. Um dies zu berechnen wird eine Matrix mit möglichen Aktivitäten als Spalte und Agenten als Zeile erstellt. Die Einträge, jeweils 1 oder 0, ergeben sich daraus, ob ein Agent die jeweilige Aktivität bearbeitet. Um die Distanz zwischen den Zeilenvektoren zu berechnen, werden verschiedene Formeln verwendet, u.a. auch die *Hamming-Distanz*. Ein Problem könnte

hierbei allerdings darin bestehen, dass unterschiedliche Aktivitäten einander durchaus ähnlich sein können. Die Ähnlichkeit unterschiedlicher Aktivitäten ist u.U. aufwendig zu bestimmen. Dieser Aspekt wird bei dieser Metrik allerdings auch nicht berücksichtigt.

Neben den Start- und Endereignissen einer Aktivität können auch andere Ereignisse während einer Prozessausführung, z.B. für Ausnahmefälle, auftreten. Welche Ereignisse das konkret sind und wie sie zu interpretieren sind, hängt von dem Prozess-Management-System bzw. dem Protokollierungsumfeld ab. Metriken der vierten Kategorie berücksichtigen solche Ereignisse. Van der Aalst und Song haben sich im Speziellen mit dem Ereignistyp *Reassign* beschäftigt. *Reassign* tritt zum Beispiel auf, wenn ein Agent einen Arbeitslisteneintrag an einen anderen Agenten weiterleitet und entspricht damit einer Delegierung der Arbeit (siehe auch Kapitel 4). Die Annahme bei dieser Metrik ist, dass eine solche Situation auf eine hierarchische Beziehung zwischen den beiden Agenten hindeutet. Zur Berechnung der *Reassignment metrics* zwischen zwei Agenten wird im wesentlichen die Anzahl der entsprechenden Ereignisse gezählt und im Verhältnis zur Anzahl möglicher *Reassign*-Ereignisse gesetzt.

Mit Hilfe der oben beschriebenen Metriken können Soziogramme mit gewichteten Kanten abgeleitet werden. Eine weitere Analyse der Soziogramme kann mit gängigen Anwendungen für die Netzwerkanalyse durchgeführt werden.

6.2.3 MiSon

Mit MiSoN haben van der Aalst und Song auch eine Anwendung vorgestellt, die ihren Ansatz umsetzt. Abbildung 6.2 zeigt einen Screenshot der Anwendung.MiSoN verwendet als Eingabe Verlaufsdaten in einem XML-Format, auf das im Abschnitt 4.2.3 eingegangen wird, und generiert daraus Soziogramme. Es ist auch in das ProM-Framework als Plug-In eingebunden.

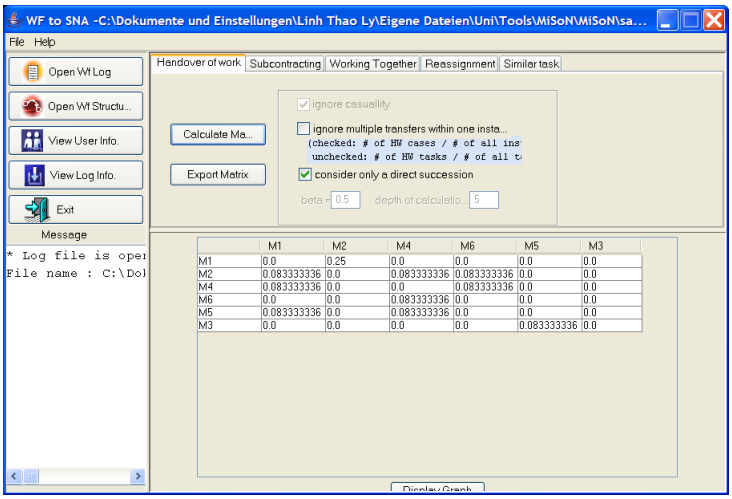

Abbildung 6.2: Screenshot von MiSoN

6.2.4 Anwendung auf realen Daten

In [84] stellen van der Aalst und Song Ergebnisse einer Studie vor, bei der die Autoren ihre Metriken auf Verlaufsdaten eines niederländischen Amtes für staatliche Bauvorhaben (*Dutch national public works department*) angewendet haben. Der betreffende Prozess umfasst 17 Aktivitäten mit fast 5000 Instanzen und über 33000 Ereignissen. 43 Agenten waren an der Prozessausführung beteiligt. Die Autoren wendeten verschiedene Techniken der Netzwerkanalyse auf die mittels ihrer Metriken extrahierten Soziogramme an.
Ein interessantes Teilergebnis stellen die mit Metriken der dritten Kategorie generierten Clustern von den Agenten dar. Interessant ist dabei, dass einige Agenten eine Brückenfunktion in den Clustern übernahmen, d.h. sie stellen die Verbindung zweier Cluster dar. Die Autoren interpretieren dies so, dass die jeweiligen Agenten mehrere Rollen übernehmen. Auch Analysen, bei denen Gruppen von Agenten die Knoten im Soziogramm darstellen, wurden durchgeführt.

Aus Diskretionsgründen gingen die Autoren allerdings nicht auf die Bewertung der Ergebnisse ihrer Studie ein. Darüber hinaus waren die Auswertungsarbeiten zum Zeitpunkt ihrer Publikation noch nicht beendet.

6.2.5 Zusammenfassung und Diskussion

Mit ihrem Ansatz gehen die Autoren in Richtung des Gebiets *Enterprise Social Network Analysis*, welches sich mit der Analyse sozialer Netze in Unternehmen beschäftigt.
Mit der Definition von prozesskontextspezifischen Metriken haben van der Aalst und Song die Netzwerkanalyse in den Prozesskontext verlagert. Insbesondere ermöglicht ihr Ansatz die Verwendung der Verlaufsdaten als Ausgangspunkt für die Netzwerkanalyse. Über die Nützlichkeit der dieser Analysen kann keine Aussage getroffen werden, da dafür noch zu wenig konkrete Ergebnisse vorhanden sind, zumal die durchaus interessanten Ergebnisse der einzigen Studie der Anwendung der Ansätze auf realen Daten nicht bewertet werden können.
Die Autoren betonen in [85, 84], dass sie dabei sind, ihre Ansätze zu erweitern. Insbesondere sollen Zusatzinformationen über die Aktivitäten, z.B. wichtige oder unwichtige Aktivität, mit einbezogen werden.

Kapitel 7

Staff Assignment Mining

7.1 Einleitung

7.1.1 Motivation

Der Modellierung von Prozessabläufen wurde auch in der Forschung viel Aufmerksamkeit gewidmet. Zahlreiche Arbeiten befassen sich damit, wie Geschäftsprozesse sinnvoll modelliert werden können. Die Verbindung zwischen organisatorischen Aspekten und Prozessabläufen dagegen wurde bisher weniger beachtet [98]. Dabei sind organisatorische Aspekte nicht zuletzt für den Workflow-Kontext von großer Bedeutung.

Eine jüngere Studie, die die häufigsten Ursachen für das Scheitern von Workflow-Projekten untersuchte, fand eine der wichtigsten Ursachen in der Verwendung schlechter Strategien zur Verteilung von Arbeitslisteneinträgen [98, 65]. Arbeitslisteneinträge werden oftmals in gemeinsam verwendete Arbeitslisten gelegt, auf denen zu viele Mitarbeiter Zugriff haben. Dadurch ist es dem einzelnen Mitarbeiter schwer möglich zu identifizieren, welche dieser Einträge in seinen Zuständigkeitsbereich fallen. Dieses Vorgehen ist kontraproduktiv. Es widerspricht einer der Grundideen, die Workflow-Management-Systemen zugrunde liegt, die Ausführung von Prozessen so zu unterstützen, dass anfallende Aufgaben zur richtigen Zeit an die richtigen Mitarbeiter weitergeleitet werden. Eine ungleiche Lastverteilung, Überforderung der Mitarbeiter und längere Bearbeitungszeiten sind mögliche Folgen einer schlechten Strategie der Arbeitsteilung.
Bearbeiterzuordnungsregeln stellen das Bindeglied zwischen dem Prozess und der Aufbauorganisation dar und reflektieren die Strategie, mit der anfallende Aufgaben an Mitarbeiter verteilt werden. Somit stellen sie auch einen kritischen Punkt für den Erfolg von Workflow-Projekten dar.

Im Rahmen dieser Arbeit beschäftigen wir uns mit der Extraktion von Bearbeiterzuordnungsregeln aus Verlaufsdaten unter Verwendung eines Organisationsmodells. Wir wollen dies

mit dem Begriff *Staff Assignment Mining* bezeichnen. Im Kontext von *Process Mining* füllt unsere Arbeit damit eine Lücke auf, da der organisatorische Aspekt in *Process Mining* bisher kaum angegangen wurde.

7.1.2 Überblick über das Kapitel

Im nächsten Abschnitt wird die Problemstellung genauer erläutert. Abschnitt 7.4 gibt einen Überblick über verwandte Arbeiten. Die Rahmenbedingungen werden in den Abschnitten 7.5, 7.6 und 7.7 dargestellt. Dabei geht 7.5 auf Anforderungen an die Ausgangsbedingungen ein. Abschnitt 7.6 stellt das verwendete Organisations-Metamodell vor, und Abschnitt 7.7 geht auf die Darstellung der Bearbeiterzuordnungsregeln ein. Im Abschnitt 7.8 wird der Lösungsansatz vorgestellt. Im Abschnitt 7.9 folgt die abschließende Zusammenfassung und Diskussion.

7.2 Problemstellung

In dieser Arbeit widmen wir uns dem Problem, Bearbeiterzuordnungsregeln aus Protokolldaten über vergangene Prozessausführungen und einem Organisationsmodell zu abzuleiten.
Organisationsmodelle sind in den meisten Unternehmen oftmals in der einen oder anderen Form schon vorhanden. Selbst falls ein solches noch nicht existiert, ist die Erstellung eines Organisationsmodells sinnvoll und im Hinblick auf Möglichkeiten, dieses weiterzuverwenden, sehr nützlich. Daher nehmen wir als Ausgangssituation die Existenz eines solchen Organisationsmodells an.
Wenn für die untersuchte Aktivität x Verlaufsdaten vorliegen, die eine eindeutige Identifizierung der Bearbeiter von x in verschiedenen Prozessinstanzen ermöglichen, stellt sich die Frage, ob und wie Bearbeiterzuordnungsregeln für x unter Verwendung des Organisationsmodells abgeleitet werden können. Diese Frage soll im Rahmen dieser Arbeit geklärt werden. Im folgenden beziehen wir uns stets auf die Aktivität x, für die Bearbeiterzuordnungsregeln abgeleitet werden sollen.

Bearbeiterzuordnungsregeln bestimmen gewissermaßen das Profil der zulässigen Bearbeiter für eine Aktivität. Dieses Profil setzt sich aus Qualifikationen zusammen, die ein Mitarbeiter erfüllen muss, um die Aktivität ausführen zu können (eine Definition von Qualifikation in diesem Kontext findet sich im Abschnitt 7.9).
Entsprechend dem Ziel von *Process Mining*, den Ist-Zustand der Prozessausführung aufzuzeigen, muss es auch das Ziel von *Staff Assignment Mining* sein, den Ist-Zustand der Bearbeiterzuordnung aufzuzeigen.
Die abgeleiteten Regeln sollen, die Bearbeiter einer Aktivität über ihre Qualifikationen zu identifizieren, so dass sie auch direkt in einem Workflow-Management-System zur *Staff Re-*

solution verwendet werden könnten. Dabei sollen sie jedoch möglichst minimal sein, d.h. sie sollen nur ein minimales Profil des Bearbeiters festlegen.
Formaler ausgedrückt, wollen wir zu einer gegebenen Aktivität x die Bearbeiterzuordnungsregeln von x, kurz BZR(x), finden. Dabei müssen die Regeln BZR(x) konsistent zu den Verlaufsdaten sein, in dem Sinne, dass unter Verwendung von BZR(x) dieselben Verlaufsdaten erzeugt werden können.

7.3 Anwendungen von Staff Assignment Mining

Spätestens wenn die Ausführung von Prozessen mit Hilfe von Workflow-Management-Systemen unterstützt werden soll, sind allgemein formulierte Bearbeiterzuordnungsregeln notwendig. Traditionelle Techniken der Wissensakquisition, z.B. Interviews und Fragebögen, erfassen oftmals nicht den Ist-Zustand der Bearbeiterzuordnung. Vor allem bei gewachsenen Prozessen ist es schwierig, allgemeine Bearbeiterzuordnungsregeln zu formulieren. *Staff Assignment Mining* kann eine Ergänzung zu traditionellen Techniken der Wissensakquisition darstellen. Insbesondere können wir mit Hilfe von *Staff Assignment Mining* den Ist-Zustand der Bearbeiterzuordnung einer Aktivität feststellen. Dies ermöglicht auch den Vergleich des Soll-Zustandes mit dem Ist-Zustand. Wenn etwa eine speziellere Regel extrahiert wurde als ursprünglich angenommen (siehe Abbildung 7.1), so deutet das darauf hin, dass die ursprüngliche Regel nicht alle Anforderungen erfasst und damit zu allgemein ist. Ebenso kann es sein, dass nur ein Teil der Mitarbeiter, die für diese Tätigkeit vorgesehen sind, ihrer Arbeit auch nachkommen. In diesem Fall kann beispielsweise eine Analyse der Auslastung der Mitarbeiter zu einer besseren Strategie zur Arbeitsverteilung führen.

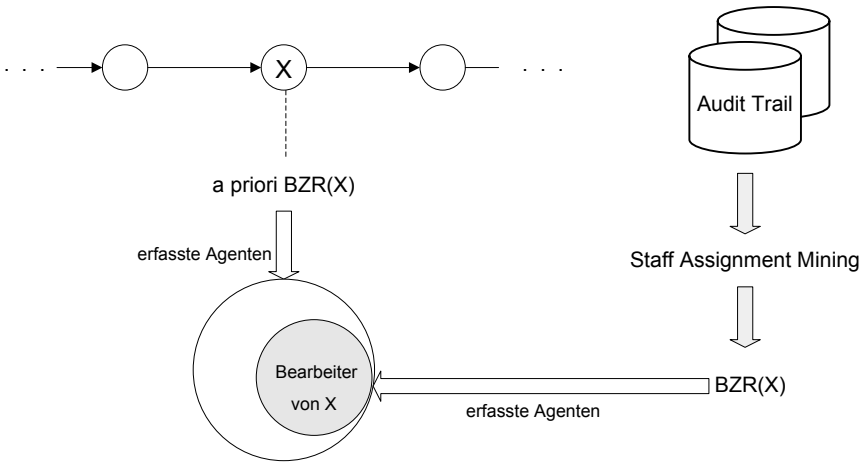

Abbildung 7.1: Einsatzszenario von Staff Assignment Mining

So kann das Wissen über den Ist-Zustand der Bearbeiterzuordnung einer Aktivität neben dem Verständnis und der Dokumentation des Prozesses auch der Verbesserung des Prozes-

ses dienen. Insbesondere bei nicht erfolgreichen Workflow-Projekten ist es sinnvoll auch die Bearbeiterzuordnung zu überprüfen. Wie in dem eingangs erwähnten Szenario über Arbeitslisteneinträge in gemeinsam verwendeten Arbeitslisten können die Ursachen auch bei den Bearbeiterzuordnungen liegen.

7.4 Verwandte Arbeiten

Mit dieser Fragestellung hat sich, so weit wir wissen, noch niemand beschäftigt. Verwandte Arbeiten können daher nur für ähnliche Fragestellungen genannt werden.
In [85, 84] stellen van der Aalst und Song ihren Ansatz zu Mining Social Networks vor (siehe Abschnitt 6.2). Dabei weisen die Autoren auch auf die Möglichkeit hin, organisatorische Strukturen, in diesem Fall Rollen, aufgrund von Verlaufsdaten zu erraten. Ihre Idee läuft darauf hinaus Mitarbeitern Rollen über die Aktivitäten, die sie ausführen, zuzuordnen. Mitarbeiter, die dieselbe Aktivität ausführen haben demnach dieselbe Rolle, die für die Ausführung dieser Aktivität notwendig ist. Allerdings wiesen die Autoren lediglich auf diese Möglichkeit hin. Eine weitere Arbeit von ihnen zu diesem Thema konnte nicht gefunden werden. Darüber hinaus beziehen sich die Autoren nicht auf die Möglichkeit der Einbeziehung eines Organisationsmodells. Außer dieser Arbeit sind uns keine anderen Arbeiten mit einer ähnlichen Fragestellung bekannt.

7.5 Anforderungen

7.5.1 Anforderungen an die Verlaufsdaten

Um Bearbeiterzuordnungsregeln aus Verlaufsdaten ableiten zu können, müssen wir die Bearbeiter der zu untersuchenden Aktivität x kennen. Dazu ist es erforderlich, dass jedem Vorkommen von x in einer Prozessinstanz eindeutig ein Bearbeiter zugeordnet werden kann.
Eine Aktivität ist typischerweise begrenzt durch ein Beginn- und ein entsprechendes Endereignis. Wir abstrahieren an dieser Stelle jedoch von konkreten Ereignissen. Dazu ist es notwendig, dass sowohl dem Beginn- als auch dem Endereignis einer Aktivität derselbe Bearbeiter zugeordnet wird, so dass wir auch einer Aktivität eindeutig einem Bearbeiter zuordnen können. Tabelle 7.1 zeigt wie eine Auflistung der Bearbeiter von x unter Abstraktion von Ereignissen aussehen kann.

Ein Audit-Trail-Eintrag enthält, neben einem Identifikator für die Instanz, die Aktivität und dem Ereignis, typischerweise auch einen Identifikator für den verantwortlichen Bearbeiter. Im Kontext von WfMS und anderen prozessorientierten Systemen ist die Forderung den Bearbeiter identifizieren zu können i.d.R. erfüllt.

Instanz	*Bearbeiter-Id*
1	Ma1
2	Ma2
3	Ma1
4	Ma1
5	Ma3
...	...

Tabelle 7.1: Verlaufsdaten für Aktivität x

Manche WfMS, z.B. ADEPT [70], protokollieren neben dem Bearbeiter einer Aktivität x auch die Stelle, über die der Benutzer bei der Ausführung von x am System angemeldet ist. Die Zusatzinformation über die Stelle des Bearbeiters ist für unsere Arbeit nicht erforderlich, zumal sie nicht für die Mehrheit der Systeme vorausgesetzt werden kann.

Im Kontext von Prozessen, deren Ausführung nicht elektronisch unterstützt wird, müssen die notwendigen Verlaufsdaten auf andere Weise gesammelt werden. Viele Aktivitäten werden mit Akten, Belegen oder Ähnlichem dokumentiert, in der auch meist der ausführende Mitarbeiter vermerkt ist, so dass solche Dokumente ebenfalls zur Ermittlung der Bearbeiter herangezogen werden können. In [94] beschreiben Wolf und Rosenblum ihr Vorgehen zur Akquisition von Verlaufsdaten über einen nicht automatisierten Prozess für eine Case Study. In ähnlicher Form kann die Akquisition der Bearbeiterdaten auch erfolgen.

7.5.2 Anforderungen an das Organisationsmodell

Neben einer eindeutigen Identifikation der verantwortlichen Bearbeiter einer untersuchten Aktivität setzen wir den Zugang zu einem Organisationsmodell voraus. Die Qualität des Organisationsmodells ist ausschlaggebend für die Qualität der abgeleiteten Regeln. Daher ist es erforderlich, dass das Organisationsmodell alle zur Definition von Zuordnungsregeln relevanten organisatorischen Objekte enthält. Ist beispielsweise die Rolle „Krankenschwester“, die eine Anforderungen für die Ausführung einer Aktivität darstellt, nicht modelliert, kann sie folglich nicht in den Bearbeiterzuordnungsregeln berücksichtigt werden. Die abgeleiteten Regeln wären demnach falsch bzw. nicht sinnvoll.
Darüber hinaus sollte die Zuordnung von Qualifikationen zu Mitarbeitern korrekt sein. Wenn einem Bearbeiter einer Aktivität beispielsweise die notwendige Rolle „Krankenschwester“ nicht zugeordnet wird, wird die abgeleitete Zuordnungsregel verfälscht. Allerdings wäre dieser Fall weniger tragisch als wenn organisatorische Objekte ganz fehlen würden.
Neben dem Organisationsmodell, das die Population eines Organisations-Metamodells darstellt, wird auch Wissen über Abhängigkeiten zwischen organisatorischen Objekten, die aus dem Organisations-Metamodell resultieren, benötigt. Im Abschnitt 7.6.2 wird genauer auf

diese Abhängigkeiten eingegangen.

7.6 Verwendetes Organisations-Metamodell

Das verwendete Organisations-Metamodell darf keine zu hohe Hürde für die Anwendung unseres Ansatzes darstellen. Daher haben wir uns bewusst auf ein einfaches Modell mit den wesentlichen Konzepten, die in gängigen Metamodellen vorkommen, beschränkt. Das verwendete Metamodell ist in Abbildung 7.2 als Entity-Relationship-Diagramm dargestellt.

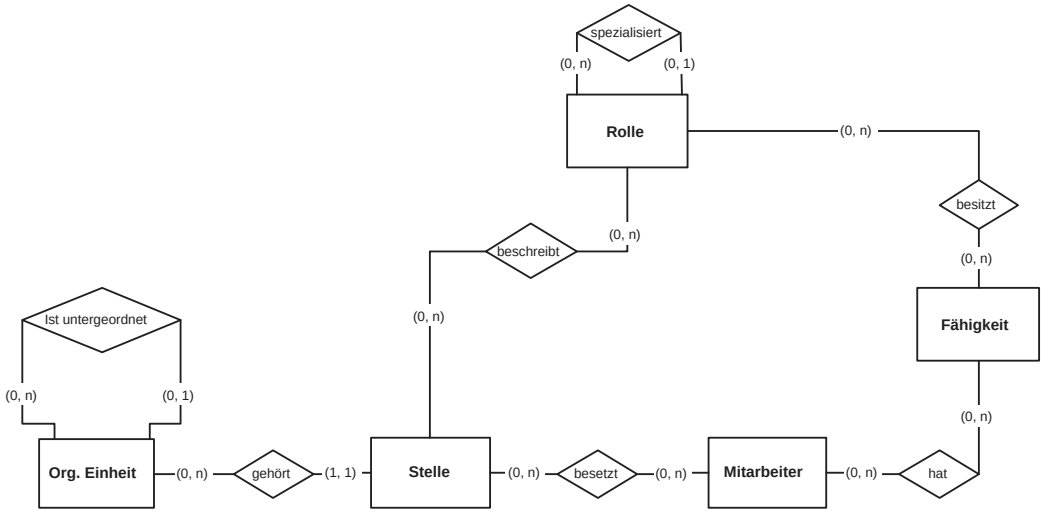

Abbildung 7.2: E-R-Diagramm des verwendeten Organisations-Metamodells

Dieses Modell ist eine vereinfachte Version des Organisations-Metamodells von Berroth [9], welches ähnlich zu Metamodellen aus Arbeiten von Kubicek und Wiedemuth-Catrinescu [56, 93] ist. Für unsere Fragestellung überflüssige Konstrukte, z.B. temporäre Konstrukte wie Projektgruppe, wurden entfernt. Dadurch ist ein sehr einfaches Modell entstanden, welches jedoch Konstrukte und Beziehungen enthält, die in anderen gängigen Metamodellen auch vorkommen. Dies ermöglicht es, dass unser Ansatz auch auf andere Organisations-Metamodelle angewendet werden kann. Auf die organisatorischen Konstrukte wird im Abschnitt 7.6.1 eingegangen.
Es ist vorgesehen, dass jede Entität in Abbildung 7.2 über Attribute verfügt. So haben alle Entitäten typischerweise das Attribut „Name“, über das sie identifiziert werden können. Die Verwendung von Attributen zur Definition von Bearbeiterzuordnungsregeln ermöglicht eine hohe Ausdrucksmächtigkeit, verändert jedoch die Situation für uns nicht wesentlich. Daher abstrahieren wir von den Attributen und beziehen uns direkt auf die organisatorischen Konstrukte. Wenn wir von der Rolle „Krankenschwester“ sprechen, so meinen wir damit diejenige Rolle, deren Name „Krankenschwester“ ist.

7.6.1 Organisatorische Konstrukte

7.6.1.1 Organisationseinheit

Organisationseinheiten beschreiben den Aufbau der Organisation. Dabei kann sich eine Organisationseinheit aus mehreren Organisationseinheiten zusammensetzen. Das heißt, die Organisationseinheiten können in hierarchischer Beziehung zueinander stehen. Typische Organisationseinheiten sind Abteilungen und Funktionsbereiche [56].

7.6.1.2 Mitarbeiter

Ein Mitarbeiter-Objekt beschreibt eine natürliche Person in der Organisation. Relationen zwischen dem Mitarbeiter-Objekt eines Mitarbeiters und anderen organisatorischen Objekten machen die Qualifikationen eines Mitarbeiters aus.

7.6.1.3 Rolle

Rollen beschreiben Aufgaben oder fassen eine Grundmenge von Qualifikationen zusammen, die jeder Rollenträger besitzt. Eine Rolle kann mit Fähigkeiten verbunden sein. In dem Fall verfügt der Rollenträger auch über die der Rolle zugeordneten Fähigkeiten. Zwischen Rollen kann außerdem eine Spezialisierungsbeziehung bestehen. Dabei erbt die speziellere Rolle alle Fähigkeiten und Rechte der allgemeineren Rolle, ähnlich wie bei objekt-orientierter Programmierung.

7.6.1.4 Stelle

Eine Stelle wird als eine Art Instantiierung eines virtuellen organisatorischen Konstrukts *Stellentyp* interpretiert, der mehrere Rollen zusammenfasst.
Im verwendeten Organisations-Metamodell erfolgt die Zuordnung eines Mitarbeiters zu einer Organisationseinheit ausschließlich über Stellen. Wir gehen davon aus, dass jeder Mitarbeiter mindestens eine Stelle besetzt.
Weiterhin lässt das Organisations-Metamodell zu, dass eine Stelle von beliebig vielen Mitarbeitern besetzt wird. Zu dieser Regelung ist anzumerken, dass der Regelfall vorsieht, eine Stelle jeweils nur durch einen Mitarbeiter zu besetzen. Die Mehrfachzuordnung muss allerdings zugelassen werden, da Aspekte wie Teilzeitarbeit berücksichtigt werden sollen [93, 9]. Damit ist Stelle ein fast ebenso personengebundenes organisatorisches Konstrukt wie der Name eines Mitarbeiters bzw. das Mitarbeiter-Objekt selbst. Da wir aber allgemeine Regeln ableiten wollen, die möglichst nicht personengebunden sind, sollen Stellen nicht zur Formulierung der Bearbeiterzuordnungsregeln verwendet werden.

7.6.1.5 Fähigkeit

Fähigkeiten können einem Mitarbeiter durch seine Rollen aber auch direkt zugeordnet werden. So kann zum Beispiel die Rolle „Krankenschwester“ mit der Fähigkeit „Blutabnehmen“ verbunden sein. Diese Fähigkeit kann einem Mitarbeiter aber auch direkt zugeordnet werden.

7.6.2 Konsequenzen aus dem Organisations-Metamodell

Aufgrund der Relationen zwischen den Entitäten im Organisations-Metamodell bestehen zwischen einigen organisatorischen Objekten Abhängigkeiten. Diese Abhängigkeiten wollen wir beim Ableiten der Regeln berücksichtigen. Die ersten drei Abhängigkeiten sind für uns jedoch nicht von Bedeutung, da Stelle nicht zur Formulierung der Bearbeiterzuordnungsregeln verwendet wird.

HatStelle(m,s) → HatRolle(m, r(s))
HatStelle(m,s) → HatFähigkeit(m, f(r(s))
HatStelle(m,s) → HatOrgEinheit(m, oe(s))
HatRolle(m,r) → HatFähigkeit(m, f(r))
HatRolle(m,r) → HatRolle(m, übergeordneteRollen(r))
HatOrgEinheit(m, oe) → HatOrgEinheit(m, übergeordneteOE(oe))

Dass diese Abhängigkeiten bestehen, lässt sich leicht an dem Organisations-Metamodell erkennen. Dadurch kann eine Menge von Abhängigkeitsbeziehungen aufgestellt werden. Im nächsten Abschnitt wird ein Beispiel-Organisationsmodell einschließlich solcher Abhängigkeiten modelliert.

7.6.3 Beispiel eines Organisationsmodells

Als Beispiel, das als Grundlage für weitere Beispiele in diesem Kapitel dient, modellieren wir ein überschaubares Organisationsmodell.
Im folgenden sind die organisatorischen Objekte in Mengenschreibweise aufgeführt. Modelliert wird die Organisationseinheit „Klinikum“ mit zwölf Mitarbeitern. Jeder Mitarbeiter besetzt eine Stelle. Insgesamt sind drei Ärzte, vier Krankenschwestern, drei medizinisch-technische Assistenten, kurz MTA, und zwei Sekretärinnen in der Organisationseinheit tätig.

Organisationseinheiten {Klinikum}
Mitarbeiter {M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12}
Stellen {1. Arzt, 2. Arzt, 3. Arzt, 1. Krankenschwester, 2. Krankenschwester, 3. Krankenschwester, 4. Krankenschwester, 1. MTA, 2. MTA, 3. MTA, 1. Sekretärin, 2. Sekretärin}
Rollen {Arzt, Ausbilder, Krankenschwester, Empfang, MTA, Buchhalter, Sekretärin}

Fähigkeiten {PC-Kenntnisse, Englisch, Türkisch, Blutabnehmen, Rezeptausstellen}

In diesem Beispiel-Modell bestehen nur Abhängigkeiten zwischen Rollen und Fähigkeiten. Die Rollen und von ihnen implizierte Fähigkeiten sind in Tabelle 7.2 aufgelistet. Eine übersichtliche Darstellung des Organisationsmodells findet sich in Tabelle 7.3. Dort sind alle Mitarbeiter und ihnen zugeordnete organisatorische Objekte aufgeführt. Mit einem * sind jeweils Fähigkeiten gekennzeichnet, die dem Mitarbeiter nicht über Rollen sondern direkt zugeordnet sind.

Rolle	*Fähigkeiten*
Arzt	Blutabnehmen, Rezeptausstellen
Krankenschwester	Blutabnehmen
Buchhalter	PC-Kenntnisse
Sekretärin	PC-Kenntnisse

Tabelle 7.2: Rollen und von ihnen implizierte Fähigkeiten

Mitarbeiter	*Org.-einheit*	*Stellen*	*Rollen*	*Fähigkeiten*
M1	Klinikum	1. Arzt	Arzt	PC-Kenntnisse*, Blutabnehmen, Rezeptausstellen, Englisch*
M2	Klinikum	2. Arzt	Arzt	PC-Kenntnisse*, Blutabnehmen, Rezeptausstellen
M3	Klinikum	3. Arzt	Arzt	PC-Kenntnisse*, Blutabnehmen, Rezeptausstellen, Englisch*
M4	Klinikum	1. Krankenschwester	Krankenschwester, Empfang, Buchhalter	PC-Kenntnisse, Blutabnehmen, Englisch*, Türkisch*
M5	Klinikum	2. Krankenschwester	Krankenschwester, Empfang, Buchhalter	PC-Kenntnisse, Blutabnehmen, Englisch*
M6	Klinikum	3. Krankenschwester	Krankenschwester, Empfang, Buchhalter	PC-Kenntnisse, Blutabnehmen
M7	Klinikum	4. Krankenschwester	Krankenschwester, Empfang, Buchhalter	PC-Kenntnisse, Blutabnehmen, Türkisch*
M8	Klinikum	1. MTA	MTA	Englisch*
M9	Klinikum	2. MTA	MTA	Englisch*
M10	Klinikum	3. MTA	MTA	
M11	Klinikum	1. Sekretärin	Sekretärin, Buchhalter, Empfang	PC-Kenntnisse, Türkisch*
M12	Klinikum	2. Sekretärin	Sekretärin, Buchhalter, Empfang	PC-Kenntnisse, Englisch*

Tabelle 7.3: Tabellarische Darstellung des Organisationsmodells

7.7 Darstellung der Bearbeiterzuordnungsregeln

Um Bearbeiterzuordnungsregeln abzuleiten muss zunächst eine geeignete Form gefunden werden, diese darzustellen. Da Bearbeiterzuordnungsregeln Anforderungen an Qualifikationen potentieller Bearbeiter darstellen, soll in diesem Zusammenhang zunächst der Begriff Qualifikation definiert werden, um Missverständnisse zu vermeiden.

Definition 7.1 (Qualifikation). *Mit Qualifikationen sind Zuordnungen zu zwischen einem Mitarbeiter-Objekt und anderen organisatorischen Objekten gemeint. So ist Rolle = „Krankenschwester“ beispielsweise eine Qualifikation, die ein Mitarbeiter haben oder auch nicht haben kann.*
Darüber hinaus ist es notwendig fordern zu können, dass potentielle Bearbeiter einer Aktivität eine Qualifikation nicht haben dürfen. Auch das Nicht-Besitzen einer Qualifikation ist somit eine Qualifikation. Diese Art der Qualifikation wollen wir als negative Qualifikationen bezeichnen, die erstere Art als positive Qualifikationen.

In [9] definiert Berroth den Aufbau von sog. *Bearbeiterformeln*, die Bearbeiterzuordnungsregeln entsprechen. Eine *Bearbeiterformeln* setzt sich aus *Bearbeiterausdrücken* zusammen. Diese werden zusammengesetzt durch Konjunktionen von Regeln der Form:
$\langle Selektor.Attributname \rangle \, \langle Vergleichsoperator \rangle \, \langle Konstante \rangle$
Die Auflösung der Bearbeiterzuordnung aus dem Beispiel 7.1 würde alle Mitarbeiter zurückgeben, die die Rolle „Krankenschwester“ innehaben.

R = „Krankenschwester“

Beispiel 7.1: Eine Bearbeiterzuordnung

Die Definition von Berroth kann im wesentlichen für diese Arbeit übernommen werden. Da wir jedoch von Attributen der organisatorischen Objekte abstrahieren, werden Selektoren direkt über einen Vergleichsoperator mit einer Konstante assoziiert. Als Konstanten kommen folglich nur die Namen der organisatorischen Objekte in Frage. Daher wird nur „=“ als Vergleichsoperator verwendet.
Da Stelle aufgrund von starker Personengebundenheit (siehe Abschnitt 7.6.1.4) nicht zur Formulierung von Bearbeiterzuordnungsregeln verwendet wird, ergeben sich aus dem verwendeten Organisations-Metamodell folgende Selektoren:

Selektor	*Organisatorisches Konstrukt*
R	Rolle
F	Fähigkeit
OE	Organisationseinheit

Tabelle 7.4: Unterstützte Selektoren

Für die Bildung komplexerer Bearbeiterzuordnungsregeln aus elementaren Bearbeiterzu-

ordnungen, wie in Beispiel 7.1, werden diese in einer Art disjunktiver Normalform (DNF) miteinander verbunden (siehe Beispiel 7.2). Die Bearbeiterzuordnungsregeln haben dadurch die Mächtigkeit aussagenlogischer Formeln.

R = „Empfang“ AND NOT R =„Krankenschwester“ OR R = „Buchhalter“

Beispiel 7.2: Eine zusammengesetzte Bearbeiterzuordnungsregel

Mit dieser Form der Darstellung der Bearbeiterzuordnungsregeln unterscheiden wir uns von Ansätzen aus [9, 93]. Diese sehen Negationen nur in Verbindung mit AND-Operatoren vor und fügen ANDNOT-Operatoren am Ende einer DNF-artigen Formel über ausschließlich positive Qualifikationen an, um unerwünschte Qualifikationen auszuschließen. Eine Formel, die dasselbe ausdrückt wie im Beispiel 7.2, würde dadurch länger und komplizierter werden.

Für die Verwendung von DNF-ähnlicher Notation spricht im allgemeinen, dass die Regeln dadurch recht intuitiv verständlich sind, vielmehr als es etwa bei der Verwendung der konjunktiven Normalform der Fall wäre. Dies ist für uns wichtig, da die extrahierten Bearbeiterzuordnungsregeln durch Experten überprüft und evt. zu weiteren Analysen verwendet werden sollen.

7.8 Lernen von Bearbeiterzuordnungsregeln

Aus den Anforderungen an die Verlaufsdaten folgt, dass für eine Aktivität x die Menge von Bearbeitern aufgelistet werden kann, die x mindestens einmal ausgeführt haben. Für jeden Mitarbeiter kann außerdem die Menge seiner Qualifikation durch seine Einordnung im Organisationsmodell bestimmt werden. Im Beispiel 7.3 sind alle Qualifikationen von Mitarbeiter M4 aufgeführt.

positive Qualifikationen(M4): {OE = „Klinikum“,
R = „Krankenschwester“, R = „Buchhalter“, R = „Empfang“,
F = „PC-Kenntnisse“, F = „Englisch“, F = „Türkisch“,
F = „Blutabnehmen“}
negative Qualifikationen(M4): {NOT R = „Arzt“, NOT R = „MTA“,
NOT R = „Ausbilder“, NOT R = „Sekretärin“,
NOT F = „Rezeptausstellen“}

Beispiel 7.3: Alle Qualifikationen des Mitarbeiters M4

Um uns dem Problem zu nähern, nehmen wir zunächst eine perfekte Prozessausführung

an. Dies soll heißen, dass x nur von befugten Mitarbeitern ausgeführt wurde und jeder Mitarbeiter, der x ausführen sollte und könnte, dies wenigstens einmal tat. Laut Annahme müssen demnach alle Bearbeiter über die zur Ausführung von x notwendigen Qualifikationen verfügen. Folglich steckt das Profil zur Ausführung von x in den Qualifikationen der Bearbeiter.
Auf der anderen Seite dürfen Nicht-Bearbeiter nicht über die Qualifikationen verfügen. Denn wenn doch, dann hätten sie, laut Annahme, x mindestens ein Mal ausgeführt.
Es kann sicherlich der Fall sein, dass auch Nicht-Bearbeiter die Anforderungen zur Ausführung einer Aktivität erfüllen. Wir müssen allerdings unterscheiden zwischen tatsächlichen Anforderung zur Ausführung einer Aktivität, im Sinne davon, dass jemand die Fähigkeit dazu hat oder dazu befugt wäre die Aktivität auszuführen, und dem tatsächlichen Profil der Bearbeiter einer Aktivität. Beides lässt sich als Bearbeiterzuordnungsregel ausdrücken. Ein Beispiel zur Veranschaulichung: Ein Arzt kann sicherlich ebenfalls die Aufgaben einer Krankenschwester, z.B. Blutabnehmen, übernehmen. Diese Aktivität wird faktisch jedoch nur von Krankenschwestern ausgeführt. Die Bearbeiterzuordnungsregel für diese Aktivität müsste demnach lauten: BZR(x): F = „Blutabnehmen“ AND R = „'Krankenschwester“ bzw. R = „Krankenschwester“, da Blutabnehmen eine von der Rolle „Krankenschwester“ implizierte Fähigkeit ist.
Insofern besteht ein Unterschied zwischen den tatsächlichen Anforderungen zur Bearbeitung einer Aktivität, hier die Fähigkeit „Blutabnehmen“, und dem Anforderungsprofil des tatsächlichen Bearbeiters. Unser Ziel ist es, das Profil der Mitarbeiter zu extrahieren, die x wirklich bearbeiten. Nur so kann der Ist-Zustand der Bearbeiterzuordnung einer Aktivität wirklich erfasst werden.

Die Aufgabe besteht nun darin, das allgemeine Profil der Bearbeiter aus den Qualifikationen der Mitarbeiter zu erkennen. Dies ist keine triviale Aufgabe, da ein Bearbeiter neben relevanten Qualifikationen typischerweise auch andere Qualifikationen aufweist. Da die Bearbeiterzuordnungsregeln disjunktive Konzepte sein können, kann auch nicht davon ausgegangen werden, dass die Schnittmenge der Qualifikationen der Bearbeiter die Lösung darstellt.

Ein naiver Ansatz, um das Profil der Bearbeiter zu erhalten, wäre eine DNF-Formel über alle Qualifikationen eines jeden Bearbeiters zu erstellen. Damit wäre die Menge der Bearbeiter so spezifisch wie möglich beschrieben. Allerdings wollen wir ein möglichst allgemein gehaltenes Profil ableiten. Darüber hinaus kann oftmals mehr als nur eine Regel abgeleitet werden, die wir dem Prozessmodellierer auch anbieten wollen.

7.8.1 Formulierung des Lernproblems

Das Problem, ein Qualifikationsprofil der Bearbeiter abzuleiten, kann als ein Problem des induktiven Lernens anhand positiver und negativer Beispiele aufgefasst werden. Anders als beim *Control Flow Mining*, wo die Existenz von negativen Beispielen eine unhaltbare Anforderung wäre, sind negative Beispiele bei unserer Fragestellung ganz natürlich gegeben. Alle

Nicht-Bearbeiter von x können als negative Beispiele aufgefasst werden.

Mit der Auffassung des Problems als Lernaufgabe fällt unsere Fragestellung in das Gebiet des *Supervised Learnings*, d.h. Lernen mit vordefinierten Klassen. *Supervised Learning* ist ein Teilgebiet des Maschinellen Lernens, welches wiederum als eine Schnittstelle zwischen der Künstlichen Intelligenz und Data Mining betrachtet wird. Für einen Überblick über das Gebiet des Maschinellen Lernens verweisen wir auf ein sehr gutes Buch von Mitchell [64].

Um eine Lernaufgabe für unsere Problemstellung zu formulieren, definieren wir zunächst die Beispiele für diesen Kontext.

Definition 7.2 (Beispiel).

Ein Beispiel ist ein Tripel $(a, m, bearbeiter(m))$*. Dabei ist* a *die untersuchte Aktivität,* m *ein Mitarbeiter aus dem Organisationsmodell,* $bearbeiter(m)$ *ist die Klassifikationsfunktion, die auf* $\{Ja, Nein\}$ *abbildet: „Ja“, wenn* m *die Aktivität* a *ausgeführt hat, „Nein“, wenn* m *die Aktivität* a *nicht ausgeführt hat. Die Menge der Beispiele ist durch die Mitarbeiter im Organisationsmodell bereits gegeben und den Daten leicht zu entnehmen.*

Da wir uns in diesem Kapitel stets auf eine untersuchte Aktivität x beziehen, gilt $a = x$ für alle verwendeten Beispiele.
Wir formulieren die Aufgabe, die Hypothese $bearbeiter\_von\_x(m)$ aus der Beispielmenge abzuleiten. $bearbeiter\_von\_x(m)$ ist eine Funktion, die die Klassifikationsfunktion $bearbeiter(m)$ approximiert. Die Form für die Hypothese ist wie folgt:
$bearbeiter\_von\_x(m) : \mathrm{BZR}(x)$
Dabei bildet $bearbeiter\_von\_x(m)$ auf „Ja“ ab, wenn m die Anforderungen der Bearbeiterzuordnungsregel erfüllt, auf „Nein“, wenn m den Anforderungen nicht genügt.

$$bearbeiter\_von\_x(m) : \mathrm{R} = \text{„Krankenschwester“ AND } \mathrm{F} = \text{„PC-Kenntnisse“}$$

Beispiel 7.4: Eine Hypothese

Die Hypothese aus Beispiel 7.4 bildet für alle Mitarbeiter, die die Rolle „Krankenschwester“ inne haben und dazu die Fähigkeit „PC-Kenntnisse“ besitzen auf „Ja“ ab. Für alle anderen Mitarbeiter auf „Nein“.

Um das formulierte Lernproblem zu lösen wird eine passende Darstellung der Daten benötigt. Dazu bietet sich eine attributbasierte Darstellung an, die im folgenden Abschnitt beschrieben wird. Im Abschnitt darauf, wird die Entscheidungsbauminduktion als ein Lernverfahren vorgestellt und anschließend auf das Lernproblem angewendet.

7.8.2 Attributbasierte Darstellung der Daten

Die Idee ist, alle möglichen Qualifikationen von Mitarbeitern als Attribute darzustellen. Für jede Organisationseinheit, jede Rolle und jede Fähigkeit wird jeweils ein Attribut erstellt, dessen Wertebereich {Ja, Nein} ist. „Ja“ für den Fall, dass der jeweilige Mitarbeiter diese Qualifikation aufweist, „Nein“ für den gegenteiligen Fall. So ist R = „Krankenschwester“ beispielsweise ein Attribut. Die Menge der Attribute entspricht der Menge der relevanten organisatorischen Objekte.
Tabelle 7.5 zeigt die attributbasierte Darstellung des Beispiel-Organisationsmodells. Aus Platzgründen werden die Attribute verkürzt dargestellt. Da alle Mitarbeiter zum gleichen Organisationsmodell gehören, wurde dieses außen vor gelassen. Die Qualifikationen der Mitarbeiter bilden die Zeilenvektoren der Tabelle.

An dieser Stelle sei angemerkt, dass sowohl die Attribut- als auch die Beispielmenge oftmals von vornherein eingeschränkt werden können. Organisatorische Objekte können häufig von Anfang an ausgeschlossen werden. Dies trifft insbesondere zu, wenn das Organisationsmodell umfangreich ist und größere Teile des Unternehmens modelliert. Wenn beispielsweise bekannt ist, dass die Fähigkeit „PC-Kenntnisse“ für die Ausführung der Aktivität x keinerlei Rolle spielt, ist es auch nicht sinnvoll, sie zur Formulierung der Bearbeiterzuordnungsregeln zu verwenden. Sie kann daher als Attribut ausgeschlossen werden. Auf der anderen Seite ist oft bereits eine Grundmenge von Qualifikationen bekannt, die eine Anforderung für die Ausführung einer Aktivität darstellen. Wenn für die Tätigkeit x beispielsweise unbedingt ein Arzt erforderlich ist, können alle Nicht-Bearbeiter, die kein Arzt sind, als Beispiele ausgeschlossen werden.

Mitarbeiter	*Kranken-schwester*	*Arzt*	*Ausbilder*	*Empfang*	*MTA*	*Buch-halter*	*Sekre-tärin*	*PC Kennt-nisse*	*Englisch*	*Türkisch*	*Blutab-nehmen*	*Rezept-ausstellen*
M1	Nein	Ja	Ja	Nein	Nein	Nein	Nein	Ja	Ja	Nein	Ja	Ja
M2	Nein	Ja	Ja	Nein	Nein	Nein	Nein	Ja	Nein	Nein	Ja	Ja
M3	Nein	Ja	Ja	Nein	Nein	Nein	Nein	Ja	Ja	Nein	Ja	Ja
M4	Ja	Nein	Nein	Ja	Nein	Ja	Nein	Ja	Ja	Ja	Ja	Nein
M5	Ja	Nein	Nein	Ja	Nein	Ja	Nein	Ja	Ja	Nein	Ja	Nein
M6	Ja	Nein	Nein	Ja	Nein	Ja	Nein	Ja	Nein	Nein	Ja	Nein
M7	Ja	Nein	Nein	Ja	Nein	Ja	Nein	Ja	Nein	Ja	Ja	Nein
M8	Nein	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Ja	Nein	Nein	Nein
M9	Nein	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Ja	Nein	Nein	Nein
M10	Nein	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein
M11	Nein	Nein	Nein	Ja	Nein	Ja	Ja	Ja	Nein	Ja	Nein	Nein
M12	Nein	Nein	Nein	Ja	Nein	Ja	Ja	Ja	Ja	Nein	Nein	Nein

Tabelle 7.5: Attributbasierte Darstellung des Organisationsmodells

7.8.3 Die Entscheidungsbauminduktion

Die Entscheidungsbauminduktion ist eines der am weitesten verbreiteten Lernverfahren. Sie ist sehr intuitiv und eignet sich für attributbasiertes Lernen von ein-parametrigen disjunktiven Konzepten. Die Entscheidungsbauminduktion wurde bereits erfolgreich auf zahlreiche Probleme angewendet, von Risiko-Prognosen bei der Kreditvergabe [64] bis hin zur Annotation von Proteinen [55]. Darüber hinaus bietet die Entscheidungsbauminduktion Mechanismen für den Umgang mit Rauschdaten und kann zudem gut grafisch dargestellt werden. Letzteres könnte für die Gestaltung einer anwenderfreundlichen Benutzeroberfläche für ein System zur Extraktion von Bearbeiterzuordnungsregeln von Belang sein. Auch Attribute mit kontinuierlichen Werten können mit der Entscheidungsbauminduktion behandelt werden. Dies ist sinnvoll, wenn etwa auch Attribute von organisatorischen Objekten zur Formulierung von Regeln herangezogen werden.
Zahlreiche Algorithmen der Entscheidungsbauminduktion wurden im Laufe der Zeit entwickelt, darunter einige sehr bekannte, wie ID3 und C4.5 von Quinlan [64, 41, 68, 69].

Die Entscheidungsbauminduktion klassifiziert Beispiele, indem sie einen Baum aufbaut. Jeder innere Knoten entspricht einem Test über ein Attribut der Beispiele. Die von einem Knoten ausgehenden Pfade entsprechen möglichen Werten des Attributs. An der Wurzel wird angefangen, ein Attribut auszuwählen. Die Beispiele werden, je nach ihrem Attributwert, entsprechend den Kinderknoten zugeordnet. Befinden sich an einem Knoten nur noch Beispiele einer Klasse der Klassifikationsfunktion, in unserem Fall *bearbeiter(m)* mit den Klassen „Ja“ oder „Nein“, so ist dieser Knoten ein Blattknoten. Ansonsten wird die Separierung weiter rekursiv vorgenommen, bis alle Knoten Blattknoten sind oder bereits alle Attribute getestet wurden.

Aus dem gelernten Baum können Regeln in DNF-Form bzw. in IF-THEN-Form auf einfache Weise abgeleitet werden. Ein Pfad vom Wurzelknoten zu einem Blattknoten der Zielklasse stellt eine Konjunktion über die Attributwerte dieses Pfades dar. Verschiedene Blätter mit Beispielen der Zielklasse bilden die Disjunktionsglieder.

Im Listing 7.1 ist ein allgemeiner Algorithmus für die Entscheidungsbauminduktion, angelehnt an dem Algorithmus aus [12], aufgeführt. Dieser beinhaltet auch Optimierungsoptionen, z.B. ein Stop-Kriterium, das Auskunft darüber gibt, wann ein Knoten nicht mehr expandiert werden sollen. Darüber hinaus sind hier auch Operationen für eine nachträgliche Optimierung enthalten. Typischerweise sind dies Pruning-Operationen, wodurch der Baum niedriger und die resultierenden Regeln kompakter werden.

```
1
2 C: Menge der Beispiele
3 I: Menge der Attribute
4 N: Ein Knoten
5 rules: Menge von Regeln
```

```
6  eval(): Funktion zur Bewertung von Attributen
7  stop(): Funktion, die zurückgibt, wann ein Knoten nicht mehr
8                  expandiert wird
9
10
11 create_tree(){
12     tree = induce_tree(C, I)
13     // Pruning-Operationen
14     post-process(tree)
15     rules = generate_rules(tree)
16 }
17
18 induce_tree(C, I){
19     if(stop(C)= true){
20         N = create_leaf_node(C)
21     }
22     else{
23         {attribute, Cleft, Cright} = get_best_att_and_partition(C, I)
24         N = create_internal_node(C, attribute)
25         left-sub-tree(N) = induce_tree(Cleft, I/attribute)
26         right-sub-tree(N) = induce_tree(Cright, I/attribute)
27     }
28 }
```

Listing 7.1: Ein allgemeiner Entscheidungsbaumalgorithmus

7.8.4 Anwendung der Entscheidungsbauminduktion

Wir wenden die Entscheidungsbauminduktion auf unser Lernproblem an. Dabei sind einige Aspekte zu berücksichtigen, auf die wir im folgenden Schritt für Schritt eingehen wollen.

7.8.4.1 Minimalität der Hypothesen und Occam's Razor

Ziel unserer Arbeit ist es, Regeln abzuleiten, die ein möglichst minimales Profil der Bearbeiter einer Aktivität darstellen. Solche Regeln werden durch niedrige Bäume repräsentiert, die wenig Attribute entlang eines Pfades besitzen. Dies bedeutet, dass die Disjunktionsglieder der resultierenden Regeln weniger Attribute und damit weniger Bedingungen an die Bearbeiter einer Aktivität stellen.

Die Präferenz von kurzen gegenüber längeren Hypothesen war Thema vieler wissenschaftlicher Diskussionen. In diesem Zusammenhang ist auch das sog. *Occam's Razor* zu nennen. Dieses besagt, dass beim Vorliegen verschiedener Hypothesen, die kürzeste gewählt werden soll, die mit den Daten konsistent ist [64, 8].
Für unseren Fall sind kürzere Regeln den längeren im allgemeinen vorzuziehen. Kürzere Regeln sind zum einen leichter verständlich als längere. Zum anderen ist die Wahrscheinlichkeit, dass zufällige, irrelevante Anforderungen in der Regel vorkommen, bei einer weniger kompakten Regel größer.

Das Problem, minimale Regeln zu finden, ist allerdings ein NP-hartes Problem. Dazu ist

eine erschöpfende Suche auf dem Suchraum aller möglichen Bäume notwendig. Ein guter Kompromiss wird getroffen, indem Heuristiken zur Steuerung der Suche verwendet werden. Besonders bekannte Heuristiken sind *Gain*, verwendet von ID3 und C4.5, und *Gain Ratio*, verwendet von C4.5 [64, 68, 69, 53]. Dabei basiert *Gain Ratio* auf *Gain*. *Gain* wiederum basiert auf der Berechnung von Entropien und greift damit auf Methoden der Informationstheorie zurück. Die Entropie ist ein Index dafür, wie homogen die Beispielmenge ist. Auf unseren Testdaten lieferte *Gain* die besseren Ergebnisse. Für ausführlichere Informationen über Entropie im Zusammenhang mit *Gain* wird auf [64] verwiesen.
Die Formeln zur Berechnung der Entropie und des *Gain*-Wertes sind unten aufgeführt. S steht für die Beispielmenge an einem Knoten, a für ein Attribut. p_{Ja} und p_{Nein} stehen für den Anteil der Beispiele der Bearbeiter bzw. Nicht-Bearbeiter. S_{Ja} und S_{Nein} stehen für die Beispielmenge, die jeweils den beiden Kinderknoten des Knoten S zugeordnet sind.
Der *Gain*-Wert wird für jedes zur Separierung in Erwägung gezogene Attribut berechnet. Das Attribut mit dem besten *Gain*-Wert wird schließlich für den Separierungsschritt ausgewählt. Damit versucht der Entscheidungsbaumalgorithmus in jedem Schritt eine maximale Reduzierung der Entropie zu erreichen. Der induktive *Bias* geht dahin, niedrige Bäume zu bevorzugen, bei denen die besten Separierungsschritte nahe bei der Wurzel liegen.

$$Entropie(S) = -p_{Ja} \log_2 p_{Ja} - p_{Nein} \log_2 p_{Nein}$$

$$Gain(S, a) = Entropie(S) - \frac{|S_{Ja}|}{|S|} Entropie(S_{Ja}) - \frac{|S_{Nein}|}{|S|} Entropie(S_{Nein})$$

Tabelle 7.6 listet die Beispielmenge für das Beispiel-Organisationsmodell aus Abschnitt 7.6.3 auf. Der Entscheidungsbaum für diese Beispielmenge unter Verwendung von *Gain* als Heuristik ist in Abbildung 7.3 zu sehen. Die grau unterlegten Flächen markieren die entscheidenen Blattknoten.

Mitarbeiter m	*bearbeiter(m)*
M1	Nein
M2	Nein
M3	Nein
M4	Ja
M5	Ja
M6	Nein
M7	Ja
M8	Nein
M9	Nein
M10	Nein
M11	Ja
M12	Ja

Tabelle 7.6: Eine Beispielmenge

Aus dem Entscheidungsbaum in Abbildung 7.3 kann folgende Bearbeiterzuordnungsregel abgeleitet werden:

BZR(x): F = „Englisch“ AND R = „Empfang“ OR F = „Türkisch“ AND NOT F = „Englisch“ AND R = „Empfang“

Eine Darstellung der Regel in IF-THEN-Form ist wie folgt:

IF F = „Englisch“ AND R = „Empfang“ THEN bearbeiter(m) = „Ja“
IF F = „Türkisch“ AND NOT F = „Englisch“ AND R = „Empfang“ THEN bearbeiter(m) = „Ja“

Aufgrund des binären Charakters der Attribute können abgeleitete Regeln oft logisch vereinfacht werden, z.B. mit Hilfe von Karnaugh-Veitch-Diagrammen. So können wir auch hier die Regel vereinfachen:

BZR(x): F = „Englisch“ AND R = „Empfang“ OR F = „Türkisch“ AND R = „Empfang“

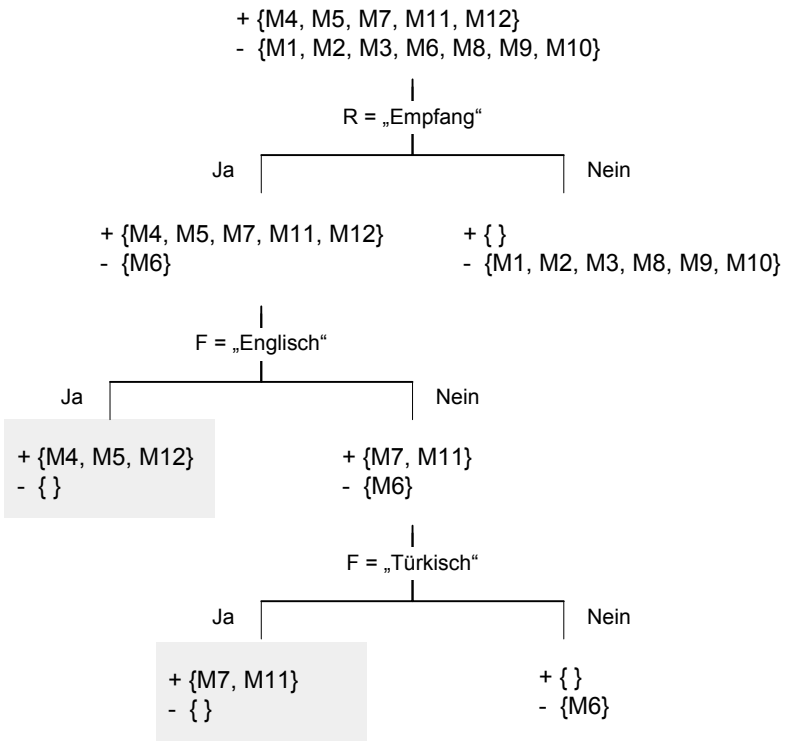

Abbildung 7.3: Entscheidungsbaum für die Beispieldaten aus Tabelle 7.6

7.8.4.2 Multiple Regeln

Häufig ist es der Fall, dass mehr als eine minimale Regel existiert. Das trifft auch für die Beispielmenge aus Tabelle 7.6 zu. Da es nicht möglich ist, zu entscheiden, welche der möglichen Regeln tatsächlich für den Prozessmodellierer interessant ist, sollen ihm die Alternativen angeboten werden.
Ähnlich wie auch schon das Problem der minimalen Regeln, müsste hierzu auch der komplette Suchraum aller möglichen Bäume überprüft werden. Ein Kompromiss ist es in diesem Fall, die ausgewählten Attribute zu variieren. Das heißt, anstatt nur das Attribut mit dem besten *Gain*-Wert zu verwenden, werden auch Bäume mit dem zweitbesten, drittbesten Attribut bis hin zu einer vordefinierten Beschränkung entwickelt.
Dazu führen wir die Parameter K und D ein. Es werden die K-besten Attribute zur Separierung verwendet. D gibt an, bis auf welcher Tiefe dies ausgeführt werden soll. Eine Einstellung der Parameter mit $K = 2$ und $D = 1$ würde bedeuten, dass die beiden Attribute mit den besten *Gain*-Werten zur Separierung verwendet werden sollen. Dies soll aber nur auf der Ebene der Wurzel geschehen.
Mit dieser Methode konnten wir auf unseren Testdaten gute Ergebnisse erzielen. Abbildung 7.4 zeigt einen alternativen Baum ($K = 2$, $D = 1$) für die Beispielmenge aus Tabelle 7.6. Die

Regel, die sich daraus ergibt, ist:

BZR(x): R = „Buchhalter“ AND F = „Englisch“ OR R = „Buchhalter“ AND F = „Türkisch“

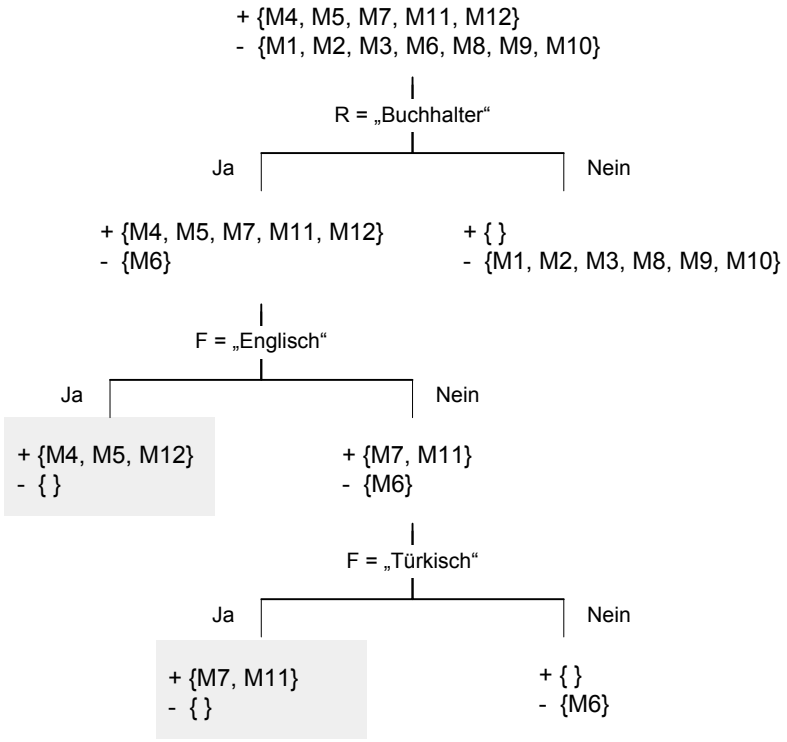

Abbildung 7.4: Alternativer Entscheidungsbaum für die Beispieldaten aus Tabelle 7.6

7.8.4.3 Abhängigkeiten zwischen Attributen

Aufgrund des verwendeten Organisations-Metamodells können sich taxonomische Beziehungen zwischen organisatorischen Objekten ergeben (siehe auch Abschnitt 7.6.2). Zum einen können Organisationseinheiten in einer hierarchischen Beziehung zueinander stehen. So lässt sich zum Beispiel die Organisationseinheit „Klinikum“ in die Untereinheiten „Verwaltung“ und „Behandlung“ gliedern. Andere taxonomische Beziehungen können bei Rollen vorkommen, die in einer Spezialisierungsbeziehung zueinander stehen können. Die Rolle „Krankenschwester“ lässt sich zum Beispiel zur Rolle „Krankenschwester mit Medizinschrankbefugnis“ spezialisieren. Letztere stellt eine Krankenschwester mit erweiterten Rechten dar.

Relevant ist für uns jedoch lediglich, dass die evt. auftretenden Beziehungen eine Implikationssemantik bewirken. Da eine Rolle lediglich die Spezialisierung einer einzigen Rolle sein

kann, kann von einer Rolle stets auf übergeordnete Rollen geschlossen werden, falls eine solche existiert. Analoges gilt für Organisationseinheiten. Man beachte dabei, dass diese Implikationsbeziehung sich rekursiv fortsetzen kann.

Der Umgang mit solchen Beziehungen ergibt sich bei unserem Vorgehen automatisch daraus, dass für jedes organisatorische Objekt auch ein Attribut erzeugt wird. Eine übergeordnete Organisationseinheit wird also genau so behandelt, wie die ihr untergeordneten Organisationseinheiten. Wenn beispielsweise ein Mitarbeiter einer untergeordneten Organisationseinheit zugeordnet ist, sind seine Attributbelegungen für die übergeordneten Organisationseinheiten ebenfalls „Ja“. Für die Entwicklung des Entscheidungsbaumes ergeben sich keine Unterschiede.

Neben den Abhängigkeiten, die sich aus den Taxonomien ergeben können, können sich aus dem Organisations-Metamodell noch weitere Abhängigkeiten ergeben. Da Fähigkeiten einem Mitarbeiter sowohl direkt als auch über Rollen zugeordnet werden können, können wir von den Rollen eines Mitarbeiters auf seine Fähigkeiten schließen. Die umgekehrte Richtung gilt jedoch nicht.

Alle Abhängigkeitsbeziehungen, die aufgrund des Organisations-Metamodells auftreten können, wurden im Abschnitt 7.6.2 genannt. Unter Ausnutzung der Abhängigkeiten können zusätzliche Optimierungen am Entscheidungsbaumverfahren vorgenommen werden.
Wurde ein Attribut entlang eines Pfades zur Separation verwendet, ist es nicht sinnvoll, Attribute, die von dem verwendeten Attribut impliziert werden, zu testen. Diese würden keine weitere Separierung auf diesem Pfad bewirken. Daher können sie gleich nach der Auswahl des Attributes aus der Menge der noch nicht verwendeten Attribute entfernt werden. Dies erspart die Kosten für überflüssige Tests. Wenn beispielsweise die Rolle „Krankenschwester“ zur Separierung verwendet wurde, muss die von ihr implizierte Fähigkeit „Blutabnehmen“ nicht mehr getestet werden.

Da eine Umkehrung der Implikation, z.B. von Fähigkeiten auf Rollen zu schließen, nicht möglich ist, kann es vorkommen, dass eine Rolle zusammen mit von ihr implizierten Fähigkeiten entlang eines Pfades vorkommt. Dies geschieht, wenn die Fähigkeiten vor den Rollen als Attribut zur Separierung ausgewählt werden. In einer nachverarbeitenden Funktion könnten diese implizierten Fähigkeiten jedoch aus den Regeln entfernt werden.

7.8.4.4 Umgang mit Rauschdaten

Die perfekte Prozessausführung, von der wir am Anfang von Abschnitt 7.8 ausgegangen sind, war nur eine Annahme, um uns dem Problem zu nähern. Eine perfekte Prozessausführung kann natürlich nicht vorausgesetzt werden. Daher müssen Mechanismen in das Lernverfahren integriert werden, um auch mit Ausnahmefällen umgehen zu können. Dazu ist es nötig, zu analysieren, mit welchen Fällen gerechnet werden muss.

Pruning auf Basis positiver Beispiele: Zum einen kann es notwendig sein, den Baum anhand von positiven Beispielen zu beschneiden. So kann es vorkommen, dass einige Bearbeiter einer Aktivität x diese nur ausnahmsweise, z.B. als Vertretung eines anderen Mitarbeiters, ausgeführt haben (siehe Verlaufsdaten in Tabelle 7.7). Diese Ausnahmebearbeiter müssen nicht zwingend die Anforderungen zur Bearbeitung von x erfüllen. In diesem Fall kann die Tatsache zu Nutze gezogen werden, dass Ausnahmebearbeiter eine Aktivität typischerweise nur mit einer geringen Frequenz ausführen. Ein ähnlicher Fall kann auch bei Fehlern beim Protokollierungsvorgang auftreten, z.B. wenn einer Ausführung von x ein falscher Bearbeiter zugeordnet wird.
Bei positiven Beispielen kann die Häufigkeit, mit der ein Bearbeiter die Aktivität ausgeführt hat, verwendet werden, um die Relevanz der Beispiele zu bewerten. Dazu erweitern wir die Definition eines Beispiels.

Definition 7.3 (Erweitertes Beispiel).

Ein Beispiel ist ein Tripel $(a, m, bearbeiter(m))$. *a, m und $bearbeiter(m)$ sind wie bereits für Beispiel im Abschnitt 7.8.1 definiert. Der Unterschied liegt darin, dass für jede protokollierte Ausführung von a ein Beispiel erzeugt wird. Daher können zwei verschiedene Beispiele b_i und b_j den gleichen Mitarbeiter m haben.*

Dadruch, dass jede Ausführung von x als ein positives Beispiel gewertet wird, wird der Entscheidungsbaum robuster gegenüber Ausnahmefällen.

Tabelle 7.7 listet eine Beispielmenge auf. Aus Platzgründen belegt jeder Bearbeiter nur einen Eintrag in der Tabelle. Die Frequenz, mit der ein Mitarbeiter x ausgeführt hat, ist in der Spalte $f(m)$ angegeben. Unter den Bearbeitern fallen M9 und M10 auf, die die Aktivität jeweils nur einmal ausgeführt haben. Abbildung 7.5 zeigt den Entscheidungsbaum für diese Beispielmenge. M10 kann mit den vorhandenen Attributen nicht von M8 separiert werden. Es entsteht eine Baumstruktur für die „richtigen“ Bearbeiter und eine länglichere Baumstruktur für die Ausnahmebearbeiter. Um Regeln abzuleiten, die nicht von den Ausnahmen beeinflusst werden, muss der Baum vereinfacht werden.

Die Entscheidungsbauminduktion kennt zwei Möglichkeiten zur Vereinfachung von Bäumen.

Mitarbeiter m	*bearbeiter(m)*	*f(m)*
M1	Nein	0
M2	Nein	0
M3	Nein	0
M4	Ja	32
M5	Ja	29
M6	Nein	0
M7	Ja	31
M8	Nein	0
M9	Ja	1
M10	Ja	1
M11	Ja	26
M12	Ja	30

Tabelle 7.7: Eine Beispielmenge mit Ausnahmebearbeitern

Zum einen können Kriterien angegeben werden, ab wann ein Knoten nicht mehr expandiert werden soll. Diese Technik nennt sich *Pre-Pruning*. Zum anderen kann der Baum vollständig entwickelt werden und in einem nachgelagerten Verfahren, dem sog. *Post-Pruning*, nach definierten Kriterien gestutzt werden. *Post-Pruning* ist das häufiger eingesetzte Verfahren. Beim Pre-Pruning besteht der Vorteil, weniger Knoten expandieren zu müssen. Allerdings liegt der Vorteil von *Post-Pruning* darin, dass durch das Entwickeln des gesamten Baumes auch Informationen über die Ausnahmebearbeiter gewonnen wird, die für den Modellierer ebenfalls vom Interesse sein könnten. Für einen ausführlichen Überblick über Pruning-Strategien wird auf die Arbeit von Breslow und Aha in [12] verwiesen.

Um den Baum zu beschneiden führen wir zwei Parameter ein:
F : Grenzwert für die prozentuale Beteiligung der Bearbeiter an einem Knoten N an der Ausführung der Aktivität x
A_p : Grenzwert für die Anzahl der Bearbeiter, die einem Knoten N zugeordnet sind

Sei $f(N)$ die Anzahl aller positiven Beispiele, die einem Knoten N zugeordnet sind. Dann entspricht $f(N)$ der Anzahl der Ausführungen von x, die dem Knoten N zugeordnet sind. $a_p(N)$ sei die Anzahl der unterschiedlichen Bearbeiter an dem Knoten N. $a_p(N) \leq f(N)$ und $a_p(N) = f(N)$ für den Fall, dass jeder Bearbeiter x exakt einmal ausgeführt hat.
$f_\#$ sei die Anzahl aller protkollierten Ausführungen von x. Dann wird aus dem Knoten N ein Blattknoten erzeugt, falls gilt: $\frac{f(N)}{f_\#} < F$ und $a_p(N) < A_p$
Befinden sich an dem Knoten N mehr Nicht-Bearbeiter als Bearbeiter, so wird N mit dem Zielprädikat „Nein“ markiert.

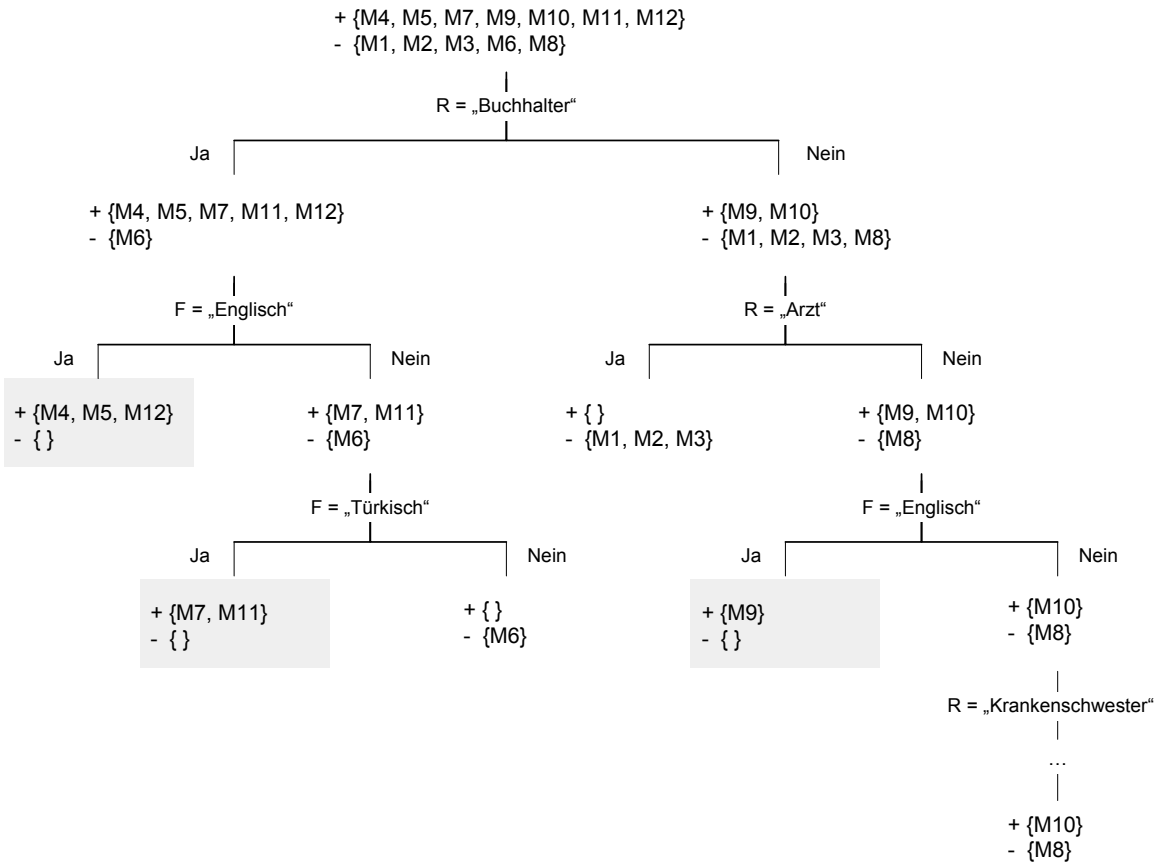

Abbildung 7.5: Entscheidungsbaum für die Beispieldaten aus Tabelle 7.7

Wenden wir das auf den Baum aus Abbildung 7.5 an ($F = 0,02, A_p = 2$) an, so kann der rechte Teilbaum gestutzt werden, da M9 und M10 mit weniger als 2% an der Ausführung von x beteiligt sind. Aus dem Knoten des rechten Teilbaums auf Ebene 2 kann ein Blattknoten gemacht werden. Damit erhalten wir dieselbe Baumstruktur wie in Abbildung 7.3.
Es sei noch angemerkt, dass mit Hilfe des Grenzwertparameters F auch das Herausfiltern von positiven Beispielen mit einer geringen Bearbeitungsfrequenz von vornherein möglich ist.

Pruning auf Basis negativer Beispiele: Anders als bei positiven Beispielen können für negative Beispiele keine Frequenzen zur Bewertung ihrer Relevanz herangezogen werden. Das Beschneiden des Baumes auf Basis negativer Beispiele kann jedoch notwendig sein. Wenn Nicht-Bearbeitern bei der Modellierung des Organisationsmodells beispielsweise falsche Qualifikationen zugeordnet werden, kann es sein, dass sie schwieriger von den Bearbeitern zu separieren sind.
Um mit solchen Fällen umgehen zu können, führen wir einen Grenzwertparameter A_n für

negative Beispiele an einem Knoten N ein.

Sei $a_n(N)$ die Menge der Nicht-Bearbeiter, d.h. die Menge der negativen Beispiele, die einem Knoten N zugeordnet sind. Dann wird aus N ein Blattknoten generiert und mit dem Zielprädikat „Ja“ markiert:
$a_n(N) < A_n$

Fehler im Organisationsmodell: Mit der Verwendung des Organisationsmodells können Fehler im Organisationsmodell, neben Fehlern die von der Prozessausführung herrühren, ebenfalls die Qualität der abgeleiteten Regeln beeinflussen. Fehler im Organisationsmodell sind schwierig zu erkennen und zu behandeln. Wenn für einen regelmäßigen Bearbeiter zum Beispiel entscheidende Qualifikationen nicht modelliert wurden, obwohl er diese besitzt, kann die Frequenz kein Indiz für ein Ausnahmefall sein. Die abgeleiteten Regeln würden nicht die tatsächlichen Anforderungen wiedergeben, jedoch die durch das Organisationsmodell implizierte Situation. Einige Modellierungsfehler können, wie in den letzten beiden Abschnitten bereits beschrieben, unter Verwendung von Grenzwerten ausgeglichen werden.

7.8.4.5 Integration in die Entscheidungsbauminduktion

Im folgenden wird der allgemeine Algorithmus aus Listing 7.1 um die in den letzten Abschnitten diskutierten Aspekte erweitert.
In einem nachgelagerten Verarbeitungsschritt wird der Baum beschnitten. Aus der resultierenden Regelmenge können mehrfach auftauchende Regeln sowie Regeln, zu denen allgemeinere Regeln in der Regelmenge vorkommen, entfernt werden. Bei der Entwicklung des Baumes können weitere Optimierungen vorgenommen werden. Wird ein Pfad erzeugt, der einer bereits existierenden Regel aus der Regelmenge entspricht oder von einer Regel der existierenden Regelmenge impliziert wird, kann die Entwicklung dieses Pfads abgebrochen werden.

```
1  C: Menge der Beispiele
2  I: Menge der Attribute
3  N: Ein Knoten
4  D: Level für Backtracking
5  K: k-Beste Attribute
6  F: Grenzwert für Anzahl positiver Beispiele
7
8  Ap: Grenzwert für Anzahl Bearbeiter
9  An: Grenzwert für Anzahl Nicht-Bearbeitern
10 f(N): Anzahl positiver Beispiele an Knoten N
11 ap(N): Anzahl Bearbeiter an Knoten N assoziiert
12 an(N): Anzahl Nicht-Bearbeiter an Knoten N assoziiert sind
13 Trees: Menge von Bäumen
14 Rules: Menge von Regeln
15 eval(): Funktion zur Bewertung von Attributen
16 stop(): Funktion, die zurückgibt, wann ein Knoten nicht mehr expandiert wird
17
18 create_trees() {
```

```
19      trees = make_all_trees(D, C, I)
20      // Pruning-Operationen
21      post-process(trees)
22      rules = generate_rules(trees)
23
24    // Doppelte Regeln und Regeln, zu denen es allgemeinere Regeln gibt,
          entfernen
25      rules = remove_duplicates_and_special_rules(rules)
26  }
27
28  make_all_trees(D, C, I){
29    trees = {}
30      for j = 1..K {
31          trees = trees UNION induce_tree(j, D-1, C, I)
32      }
33      return trees
34  }
35
36  induce_tree(K, D, C, I){
37      // C separiert oder keine Attribute übrig
38      if(stop(C)= true){
39          N = create_leaf_node(C)
40      }
41      // C noch nicht separiert
42      else{
43          {attribute, CYes, CNo} = get_k_best_att_and_partition(k, C, I, eval
                ())
44          create_internal_node(C, attribute)
45          // von attribute abhängige Attribute aus I entfernen
46          I = remove_dependant_atts(attribute)
47          //
48          if(D > 0){
49              // jede Kombination eines Teilbaums aus
50              // left-sub-trees und eines Teilbaums aus
51              // right-sub-trees ergeben legale Teilbäume
52              left-sub-trees(N) = make_all_trees(D, CYes, I/attribute)
53              right-sub-trees(N) = make_all_trees(D, CNo, I/attribute)
54          }
55          else{
56              left-sub-tree(N) = induce_tree(1,0, CYes, I/attribute)
57              right-sub-tree(N) = induce_tree(1,0, CNo, I/attribute)
58          }
59      }
60  }
61
62  post_process(trees){
63      while trees not empty{
64          tree = get_next_tree(trees)
65          // Pruning auf Basis positiver Beispiele
66          if(tree contains node N with (ap(N) < Ap) and   (f(N) < F)){
67              N = create_leaf_node(N)
68          }
69          // Pruning auf Basis negativer Beispiele
70          if(tree contains node N with (an(N) < An)){
71              N = create_leaf_node(N)
72          }
73      }
74  }
```

Listing 7.2: Erweiterter Entscheidungsbaumalgorithmus

7.9 Zusammenfassung und Ausblick

7.9.1 Verbesserungsmöglichkeiten und alternative Vorgehensweisen

Mit der Verwendung einer lokalen Suche, die in jedem Schritt ein lokales Maximum zu erreichen versucht, laufen wir Gefahr, nicht die allgemeinsten Regeln zu finden. Allerdings ist dies ein Kompromiss, um die Komplexität gering zu halten. Bei kleinen Datenmengen, so wie es oft der Fall sein wird, kann auch eine erschöpfende Suche durchgeführt werden.
Weitere Verbesserungen am Entscheidungsbaumverfahren können vorgenommen werden. *Gain* als eingesetzte Heuristik macht keinen Unterschied, ob Bearbeitern negative oder positive Qualifikationen zugeordnet werden. Das heißt, es spielt für die *Gain*-Heuristik zum Beispiel keine Rolle, ob die Regel BZR(x): NOT R = „Krankenschwester“ oder die Regel BZR(x): R = „Arzt“ die Bearbeiter identifiziert. Eine Formulierung der Bearbeiterzuordnungsregeln über positive Qualifikationen wird jedoch eher der Normalfall sein. Daher könnte eine Heuristik, die dies berücksichtigt, qualitativ bessere Regeln ableiten.
Eine alternative Vorgehensweise könnte auch für den Umgang mit Taxonomien und anderen Abhängigkeiten erwogen werden. Die Verwendung einer Reasoning-Komponente wäre eine Alternative zu der attributbasierten Darstellung, die wir verwenden [78].
Auch die Verwendung von bekannten Strategien zur Bewertung eines Baumes kann hilfreich sein. In diesem Zusammenhang ist die *Cross Validation* zu nennen, bei der die Beispielmenge in Trainings- und Testmengen eingeteilt werden. Anhand der Fehlerrate auf der Testmenge kann die Qualität des Baumes bestimmt werden. Dies kann auch eine Hilfe sein, um zu entscheiden, ob das Beschneiden des Baumes sinnvoll ist. Vor allem bei größeren Datenmengen würde sich eine Unterteilung der Daten in Trainings- und Testmengen anbieten. Für weitere Informationen dazu verweisen wir auf [69, 53, 41].

Die Entscheidungsbauminduktion ist sicherlich nicht das einzige Verfahren, das für die untersuchte Fragestellung eingesetzt werden kann. Neben der Entscheidungsbauminduktion gibt es noch eine Reihe von Verfahren, um Klassifikationsregeln abzuleiten [46, 24, 60]. Die Entscheidungsbauminduktion bietet jedoch die Möglichkeit einer grafischen Darstellung. Vor allem für die Entwicklung und Gestaltung einer entsprechenden Anwendungssoftware könnte dies interessant sein. Eine interessante Alternative zur Entscheidungsbauminduktion, die das Problem von einer anderen Richtung angehen würde, ist die Ableitung von *Association Rules* [41, 80] aus der Datenmenge. *Association Rules* sind Regeln der Form: IF R = „Krankenschwester“ AND F = „PC-Kenntnisse“ THEN bearbeiter(m) = Ja mit einer Wahrscheinlichkeit von p. Bei der Suche nach *Association Rules* geht es darum, häufige Mustern in Daten zu finden und diese mit einer Zielfunktion zu verbinden. Insbesondere sind *Association Rules* für die Ableitung der tatsächlichen Anforderungen einer Aktivität geeignet, weniger für das Ableiten des Profils der tatsächlichen Bearbeiter. Zukünftige Arbeiten sollten *Association Rules* als alternative Vorgehensweise erwägen.

7.9.2 Weiterführende Fragestellungen

Da Process Mining ein interaktiver Prozess ist, der Evaluationen durch Modellierungsexperten erfordert, spielen auch die Darstellung der Regeln sowie die Interaktionsmöglichkeiten des Modellierungsexpertens eine große Rolle. So wäre es zum Beispiel sinnvoll für Disjunktionsglieder einer Regel jeweils die Menge der Bearbeiter anzugeben, die mit diesem Disjunktionsglied erfasst werden. Das ermöglicht dem Modellierer ein besseres Verständnis der Regeln und gibt dem Modellierer eine breitere Informationsgrundlage für die Evaluierung der Regeln.
Auch die Möglichkeiten zur Einschränkung der relevanten Beispiel- sowie der Attributmenge (siehe Abschnitt 7.8.2) sollte eine Anwendung zur Ableitung von Bearbeiterzuordnungsregeln unterstützen. Auch weitere Optionen, z.B. Überprüfung ob eine gegebene Regel konsistent mit der Datenmenge ist, können für eine solche Anwendung sinnvoll sein.
Beim Erhalten vieler Regeln kann die Analyse der am häufigsten auftretenden Attribute der Regelmenge eine zusätzliche Hilfe zur Bewertung der Regeln sein. Insofern sollte eine entsprechende Anwendung diese Option ebenfalls unterstützen.

Im Rahmen dieser Arbeit haben wir uns auf wesentliche organisatorische Konstrukte als Selektoren beschränkt. Selektoren, die rekursive Beziehungen enthalten, z.B. Mitarbeiter aus allen Organisationseinheiten, die der Organisationseinheit „Behandlung“ übergeordnet sind, sind mit unserem Ansatz nicht ohne weiteres möglich.
Auch abhängige Bearbeiterzuordnungen, z.B. Bearbeiter von x muss auch y ausgeführt haben, wären eine sinnvolle Erweiterung. Wenn wir den Sachverhalt „Mitarbeiter m hat y in dieser Instanz ausgeführt“ als ein Attribut, etwa $y(\mathrm{m})$, ausdrücken können, könnte eine Lösung darin liegen, mit $y(\mathrm{m})$ wie mit jedem anderen Attribut zu verfahren. Zumindest für den Fall, dass für die Aktivität derselbe Bearbeiter verlangt wird wie y, könnte die Lösung auch mit unserem Ansatz gefunden werden.
Weiterhin interessant ist auch die Verknüpfung verschiedener Aspekte eines Prozesses. Eine Analyse, die die Informationsperspektive mit den Bearbeiterzuordnungsregeln in Verbindung setzt, z.B. bei einer Kreditsumme von mehr als 50,000 Euro bestehen andere Bearbeiteranforderungen als bei geringeren Kreditsummen, könnte zum Beispiel von Interesse sein.

7.9.3 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein bisher nicht beachteter Aspekt von *Process Mining* angegangen. Mit dem Wissen über den Ist-Zustand der Bearbeiterzuordnung von Aktivitäten können die Ergebnisse aus *Control Flow Mining* um zusätzliche Informationen zu den Aktivitäten erweitert werden. Dies erlaubt eine vollständigere Darstellung davon, wie ein Prozess tatsächlich ausgeführt wurde und daher auch eine vollständigere Grundlage, um den Prozess zu analysieren.

Wir haben gezeigt, dass das Problem Bearbeiterzuordnungsregeln abzuleiten als ein Lernproblem aufgefasst werden kann. Um dieses Problem zu lösen wurde eine geeignete Darstellung der Daten gefunden und ein bekanntes Lernverfahren, die Entscheidungsbauminduktion, adaptiert. Die abgeleiteten Regeln identifizieren die Bearbeitermenge und können für weitere Analysen verwendet oder zur *Staff Resolution* eingesetzt werden.

Unsere Fragestellung unterscheidet sich wesentlich von *Control Flow Mining* darin, dass wir noch Zusatzwissen, in Form des Organisationsmodells, verwenden. Dadurch sind die Ergebnisse nicht nur von der Qualität der Verlaufsdaten sondern im höchsten Maße auch von der Qualität des Organisationsmodells abhängig. Es ist daher notwendig, dass das Organisationsmodell alle notwendigen Objekte und Beziehungen enthält, die für die Bearbeiterzuordnung eine Rolle spielen. Ein unvollständiges Organisationsmodell führt zwangsläufig zu verfälschten Regeln. Zwar geben die Regeln das Profil der Bearbeiter in Anbetracht des verwendeten Organisationsmodells wieder, jedoch sind sie nicht unbedingt sinnvoll bzw. aufschlussreich für den Modellierer. Solche Regeln erlauben zumindest Rückschlüsse auf die Qualität des Organisationsmodells.
Obwohl wir nur ein sehr einfaches Organisations-Metamodell voraussetzen, wird die Bereitstellung eines vollständigen Organisationsmodells sicherlich dennoch einer der größten Hürden für die Anwendung von *Staff Assignment Mining* sein.

Wie auch bei anderen Ansätzen zu *Process Mining* ist es schwierig, die Nützlichkeit der Ergebnisse zu evaluieren. Dazu bedarf es noch vieler Tests auf Daten aus realen Umgebungen. Auch an interessanten weiterführenden Fragestellungen mangelt es nicht. Angefangen von Möglichkeiten der weiteren Analyse und Verwendung der Bearbeiterzuordnungen, z.B. in Verbindung mit Lastverteilung, bis hin zu Anforderungen für eine sinnvolle, interaktive Anwendung zur Analyse von Bearbeiterzuordnungen, sind noch viele Punkte offen. Mit dieser Arbeit wurde jedoch ein Grundstein gelegt, um *Process Mining* auch um den Aspekt von Bearbeiterzuordnungen zu erweitern.

Literaturverzeichnis

[1] Process Mining: Discovering Workflow Models from Event-Based Data. In *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, Seiten 283–290, 2001.

[2] Process Mining: Discovering Direct Successors in Process Logs. In *Proceedings of the 5th International Conference on Discovery Science (Discovery Science 2002)*, volume 2534 of *Lecture Notes in Artificial Intelligence*, Seiten 364–373. Springer-Verlag, Berlin, 2002.

[3] Rediscovering Workflow Models from Event-Based Data. In *Proceedings of the Third International NAISO Symposium on Engineering of Intelligent Systems (EIS 2002)*, Seiten 65–73. NAISO Academic Press, Sliedrecht, The Netherlands, 2002.

[4] R. Agrawal, D. Gunopulos und F. Leymann. Mining Process Models from Workflow Logs, 1998.

[5] R. Agrawal, D. Gunopulos und F. Leymann. Mining Process Models from Workflow Logs (extended Version), 1998.

[6] R. Agrawal und R. Srikant. Fast Algorithms for Mining Association Rules. In J. Bocca, M. Jarke und C. Zaniolo (Hrsg.), *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, Seiten 487–499. Morgan Kaufmann, 12–15 1994.

[7] R. Agrawal und R. Srikant. Mining Sequential Patterns. In *ICDE*, Seiten 3–14, 1995.

[8] N. Berkman und T. Sandholm. What should be minimized in a decision tree: A re-examination, 1995.

[9] M. Berroth. Konzeption und Implementierung einer Komponente für Organisationsmodelle. Master's thesis, Universität Ulm, 2005. unpublished.

[10] F. Bertele. Emergent Workflow. Master's thesis, Universität Ulm, 2005.

[11] A. Bonifati, F. Casati, U. Dayal und M.-C. Shan. Warehousing Workflow Data: Challenges and Opportunities. In *VLDB*, Seiten 649–652, 2001.

[12] L. Breslow und D. Aha. Simplifying decision trees: a survey. *Knowledge Engineering Review*, 12(1):1–40, 1997.

[13] F. Casati, U. Dayal, M. Sayal und M. Shan. Business Process Intelligence, 2002.

[14] P. Clark und R. Boswell. Rule Induction with CN: Some recent improvements. In *Machine Learning - Proceedings of the Fifth European Conference (EWSL-91)*, Seiten 151–163. Springer-Verlag, Berlin, 1991.

[15] J. Cook. *Process Discovery and Validation through Event-Data Analysis*. PhD thesis, University of Colorado, 1996.

[16] J. Cook, Z. Dua, C. Liua und A. Wolf. Discovering Models of Behavior for Concurrent Workflows. *Computers in Industry*, (53), 2004.

[17] J. Cook, L. Votta und A. Wolf. Cost-Effective Analysis of In-Place Software Processes. *IEEE Trans. Software Eng.*, 24(8):650–663, 1998.

[18] J. Cook und A. Wolf. Discovering Models of Software Processes form Event-Based data. *ACM Transactions on Software Engineering and Methodology*, 7(3), 1998.

[19] J. Cook und A. Wolf. Event-Based Detection of Concurrency. Technical Report CU-CS-860-89, University of Colorado, 1998.

[20] J. Cook und A. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *Keine Ahnung*, 1999.

[21] F. Casati D. Grigori, M. Castellanos, U. Dayal, M. Sayal und M.-C. Shan. Business Process Intelligence. *Computers in Industry*, 2003.

[22] A. Datta, K. Lyytinen, L. Mathiasen und J. Roppenen. Automating the discovery of AS-IS Business Process Models: Probalistic and Algorithmic Approaches. *Information Systems Research*, 9(3), 1998.

[23] U. Dayal, M. Hsu und R. Ladin. Business Process Coordination: State of the Art, Trends, and Open Issues. In *The VLDB Journal*, Seiten 3–13, 2001.

[24] K. de Jong und W. Spears. Learning Concept Classification Rules using Genetic Algorithms. In *Proceedings of the Twelfth International Conference on Artificial Intelligence IJCAI-91*, volume 2, 1991.

[25] A. de Medeiros, W. van der Aalst und A. Weijters. Workflow Mining: Current Status and Future Directions, 2003.

[26] A. de Medeiros, B. van Dongen, W. van der Aalst und A. Weijters. Process Mining: Extending the a-algorithm to Mine Short Loops. Technical Report WP 113, Eindhoven University of Technology, 2004. BETA Working Paper Series.

[27] A. de Medeiros, B. van Dongen, W. van der Aalst und A. Weijters. Process Mining for Ubiquitous Mobile Systems: An Overview and a Concrete Algorithm. In L. Baresi, S. Dustdar, H. Gall und M. Materaseries (Hrsg.), *Ubiquitous Mobile Information and Collaboration Systems (UMICS 2004)*, volume 3272 of *Lecture Notes in Computer Science*, Seiten 154–168. Springer-Verlag, Berlin, 2004.

[28] A. de Medeiros, A. Weijters und W. van der Aalst. Using Genetic Algorithms to Mine Process Models: Representation, Operators and Results. Technical Report WP 124, Eindhoven University of Technology, 2004. BETA Working Paper Series.

[29] R. Diaz-Bone. Eine kurze Einführung in die sozialwissenschaftliche Netzwerkanalyse (SNA).

[30] S. Dustdar, T. Hoffmann und W. van der Aalst. Mining of ad-hoc business processes with TeamLog. Technical Report TUV-1841-2004-07, Vienna University of Technology, 2004.

[31] J. Eder, G. Olivotto und W. Gruber. A Data Warehouse for Workflow Logs. In *EDCIS*, Seiten 1–15, 2002.

[32] W. Gaaloul, S. Alaoui, K. Baina und C. Godart. Mining Workflow Patterns through Event-Data Analysis. 2005.

[33] W. Gaaloul, S. Alaoui, H. Bakkali, K. Baina und C. Godart. WorkflowMiner: An infrastructure for Mining Workflow Patterns. 2004.

[34] M. Golani und S. Pinter. Discovering workflow models from activities' lifespans. *Computers in Industry*, (53), 2004.

[35] M. Golani und S. Pinter. Generating a Process Model from a Process Audit Log, 2004.

[36] G. Greco, A. Guzzo, G. Manco, L. Pontieri und D. Saccà. Mining Constrained Graphs: The Case of Workflow Systems, 2004.

[37] G. Greco, A. Guzzo, G. Manco und D. Saccà. Mining and Reasoning on Workflows, 2003.

[38] G. Greco, A. Guzzo, G. Manco und D. Saccà. Mining Frequent Instances on Workflows. In *PAKDD*, Seiten 209–221, 2003.

[39] M. Hammori. InteractiveWorkflow Mining. Master's thesis, Universität Ulm, 2003.

[40] M. Hammori, J. Herbst und N. Kleiner. Interactive Workflow Mining. 2004.

[41] D. Hand, H. Mannila und P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

[42] J. Herbst. Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen, 2001.

[43] J. Herbst und D. Karagiannis. Workflow mining with InWoLvE. *Comput. Ind.*, 53(3):245–264, 2004.

[44] J. Herbst und N. Kleiner. Workflow Mining: A Case Study from Automotive Industry. 2003.

[45] D. Hollingsworth. Workflow Management Coalition The Workflow Reference Model. Technical report, The Workflow Management Coalition, 1995.

[46] S. Hong. R-MINI: An Iterative Approach for Generating Minimal Rules from Examples. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):709–717, / 1997.

[47] S.-Y. Hwang, C.-P. Wei und W.-S. Yang. Discovery of Temporal Patterns from Process Instances. *Computers in Industry*, (53), 2004.

[48] S.-Y. Hwang und W.-S. Yang. On the discovery of process models from their instances. *Decision Support Systems*, 2002.

[49] IBM. *IBM WebSphere MQ Workflow Programming Guide*, 9 edition, 2003.

[50] IBM. *IBM WebSphere MQ Workflow Administration Guide*, 8 edition, 2004.

[51] A. Inokuchi, T. Washio und H. Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, Seiten 13–23, London, UK, 2000. Springer-Verlag.

[52] N. Kleiner. Supporting Usage-Centered Workflow Design: Why and How? 2004.

[53] R. Kohavi und R. Quinlan. Decision Tree Learning.

[54] P. Koksal, S. Arpinar und A. Dogac. Workflow History Management. *SIGMOD Record*, 27(1):67–75, 1998.

[55] E. Kretschmann, W. Fleischmann und R. Apweiler. Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT. *Bioinformatics*, 17(10 2001):920–926, 2001.

[56] M. Kubicek. Organisatorische Aspekte in flexiblen Workflow-Management-Systemen. Master's thesis, Universität Ulm, 1998.

[57] M. Kuramochi und G. Karypis. Frequent Subgraph Discovery. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, Seiten 313–320, Washington, DC, USA, 2001. IEEE Computer Society.

[58] F. Leymann und D. Roller. *Production Workflow.* Prentice Hall, 1999.

[59] H. Mannila, H. Toivonen und A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[60] R. Marmelstein und G. Lamont. A Method for Mining Simplified Decision Rule Sets. In *Proceedings of the International ICSC Congress on Computational Intelligence: Methods and Application, Rochester, 1999*, 1999.

[61] L. Maruster. *A machine learning approach to understand business processes.* PhD thesis, Technische Universiteit Eindhoven, 2003.

[62] L. Maruster, J. Wortmann, A. Weijters und W. van der Aalst. Discovering Distributed Processes in Supply Chains. In *Proceedings of the International Conference on Advanced Production Management Systems (APMS 2002)*, Seiten 119–128, 2002.

[63] L. Miclet und P. Dupont. Inférence grammaticale régulière: fondements théoriques et principaux algorithmes. Technical Report 3449, Institut national de recherche en informatique et en automatique, 1998.

[64] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[65] C. Moore. Common Mistakes in Workflow Implementations, 2002.

[66] P. Norvig und S. Russell. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2002.

[67] F. Olken und D. Rotem. Workflow Execution History Data Management: A Framework. In *ICWS*, Seiten 55–61, 2003.

[68] J. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[69] R. Quinlan. Learning Decision Tree Classifiers, 1996.

[70] M. Reichert. *Dynamische Ablaufänderungen in Workflow-Management-Systemen.* PhD thesis, Universität Ulm, 2000.

[71] M. Reichert und P. Dadam. $\text{ADEPT}_{\text{flex}}$-Supporting Dynamic Changes of Workflows Without Losing Control. *J. Intell. Inf. Syst.*, 10(2):93–129, 1998.

[72] M. Reichert, P. Dadam und K. Kuhn. Clinical Workflows - The Killer Application for Process-oriented Information Systems? In *Proceedings of the 4th Int'l Conference on Business Information Systems*, Seiten 36–59. Springer Verlag, 2000.

[73] M. Reichert, S. Rinderle und P. Dadam. ADEPT Workflow Management System: Flexible Support for Enterprise-Wide Business Processes. In *Business Process Management*, Seiten 370–379, 2003.

[74] G. Schimm. Process Mining linearer Prozessmodelle - Ein Ansatz zur automatisierten Akquisition von Prozesswissen. 2001.

[75] G. Schimm. Mining most specific Workflow Models from event-based Data, 2003.

[76] G. Schimm. Process Mining - Ein kurzer Überblick über ein Data-Mining-Forschungsgebiet. 2003.

[77] G. Schimm. Mining Exact Models of Concurrent Workflows, 2004.

[78] I. Shioya und T. Miura. Inductive Classification Using Taxonomy. In *Knowledge Representation Meets Databases*, Seiten 87–98, 2000.

[79] R. Silva, J. Zhang und J. Shanahan. Probabilistic Workflow Mining, 2005.

[80] R. Srikant und R. Agrawal. Mining Generalized Association Rules. In *Proceedings of the 21st VLDB Conference, Zürich, Swizerland, 1995*, 1995.

[81] Staffware plc. *Staffware 2000/GWD User Manual*, 2000.

[82] W. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In *CAiSE Workshops (2)*, Seiten 138–145, 2004.

[83] W. van der Aalst, A. De Medeiros und A. Weijters. Genetic Process Mining. 2004.

[84] W. van der Aalst und M. Song. Discovering Social Networks from Event Logs. *BETA Working Paper Series, Eindhoven University of Technology, Eindhoven*, 2004.

[85] W. van der Aalst und M. Song. Mining Social Networks: Uncovering Interaction Patterns in Business Processes, 2004.

[86] W. van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm und A. Weijters. Workflow mining: A survey of issues and approaches, 2003.

[87] W. van der Aalst und A. Weijters. Process Mining: A Research Agenda, 2004.

[88] W. van der Aalst, A. Weijters und L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 2004.

[89] B. van Dongen und W. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. *Keine Ahnung*, 2004.

[90] M. Weske W. van der Aalst, A. Hofstede. Business process management: A survey, 2003.

[91] C.-P. Wei, S.-Y. Hwang und W.-S. Yang. Mining Frequent Tenporal Patterns in Process Databases. 2000.

[92] L. Wen, J. Wang, W. van der Aalst, Z. Wang und J. Sun. A Novel Approach for Process Mining Based on Event Types. Technical Report WP 118, Eindhoven University of Technology, 2004. BETA Working Paper Series.

[93] U. Wiedemuth-Catrinescu. Evolution von Organisationsmodellen in Workflow-Management-Systemen. Master's thesis, Universität Ulm, 2002.

[94] A. Wolf und D. Rosenblum. A Study in Software Process Data Capture and Analysis. 1993.

[95] X.Yan und J. Han. gSpan: Graph-Based Substructure Pattern Mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, Seite 721, Washington, DC, USA, 2002. IEEE Computer Society.

[96] M. Zapf. Pattern-driven Process Design, 2003.

[97] M. zur Mühlen. Process-driven Management Information Systems - Combining Data Warehouses and Workflow Technology. In *International Conference on Electronic Commerce Research (ICECR-4)*, Seiten 550–566, 2001.

[98] M. zur Mühlen. Organizational Management in Workflow Applications. *Information Technology and Management Journal*, Seiten 271–291, 2004.

[99] M. zur Mühlen und M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000.

Abbildungsverzeichnis

Tabellenverzeichnis

Erklärung

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

Ulm, den 02.05.2005