UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

LEHRSTUHL VIII
KÜNSTLICHE INTELLIGENZ

Datengesteuertes Lernen von syntaktischen Einschränkungen des Hypothesenraumes für modellbasiertes Lernen

LS–8 Report 15

Marcus Lübbe

Dortmund, 17. März 1995

Universität Dortmund
Fachbereich Informatik

University of Dortmund
Computer Science Department

Forschungsberichte des Lehrstuhls VIII (KI)
Fachbereich Informatik
der Universität Dortmund

ISSN 0943-4135

Anforderungen an:

Universität Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

Research Reports of the unit no. VIII (AI)
Computer Science Department
of the University of Dortmund

ISSN 0943-4135

Requests to:

University of Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

e-mail: reports@ls8.informatik.uni-dortmund.de
ftp: ftp-ai.informatik.uni-dortmund.de:pub/Reports
www: http://www-ai.informatik.uni-dortmund.de/ls8-reports.html

Datengesteuertes Lernen von syntaktischen Einschränkungen des Hypothesenraumes für modellbasiertes Lernen

LS–8 Report 15

Marcus Lübbe

Dortmund, 17. März 1995

Universität Dortmund
Fachbereich Informatik

Zusammenfassung

Lernverfahren für prädikatenlogische Formalismen eignen sich als Werkzeuge, die den Aufbau und die Wartung komplexer Sachbereichstheorien unterstützen, da sie sowohl Hintergrundwissen in den Lernvorgang einbeziehen als auch relationale Beziehungen zwischen den Objekten der Theorie behandeln können. Die im Vergleich zu klassischen, auf Aussagenlogik basierenden Verfahren erweiterte Ausdrucksstärke führt aber auch zu einer größeren Komplexität der Lernaufgabe. Das induktive Lernverfahren RDT der Werkbank MOBAL verwendet Modellwissen in Form von Regelmodellen um den Suchraum einzuschränken. Diese syntaktischen Vorgaben an das Lernziel ermöglichen zwar eine genaue Steuerung der Lernaufgabe durch den Benutzer, fehlen aber die zum Lernziel korrespondierenden Formelschemata, kann das Lernziel nicht erreicht werden.

Die vorliegende Arbeit präsentiert daher einen heuristischen Ansatz zum automatischen Erwerb von Regelmodellen, der auf der Berechnung speziellster Generalisierungen beruht. Um Hintergrundwissen zu berücksichtigen, werden die für das Lernziel relevanten Teile dieses Wissens mit den Beispielen verknüpft. Die Berechnung speziellster Generalisierungen von Regelmodellen dient zur schrittweisen Verallgemeinerung der Regelmodelle. Eine neue Erweiterung der θ–Subsumtion auf Regelmodelle und ein Redundanzbegriff für solche Formelschemata sind weitere Bestandteile dieser Arbeit.

Inhaltsverzeichnis

Kapitel 1

Hintergrund, Zielsetzung und Aufbau der Arbeit

Der Bereich der *induktiven logischen Programmierung (ILP)* ist ein gemeinsames Forschungsgebiet des maschinellen Lernens und des logischen Programmierens [Muggleton, 1992]. Er beschäftigt sich mit der Charakterisierung von Begriffen beziehungsweise dem Entdecken von Regelhaftigkeiten durch Induktion von prädikatenlogischen Formeln aus Beispielen unter Einbeziehung von Hintergrundwissen. Dieser Ansatz versucht die Ausdrucksschwäche der vielen erfolgreichen und effizienten Lernalgorithmen im Bereich der Aussagenlogik (hierzu gehören aufgrund der gleichen Ausdrucksstärke auch Sprachen mit einer Attribut–Werte–Repräsentation) zu überwinden.

Im Gegensatz zu Verfahren wie Quinlans ID3 [Quinlan, 1983] zur Induktion von Entscheidungsbäumen sind Lernverfahren aus dem Gebiet der induktiven logischen Programmierung in der Lage,

- logische Sprachen erster Stufe oder Varianten davon als Repräsentationsformalismen zu benutzen
- und Hintergrundwissen in den Lernvorgang einzubeziehen.

Die erste Eigenschaft befähigt ILP–Verfahren zum Lernen in Sachbereichen, die Relationen zwischen den dargestellten Objekten behandeln und somit einen prädikatenlogischen Repräsentationsformalismus erfordern und in denen klassische, auf aussagenlogischen Formalismen basierende Lernverfahren daher nicht effektiv sein können. Die Einbeziehung von Hintergrundwissen ist generell unerläßlich zur Darstellung und Beschreibung komplexer Sachverhalte.

Durch diese Fähigkeiten sind Lernverfahren für prädikatenlogische Formalismen zwingend erforderlich und besonders geeignet, den Aufbau und die Wartung komplexer Sachbereichstheorien teilweise zu automatisieren. In Systemen wie der Werkbank MOBAL [Morik *et al.*, 1993] kooperieren Lernverfahren für logische Repräsentationen mit dem Benutzer bei der Erstellung eines operationalen Modelles eines Weltausschnittes.

Angewendet wurden ILP–Verfahren unter anderem

- zur Überwachung von Sicherheitsstrategien in offenen, verteilten Telekommunikationsnetzen[1] (SPEED) [Sommer *et al.*, 1994] mit dem System MOBAL,
- zur quantitativen Modellierung der Beziehung zwischen chemischer Struktur und Wirkung von Medikamenten (engl. *Quantitative Structure–Activity Relationship*) [Bratko und King, 1993] mit dem Lernverfahren GOLEM [Muggleton und Feng, 1992] und
- zum Design von Maschen für numerische Berechnungen mit Methoden der finiten Elemente (engl. *Finite–Element Mesh Design*) [Dolsak und Muggleton, 1992].

1.1 Problemstellung und Zielsetzung der Arbeit

Die Aufgabenstellung von Verfahren aus dem Bereich der induktiven logischen Programmierung (ILP–Problem) zum Lernen einer Begriffsbeschreibung läßt sich mittels der Terminologie der Prädikatenlogik beziehungsweise der logischen Programmierung beschreiben[2]: Aus einer Menge von Beispielen und Gegenbeispielen eines Begriffes und einer logischen Theorie, dem Hintergrundwissen, soll eine Menge logischer Ausdrücke, die *Hypothese*, induktiv gefolgert werden. Diese Hypothese soll konsistent mit der Theorie sein und zusammen mit dieser Theorie alle Beispiele erklären (logisch folgern), jedoch keines der Gegenbeispiele.

Die erweiterte Ausdruckstärke entsprechender induktiver Verfahren führt zu einer wachsenden Komplexität der Lernaufgabe. Ein ILP–Lernverfahren kann als Suche im Raum aller in der Hypothesensprache konstruierbaren Ausdrücke nach einer Hypothese verwirklicht werden, die die positiven Beispiele erklärt, aber kein Negatives [Mitchell, 1982]. Die Ausdrücke dieses Suchraumes werden durch den logischen Folgerungsbegriff geordnet. Die Unentscheidbarkeit der logischen Folgerung in der Prädikatenlogik ist verantwortlich dafür, daß das ILP–Problem im allgemeinen unentscheidbar ist, wenn Beispiele, Hintergrundwissen und Hypothesen in einer beliebigen logischen Sprache repräsentiert werden [Plotkin, 1971a]. Daher verwendet man beim induktiven logischen Programmieren meist eine nicht vollständige, aber korrekte und entscheidbare Ableitungsregel zur Operationalisierung der logischen Folgerung. Derartige Operatoren definieren ebenso eine Ordnung über dem Suchraum und schränken die Menge der lernbaren Hypothesen zusätzlich ein, da die gesuchte Hypothese bezüglich der durch die Ableitungsregel definierten Ordnung in Relation zu den Beispielen stehen muß.

[1] siehe Abschnitt 5.3

[2] Das Entdecken von Regelhaftigkeiten besitzt eine andere Semantik, die in Abschnitt 2.1 als nicht monotone ILP Semantik eingeführt wird.

Aber auch unter Verwendung von entscheidbaren Ableitungsregeln, wie z.B. der θ–Subsumtion [Robinson, 1965; Plotkin, 1970], ist das ILP–Problem im allgemeinen nicht effizient berechenbar (*NP*–schwierig oder noch höher in der Komplexitätshierarchie).

Es gibt zwei Ansatzpunkte, um die Komplexität dieses Suchproblems zu reduzieren. Die Größe des Suchraums wird durch die gewählte Repräsentationssprache für die Beispiele, das Hintergrundwissen und die Hypothesen bestimmt. Daher gilt es, diese Sprachen auf solche Teilmengen der Prädikatenlogik erster Stufe zu reduzieren, die einen möglichst kleinen Suchraum aufspannen (engl. *language bias*). Beim induktiven logischen Programmieren wurden vor allem verschiedene Hypothesensprachen hinsichtlich ihrer Lernbarkeit untersucht. So wurde z.B. nachgewiesen, daß die Klasse der tiefenbeschränkten determinierenden Klauseln effizient lernbar ist (PAC-lernbar) [Muggleton und Feng, 1992; Dzeroski *et al.*, 1992], während dagegen determinierende Klauseln mit einer Tiefenbeschränkung logarithmisch zur Größe der Beispiele [Cohen, 1993] oder gar ohne Tiefenbeschränkung [Kietz, 1993b] nicht effizient lernbar sind. Nicht determierende Klauseln sind sogar bei einer Tiefe von maximal zwei nicht in polynomieller Zeit lernbar [Kietz, 1993b]. Natürlich muß die gesuchte Hypothese in der gewählten Sprache noch formulierbar sein. Die polynomiell lernbaren Sprachen sind aber für viele Problemstellungen nicht ausdrucksstark genug. Der zweite Ansatzpunkt liegt in einer effizienten Suchstrategie, also der Art und Weise, wie der durch diese Sprachen festgelegte Raum durchsucht wird (engl. *search bias*).

Das System MOBAL [Morik *et al.*, 1993] ist eine modulare Werkbank mit einem eingeschränkten logischen Repräsentationsformalismus höherer Ordnung, die den Aufbau und die Wartung einer Wissensbasis unterstützt. MOBAL verfolgt beim Modellierungsprozeß das Paradigma der ausbalancierten Kooperation (engl. *balanced cooperation*) [Morik, 1993] zwischen dem Benutzer und den Lernwerkzeugen des Systems. Das Werkzeug RDT (engl. *rule discovery tool*) [Kietz und Wrobel, 1991] dieser Werkbank gehört zu den induktiven Lernverfahren, die die Komplexität der Lernaufgabe in prädikatenlogischen Formalismen reduzieren, indem sie die Hypothesensprache durch die Vorgabe syntaktischer Schablonen einschränken. Diese Schablonen präsentieren ein syntaktisches Modell[3] der zu lernenden Ausdrücke, der Hypothesenraum kann unabhängig vom eigentlichen Lernverfahren eingeschränkt werden. Solche *modellbasierten* Lernverfahren können bei geeigneter Wahl des Modellwissens effizient einen Begriff charakterisieren bzw. Regelhaftigkeiten in einer Theorie entdecken.

Als Modellwissen im Werkzeug RDT werden vor allem *Regelmodelle* (auch Regelschemata)[4] eingesetzt [Kietz und Wrobel, 1991]. Regelmodelle erwei-

[3] Der Begriff des Modells wird im folgenden nicht im Sinne des logischen Modellbegiffes verwendet. Der Begriff Modell zur Beschreibung von Wissen über die Form der gesuchten Hypothesen stammt aus [Morik *et al.*, 1993, Kapitel 6]. Soweit nicht durch den Kontext eindeutig, wird die Benutzung des Modellbegiffes im Sinne der Logik in dieser Arbeit stets gekennzeichnet.

[4] In dieser Arbeit werden die Begriffe Regelmodell und Regelschema synonym verwendet.

tern die Syntax von Klauseln um die Verwendung von Prädikatsvariablen und erlauben somit die Behandlung von Prädikaten als Datenobjekte. Modellwissen für ein Lernverfahren kann auch in anderer Form, etwa durch eine Darstellung von semantischen Beziehungen zwischen den Prädikaten des Sachbereiches, vorliegen.

Das Werkzeug RDT versucht, vom Benutzer vorgegebene Regelmodelle sukzessiv zu instanziieren und zu testen, ob die resultierenden, möglicherweise nur zum Teil instanziierten, Regelmodelle gemäß eines vom Benutzer wählbaren Kriteriums akzeptabel sind[5]. Die Regelmodelle schränken somit den Suchraum ein, da sie nur eine bestimmte Teilmenge von Regeln als Instanzen und somit Hypothesen zulassen.

Existieren für ein Lernproblem gut angepaßte Regelmodelle, ist das Verfahren RDT oft schneller als andere Lernverfahren (z.B. FOIL [Quinlan, 1990]). Bei genauer Kenntnis der Lernaufgabe kann der Lernprozeß durch die Vorgabe der Regelmodelle sehr genau gesteuert werden. Liegt die gesuchte Regel aber nicht im von den Regelmodellen aufgespannten Hypothesenraum, fehlt also das korrespondierende Regelmodell, läßt sich diese Regel mit dem Werkzeug RDT nicht lernen. Voraussetzung für ein gutes Lernergebnis ist daher die Vorgabe der richtigen Regelmodelle.

Die Festlegung geeigneter Regelmodelle ist im System MOBAL momentan Aufgabe des Benutzers. Das Werkzeug MAT (engl. *model acquisition tool*) dient der Akquisition von Regelmodellen, indem es Prädikatssymbole aus eingegebenen Klauseln in Prädikatsvariablen umwandelt. Für ein korrektes Lernergebnis ist aber auch hier die Vorgabe einer Klausel notwendig, die der gesuchten Hypothese syntaktisch so entspricht, daß beide Instanzen derselben Regelmodelle sind.

Abbildung 1.1 veranschaulicht, wie die Werkzeuge RDT, MAT und die Inferenzmaschine in der Werkbank MOBAL kooperieren[6]. Die von RDT gefundenen Regeln stehen durch die Inferenzmaschine weiteren Lernläufen oder anderen Werkzeugen von MOBAL zur Inspektion und Revision der Wissensbasis zur Verfügung (engl. *closed loop learning*). Dieses Zusammenspiel wird jedoch bei der Gewinnung von Modellwissen unterbrochen, da das Werkzeug MAT nur die unzureichende Abstraktion von Regelmodellen aus der Werkbank bekannten Regeln erster Stufe leistet. Um die skizzierte Lücke im Zusammenspiel der Werkzeuge zu schließen und dadurch den Wissensakquisitionsprozeß weiter zu automatisieren, wird daher ein Verfahren zum Erwerb von Regelmodellen für eine zu lernende Menge von Hypothesen benötigt.

Die Eingabe eines solchen Lernverfahrens setzt sich aus dem Hintergrundwissen und Beispielen einer Relation zusammen. Auszugeben sind die zur ge-

[5] RDT ist ein Verfahren zum Entdecken von Regelhaftigkeiten, es werden keine Begriffsbeschreibungen gelernt. Bei geeigneter Festlegung der Parameter der Werkbank RDT kann dieses Verfahren eingeschränkt auch zum Lernen einer Begriffsbeschreibung benutzt werden (für eine formale Diskussion siehe Abschnitte 2.1, 3.3 und 4.1).

[6] Die Abbildung zeigt nur einen Teil der in der Werkbank MOBAL vorhandenen Werkzeuge. Auch die übrigen Werkzeuge des Systems kooperieren, indem sie zum Teil das Lernverfahren RDT aufrufen oder z.B. Modellwissen in Form einer Prädikatstopologie bereitstellen (siehe Abschnitt 3.1.2).

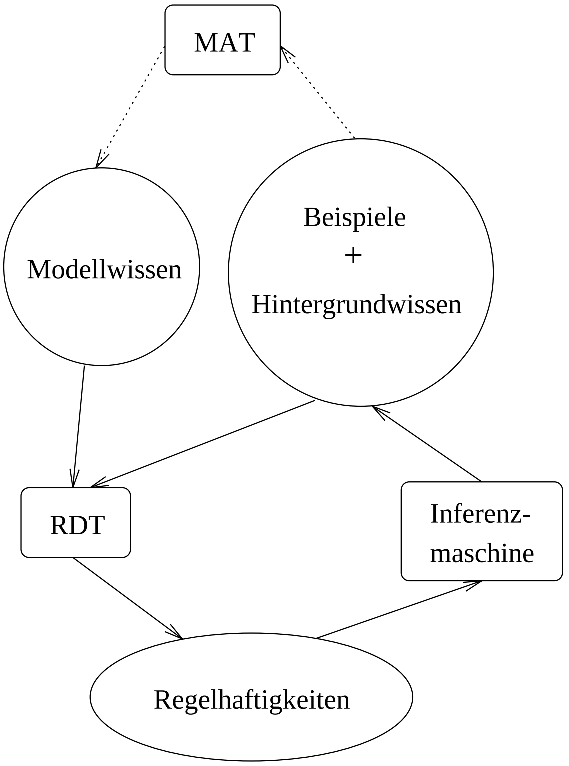

Abbildung 1.1: Kooperation der Werkzeuge RDT, MAT und der Inferenzmaschine in MOBAL

suchten Hypothese korrespondierenden Regelmodelle oder eine Obermenge davon. Die Existenz eines vollständigen und zugleich effizienten Verfahrens zum Lernen einer in der Größe begrenzten Menge von Regelmodellen in Kombination mit einem polynomiell zeitbeschränkten modellbasierten Lernverfahren würde generell ein effizientes Lernen von Regelhaftigkeiten und das Lernen einer Begriffsbeschreibung in einer bestimmten Hypothesensprache erlauben[7]. Insbesondere der letzte Punkt steht im Gegensatz zu den bereits angeführten negativen Ergebnissen der Lerntheorie[8]. Die Zielsetzung dieser Arbeit besteht daher in der Entwicklung eines Lernverfahrens, das innerhalb von möglichst polynomiellen Zeit- und Platzbeschränkungen eine Approximation einer für die Lernaufgabe vollständigen Menge von Regelmodellen berechnet.

Ein entsprechendes datengesteuertes Lernverfahren zum Erwerb von Modellwissen wird in dieser Arbeit vorgestellt. Dabei werden Techniken des induktiven logischen Programmierens zum Lernen von prädikatenlogischen Ausdrücken eingesetzt und zum Teil zur Induktion von Regelmodellen er-

[7] Diese Aussage wird in Abschnitt 4.1 bewiesen.

[8] Resultate zur Lernbarkeit von verschiedenen ILP–Problemen unter Einschränkung der Repräsentationsformalismen werden in Kapitel 2.2.3 ausführlich vorgestellt.

Beispiele	Hintergrundwissen	
großmutter(ann,sue) $\oplus$	weiblich(ann)	tochter(sue,tom)
großmutter(ann,bob) $\oplus$	mutter(ann,tom)	vater(tom,sue)
großmutter(bob,sue) $\ominus$	mutter(ann,jim)	mutter(eve,sue)
großmutter(tom,bob) $\ominus$	elternteil(ann,tom)	vater(zak,tom)
	elternteil(ann,jim)	elternteil(zak,jim)
	vater(tom,bob)	mutter(liz,pat)
	sohn(X,Y) $\leftarrow$ vater(Y,X)	
	bruder(X,Y) $\leftarrow$ tochter(Y,Z) & vater(Z,X)	

Tabelle 1.1: Ein einfaches ILP-Problem

weitert. Der in dieser Arbeit verfolgte Ansatz basiert auf der Konstruktion speziellster Generalisierungen der Beispiele der zu beschreibenden Relation [Plotkin, 1970; Haussler, 1989]. Das resultierende System erweitert bzw. ersetzt das bisherige Werkzeug MAT der Werkbank MOBAL zum Erwerb von Regelmodellen. Beispiele, Hintergrundwissen, Modellwissen und gelernte Regeln werden in dem eingeschränkten, höherwertigen logischen Formalismus von MOBAL repräsentiert.

1.2 Ein einführendes Beispiel

Tabelle 1.1 stellt ein einfaches ILP-Problem dar. Gesucht ist eine Definition für den Begriff großmutter. Mögliche Hypothesen, die diesen Begriff charakterisieren, sind z.B. die Klauseln:

großmutter(X,Y) $\leftarrow$ elternteil(U,Y) & mutter(X,U),
großmutter(X,Y) $\leftarrow$ mutter(U,Y) & mutter(X,U),
großmutter(X,Y) $\leftarrow$ vater(U,Y) & mutter(X,U),
großmutter(X,Y) $\leftarrow$ vater(Z,Y) & mutter(X,Z) & vater(Z,V),
großmutter(X,Y) $\leftarrow$ elternteil(X,Z) & tochter(Y,Z) & weiblich(X).

Die erste Regel entspricht der intuitiven Beschreibung des Begriffes großmutter. Aber auch die anderen Regeln stellen mögliche Definitionen dar. Ein ILP–Lernverfahren muß daher festlegen, welche der denkbaren Begriffsbeschreibungen bevorzugt werden. Durch Vorgabe von Modellwissen in Form des Regelmodells

großmutter(X,Y) $\leftarrow$ P1(U,Y) & mutter(X,U)

wird erreicht, daß ein Lernverfahren nur solche Regeln lernt, die Instanzen dieses Modells sind. P1 ist eine Prädikatsvariable, die durch entsprechende Prädikate gleicher Stelligkeit ersetzt werden kann. Regeln mit mehr als zwei Literalen im Körper werden durch dieses Modell ausgeschlossen. Außerdem wird die Verknüpfung der Literale über gemeinsame Termvariablen vorgegeben. Der Suchraum dieses Problems ist jetzt stark reduziert, nur die ersten drei Regeln im obigen Beispiel sind Instanzen dieses Regelmodells.

Oft sind solche Kenntnisse über die zu lernende Begriffsdefinition jedoch nicht vorhanden. Im vorliegenden Beispiel stellt sich dann die Aufgabe, aus den Beispielen des Begriffes **großmutter** und dem Hintergrundwissen für die gesuchte Begriffscharakterisierung adäquates Modellwissen zu akquirieren.

1.3 Aufbau der Arbeit

Zu Beginn dieser Arbeit werden in Kapitel 2 die theoretischen Grundlagen und wichtigsten Techniken zum Lernen aus Beispielen in prädikatenlogischen Formalismen beschrieben. In Abschnitt 2.1 werden die für diese Arbeit relevanten Szenarien zum Lernen aus Beispielen in logischen Sprachen formalisiert. Zwei weitere Szenarien, die Identifikation im Grenzwert und das wahrscheinlich annähernd korrekte (PAC–)Lernen, werden in diesen Rahmen eingeordnet. Außerdem werden Zusammenhänge wiedergegeben, die es erlauben, Aussagen über die Lernbarkeit von Problemstellungen in diesen verschiedenen Szenarien in Beziehung zu setzen. Das Lernen aus Beispielen wird dann in Abschnitt 2.2 als Suchproblem dargestellt. Für die Komplexität dieses Suchproblem sind folgende Bestimmungsgrößen maßgeblich:

- die verwendete Ableitungsregel (auch allgemeiner–als–Relation oder Generalisierungsmodell) (Abschnitt 2.2.1),
- die Repräsentation des Hintergrundwissens (Abschnitt 2.2.2) und
- die Beschränkung der Hypothesensprache auf Teilmengen der Prädikatenlogik erster Stufe (engl. *language bias*) (Abschnitt 2.2.3).

Die diskutierten Generalisierungsmodelle geben keine Anleitungen zur Konstruktion der gesuchten Generalisierungen der Beispiele. Inverse Resolutionsoperatoren [Muggleton, 1992; Jung, 1993] und speziellste Generalisierungen (engl. *least general generalisations, most specific generalisations*) [Plotkin, 1970; Haussler, 1989] operationalisieren diese Generalisierungsmodelle. Letztere werden in dieser Arbeit zum datengesteuerten Lernen eingesetzt und daher in Abschnitt 2.3 ausführlich vorgestellt.

Kapitel 3 beschreibt detailliert, wie in der Werkbank MOBAL unter Einsatz des modellbasierten Lernverfahrens RDT induktiv Regeln gelernt werden. Zu Beginn des Kapitels werden in Abschnitt 3.1 kurz die Idee und insbesondere der Repräsentationsformalismus und der grundlegende Aufbau der Werkbank MOBAL vorgestellt. Regelmodelle als wichtigster Bestandteil des Modellwissens werden in Abschnitt 3.2 eingeführt und anschließend skizziert, wie das Werkzeug RDT unter Verwendung dieses Modellwissens effektiv Regelhaftigkeiten in der Sachbereichstheorie entdeckt (Abschnitt 3.3). Besonderes Augenmerk in diesem Kapitel gilt den Regelmodellen. Zur Strukturierung des Suchvorganges sind diese bezüglich einer Generalisierungsrelation anzuordnen. In Abschnitt 3.2.1 werden solche Ordnungsrelationen für

Regelmodelle diskutiert und erläutert, welche dieser Relationen für ein modellgestütztes Lernverfahren geeignet ist[9]. Ein entsprechender Redundanzbegriff für Regelmodelle wird in Abschnitt 3.2.2 definiert. Zum Abschluß dieses Kapitels werden in Abschnitt 3.4 verwandte modellbasierte Ansätze vorgestellt.

Der Schwerpunkt dieser Arbeit liegt auf Kapitel 4. Dort wird ein datengesteuerter Ansatz zum Erwerb von Modellwissen mit Methoden des induktiven Lernens vorgestellt. Zunächst wird die Komplexität dieser Aufgabe formal untersucht. Dann werden die wichtigsten Schritte und Verfahren dieses Ansatzes beschrieben und mit den bekannten Methoden und Einschränkungen aus Kapitel 2 in Bezug gesetzt. Abschnitt 4.3 stellt das resultierende Werkzeug des Systems MOBAL zum Erwerb von Regelmodellen vor. In Übereinstimmung mit den in der Problemstellung bereits erwähnten und in Kapitel 2 ausführlich zu präsentierenden negativen Ergebnissen zur Lernbarkeit in prädikatenlogischen Formalismen basiert auch dieser Ansatz auf zum Teil nicht in polynomieller Zeit berechenbaren Teilproblemen. Daher werden für diese Probleme möglichst effiziente, zum Teil heuristische oder approximative Algorithmen eingesetzt (Abschnitte 4.3.1 bis 4.3.5). Anschließend werden in Abschnitt 4.3.6 die vom Benutzer einzustellenden Parameter des Systems erklärt. Zum Abschluß werden in Abschnitt 4.4 andere Ansätze zum Erwerb von syntaktischen Vorgaben an ein Lernverfahren mit diesem neuen Ansatz verglichen.

Kapitel 5 gibt Experimente und dabei festgehaltene Resultate mit diesem Verfahren wieder.

Eine zusammenfassende Bewertung und ein Ausblick beenden diese Arbeit (Kapitel 6).

[9] Die dort gewählte Relation ist eine Modifikation der in [Kietz und Wrobel, 1991] und [Morik *et al.*, 1993, Kapitel 6] definierten Erweiterung der θ–Subsumtion auf Regelmodelle und korrigiert einige der, auf dieser Definition beruhenden, kontraintuitiven Effekte und Inkonsistenzen.

Kapitel 2

Lernen aus Beispielen in prädikatenlogischen Formalismen

Wie bei allen Problemen in der Informatik ist es auch bei der Entwicklung von Lernverfahren für prädikatenlogische Formalismen notwendig, sich vorab zu überlegen, ob das vorliegende Problem überhaupt entscheidbar ist, d.h. auf einem Computer berechenbar, und welche Komplexität es relativ zu schwierigen Problemen ohne effiziente Lösungsverfahren hat.

Eine Diskussion dieser Fragestellung bedingt eine geeignete Formalisierung dieses Problems. Die relevanten Aufgabenstellungen für Lernverfahren aus dem Bereich der induktiven logischen Programmierung werden in Abschnitt 2.1 definiert. Ergänzend werden weitere wichtige theoretische Szenarien zur Lernbarkeit in den vorliegenden Rahmen eingeordnet.

Abschnitt 2.2 beschreibt Lernverfahren als Suchalgorithmen und stellt Generalisierungsmodelle (Abschnitt 2.2.1), die Repräsentation des Hintergrundwissens (Abschnitt 2.2.2) und die Beschränkung der Repräsentationssprachen auf Teilmengen der Prädikatenlogik (Abschnitt 2.2.3) als Einflußfaktoren auf Entscheidbarkeit und Komplexität eines Lernproblems vor.

Die vorgestellten Generalisierungsmodelle definieren unabhängig von konkreten Generalisierungsoperatoren eine allgemeiner–als–Relation zwischen den Formeln des Hypothesensuchraumes. In Abschnitt 2.3 werden speziellste Generalisierungen als Operatoren beschrieben, die in Umkehrung deduktiver Verfahren Formeln einer Sprache generalisieren. Andere Generalisierungsoperatoren, wie inverse Resolutionsverfahren [Muggleton und Buntine, 1992], werden in dieser Arbeit nicht eingesetzt[1].

[1]Eine ausführliche Diskussion der Vollständigkeit von Generalisierungsoperatoren bezüglich der verschiedenen Generalisierungsmodelle findet sich in [Jung, 1993].

2.1 Szenarien für logikbasierte Lernverfahren

Das Forschungsgebiet der induktiven logischen Programmierung hat hauptsächlich zwei verschiedene Szenarien für induktive Lernverfahren hervorgebracht, eingeführt von Plotkin [Plotkin, 1971b] und Helft [Helft, 1989]. Die von Plotkin beschriebene Zielsetzung liegt in der Induktion einer *Begriffsbeschreibung* (engl. *concept learning*), die konsistent mit dem Hintergrundwissen ist und die Beispielmenge erklärt. Abweichend von dieser Aufgabenstellung definiert Helft Induktion in prädikatenlogischen Formalismen als *nicht monotone Schlußfolgerung*[2], die für eine gegebene Formelmenge eine minimale vollständige Theorie in einer bestimmten Sprache induziert.

Die Induktion einer Begriffsbeschreibung aus einer Menge von Beispielen für eine gesuchte Relation[3] wird als die klassische Semantik für ILP–Probleme angesehen.

Definition 2.1 (ILP Semantik) *Gegeben seien Hintergrundwissen B in einer Sprache $\mathcal{LB}$, positive und negative Beispiele $E = E^+ \cup E^-$ eines Begriffes in einer Sprache $\mathcal{LE}$, mit der Eigenschaft, daß sich kein Beispiel aus dem Hintergrundwissen logisch folgern läßt ($\forall e \in E : B \not\models e$) und die Beispiele konsistent mit dem Hintergrundwissen sind ($B \wedge E \not\models \Box$), und eine Hypothesensprache $\mathcal{LH}$. Alle Sprachen seien Teilmengen der Prädikatenlogik erster Stufe.*

Die Aufgabenstellung der **induktiven logischen Programmierung (ILP-Problem)** *besteht im Finden einer Hypothese $H \subset \mathcal{LH}$, für die gilt:*

1. *($B \wedge H \wedge E \not\models \Box$), Die Hypothese H ist konsistent mit dem Hintergrundwissen B und den Beispielen E,*
2. *($B \wedge H \models E^+$), die positiven Beispiele E^+ folgen aus dem Hintergrundwissen B und der Hypothese H,*
3. *($\forall e^- \in E^- : B \wedge H \not\models e^-$), die negativen Beispiele E^- folgen nicht aus dem Hintergrundwissen B und der Hypothese H.*

Eine konkrete Instanz dieser Problemklasse wird durch das Tripel ($\mathcal{LB}, \mathcal{LE}, \mathcal{LH}$) bestimmt. Die Entscheidungsvariante dieses Problems, also die Frage, ob solch eine Hypothese $H \subset \mathcal{LH}$ existiert, heißt **Konsistenzproblem**.

Fast alle ILP–Lernverfahren beschränken ihren Repräsentationsformalismus auf definite Klauseln. Dies erlaubt die Definition von Lernproblemen über das minimale Herbrandmodell $\mathcal{M}(T)$ einer definiten Theorie T:

[2] Natürlich sind alle induktiven Lernverfahren nicht monoton, aber Helft stellt diesen Aspekt der Induktion besonders heraus.

[3] Die zu beschreibende Relation wird im folgendem auch Zielbegriff oder –relation genannt.

Definition 2.2 (definite ILP Semantik) *Gegeben seien Hintergrundwissen B in einer Sprache $\mathcal{LB}$, positive und negative Beispiele $E = E^+ \cup E^-$ eines Begriffes in einer Sprache $\mathcal{LE}$, mit der Eigenschaft, daß einige positive und alle negativen Beispiele im minimalen Herbrandmodell des Hintergrundwissens falsch sind $((\exists e \in E^+ : e \text{ ist falsch in } \mathcal{M}(B)) \wedge (\forall e \in E^- : e \text{ ist falsch in } \mathcal{M}(B)))$, und eine Hypothesensprache $\mathcal{LH}$. Alle Sprachen seien definite Theorien.*

Die **definite** *Aufgabenstellung der* **induktiven logischen Programmierung** *besteht im Finden einer Hypothese $H \subset \mathcal{LH}$, für die gilt:*

1. *Alle negativen Beispiele sind falsch im minimalen Modell des Hintergrundwissens und der Hypothese ($\forall e \in E^- : e$ ist falsch in $\mathcal{M}(B \wedge H)$) und*

2. *alle positiven Beispiele sind wahr im minimalen Modell des Hintergrundwissens und der Hypothese ($\forall e \in E^+ : e$ ist wahr in $\mathcal{M}(B \wedge H)$).*

Zusammen mit dem Hintergrundwissen können aus der induzierten Begriffsbeschreibung alle positiven Beispiele gefolgert werden. Darüber hinaus erlaubt die Hypothese möglicherweise die Vorhersage (engl. *prediction*) zum Lernzeitpunkt unbekannter Beispiele des Zielbegriffes.

Lernverfahren für Helfts nicht monotone Semantik zum Lernen in prädikatenlogischen Formalismen entdecken dagegen Regelhaftigkeiten in einer Menge von Beobachtungen. Es gibt keinen Zielbegriff und somit auch keine expliziten Beispiele und Gegenbeispiele[4].

Definition 2.3 (nichtmonotone ILP Semantik) *Gegeben seien Hintergrundwissen B in einer definiten Theorie $\mathcal{LB}$ und eine Hypothesensprache $\mathcal{LH}$.*

Die **nichtmonotone** *Aufgabenstellung der* **induktiven logischen Programmierung** *besteht im Finden einer Hypothese $H \subset \mathcal{LH}$, für die gilt:*

1. *Alle Formeln der Hypothese sind gültig im minimalen Modell des Hintergrundwissens ($\forall h \in H : h$ ist wahr in $\mathcal{M}(B)$),*

2. *die Hypothese ist vollständig (ist eine Formel $g \subset \mathcal{LH}$ wahr in $\mathcal{M}(B)$, dann gilt $H \models g$) und*

3. *H ist minimal, d.h. es gibt keine gültige und vollständige, echte Teilmenge G von H.*

Die Hypothese H beschreibt alle Eigenschaften der Formelmenge B, die in der vorgegebenen Hypothesensprache $\mathcal{LH}$ ausdrückbar sind. Zugleich ist sie

[4] Die folgende Definition stammt aus [Muggleton und Raedt, 1993] und entspricht der Semantik des Systems CLAUDIEN [De Raedt und Bruynooghe, 1993].

die minimale aller Theorien, die diese Eigenschaften erfüllen, da sie keine redundanten Formeln enthält.

Die beiden Definitionen unterscheiden sich in mehreren Punkten. Grundlage zum Lernen von Begriffsbeschreibungen sind Beispiele E des Zielbegriffes. Bei der nicht monotonen Semantik sind die positiven Beispiele Teile des Hintergrundwissens, eine Unterscheidung der Mengen B und E ist hier überflüssig. Alle aus der Formelmenge $(B \wedge E)$ ableitbaren Fakten werden als Beispiele betrachtet, d.h. es gibt Beispiele für verschiedene Relationen und für alle vorkommenden Prädikate werden Regelhaftigkeiten in $(B \wedge E)$ entdeckt (engl. *multiple predicate learner*).

Der wichtigste Unterschied liegt aber darin, daß die nicht monotone Semantik nicht fordert, daß alle positiven Beispiele eines Begriffes von der Hypothese abgedeckt werden $(B \wedge H \models E^+)$. Es wird keine Konzeptbeschreibung gesucht, sondern nur die minimale Theorie in einer Sprache $\mathcal{LH}$, die alle in dieser Sprache ausdrückbaren Regelhaftigkeiten beschreibt. Die gefundenen Regelhaftigkeiten besitzen zudem im Gegensatz zur klassischen ILP Semantik keine Vorhersagekraft innerhalb der vorhandenen Signatur, da durch Definition über das minimale Herbrandmodell ähnlich zur Annahme der abgeschlossenen Welt (engl. *closed world assumption*) indirekt negative Beispiele hergeleitet werden[5].

Zur Verdeutlichung dieses Aspektes soll folgendes Beispiel betrachtet werden:

Beispiel 2.1 *Seien*

$$\begin{aligned} B &= \left\{ \begin{array}{l} \mathsf{weiblich(ann)}, \\ \mathsf{weiblich(sue)} \end{array} \right\} \textit{ und} \\ E^+ &= \{\mathsf{mutter(ann, sue)}\}. \end{aligned}$$

Dann ist

$$H = \mathsf{mutter(X,Y)} \leftarrow \mathsf{weiblich(X)} \;\&\; \mathsf{weiblich(Y)}$$

eine gültige Hypothese der normalen ILP Semantik, die z.B. mutter(sue,ann) *als weiteres Beispiel der Zielrelation vorhersagt. H ist aber keine Lösung für die nicht monotone ILP Semantik, da mit $H\{\mathsf{X/sue, Y/ann}\}$ eine Instanz von H existiert, die im minimalen Modell $M(B \wedge E^+)$ nicht gültig ist (da implizit $E^- = \{\mathsf{mutter(sue, ann)}, \mathsf{mutter(ann, ann)}, \mathsf{mutter(sue, sue)}\}$ gilt).*

Beschränkt man dagegen die Hypothesensprache $\mathcal{LH}$ auf Formelmengen mit nur einer Klausel, dann sind die Hypothesen

$$H_1 = \mathsf{mutter(ann,sue)} \leftarrow \mathsf{weiblich(ann)}$$

und

$$H_2 = \mathsf{weiblich(X)} \leftarrow \mathsf{mutter(X,Y)}$$

[5] Eine formale Diskussion der Unterschiede zwischen Plotkins und Helfts Semantik im Hinblick auf die *Closed World Assumption* findet sich in [Bell und Weber, 1993].

zwei unterschiedliche Lösungen für die nicht monotone ILP Semantik. Wendet man bei der normalen ILP Semantik auch die Annahme der abgeschlossenen Welt an, dann sind alle so gefundenen Hypothesen auch gültig für die nicht monotone Semantik, wie im obigen Beispiel die Klausel H_1. ◇

In realen Anwendungen enthalten die Beispiele und das Hintergrundwissen oft Fehler, d.h. die Daten sind unvollständig, verrauscht oder mit Unsicherheit behaftet (engl. *noise*). Viele ILP–Lernverfahren erlauben daher eine begrenzte Anzahl von Ausnahmen bzgl. der in den beiden ILP–Definitionen gemachten Forderungen.

Auf Grundlage der Theorie der Berechenbarkeit und der Komplexitätstheorie[6] wurden im Rahmen des maschinellen Lernens auch andere Szenarien als die des induktiven logischen Programmierens zur Untersuchung von Lernproblemen entwickelt. Diese Szenarien stellen stärkere Anforderungen an die Korrektheit der gesuchten Hypothese oder an die Berechenbarkeit einer Problemlösung. Die zwei wichtigsten sind Golds Ansatz der *Identifikation im Grenzwert* (engl. *identification in the limit*) [Gold, 1967], basierend auf der Theorie der Berechenbarkeit, und Valiants Ansatz des *wahrscheinlich annähernd korrekten Lernens* (engl. *probably approximately correct (PAC–)learning*) [Valiant, 1984], der Anforderungen an die Komplexität eines Lernverfahrens stellt. Beide Ansätze lassen sich analog zum Lernen einer prädikatenlogischen Begriffsbeschreibung aus Beispielen (Definition 2.1) formalisieren.

Golds Szenarium der Identifikation im Grenzwert beschreibt Lernverfahren, die bei Eingabe einer Folge von Beispielen und Gegenbeispielen eines Begriffes in einer Sprache $\mathcal{LE}$ nach jedem Eingabewert eine Hypothese in einer Sprache $\mathcal{LH}$ ausgeben. Ein Lernverfahren identifiziert die Hypothese H im Grenzwert, wenn ab einem Zeitpunkt (dem Grenzwert) die berechnete Hypothese den Bedingungen des ILP–Problems genügt und sich die berechnete Ausgabe bei weiteren Eingaben nicht mehr verändert. Wichtig ist hierbei, daß dieses Szenarium nicht fordert, daß der Zeitpunkt, ab dem das richtige Ergebnis berechnet wird, bestimmt werden kann. Dieser Grenzwert wird vom Lernverfahren nicht angezeigt. Kietz und Dzeroski haben dies im Rahmen der induktiven logischen Programmierung formalisiert [Kietz und Dzeroski, 1993][7]:

Definition 2.4 (Identifikation im Grenzwert)
Sei $E_\infty = \{e_1, e_2, e_3, \ldots\}$ eine unendliche Folge von Beispielen, wobei $e_i \in E \times \{+, -\}$, und sei E_∞ eine vollständige Aufzählung aller Beispiele in $\mathcal{LE}$, d.h., $\forall e \in \mathcal{LE} : \exists i : (e_i = \langle e, + \rangle \vee e_i = \langle e, - \rangle)$. $E_i = \{e_1, e_2, e_3, \ldots, e_i\}$ bezeichne die Menge aller zum Zeitpunkt i bekannten Beispiele, $E_i^+ = \{\langle e, + \rangle \in$

[6] Die in dieser Arbeit benötigten Grundlagen der Komplexitätstheorie werden in Anhang A rekapituliert.

[7] Eine Aufgabenstellung zum Lernen in prädikatenlogischen Formalismen wird im folgenden als Quadrupel $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ notiert. Dabei steht $\vdash$ für eine Ableitungsregel als Operationalisierung der logischen Folgerung (siehe Abschnitt 2.2.1).

$E_i\}$ sei die Menge aller zum Zeitpunkt i bekannten positiven Beipsiele und $E_i^- = \{\langle e, -\rangle \in E_i\}$ die Menge der negativen Beispiele zum Zeitpunkt i. Ein Hintergrundwissen B heißt genau dann geeignet für E_∞ und $\mathcal{LH}$, wenn $\exists H \in \mathcal{LH} : (B, H \vdash E_\infty^+ \wedge \forall e \in E_\infty^- : B, H \not\vdash e)$.

Ein Lernproblem $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ wird genau dann im Grenzwert von einem Algorithmus identifiziert, wenn dieser Algorithmus für jede Folge E_∞ und jedes geeignete Hintergrundwissen $B \in \mathcal{LB}$ E_∞ inkrementell als Eingabe akzeptiert, nach jedem Beispiel ein $H_i \in \mathcal{LH}$ berechnet und es einen Zeitpunkt i gibt, für den gilt: $\forall j \geq i : H_i = H_j$ und $(B, H_i \vdash E_\infty^+ \wedge \forall e \in E_\infty^- : B, H_i \not\vdash e)$.

Eines der wichtigsten Resultate zur Identifikation im Grenzwert stammt von Shapiro. Er wies nach, daß sein MIS–System für ein h–leichtes Modell mit Unterstützung eines Orakels eine Theorie im Grenzwert identifiziert [Shapiro, 1981].

Zur Beurteilung eines Lernverfahrens ist es jedoch relevanter, Aussagen über die Effizienz dieses Verfahrens zu gewinnen. Die Komplexität eines Problems untersucht die Komplexitätstheorie, auf der das Paradigma des wahrscheinlich annähernd korrekten (PAC–) Lernens beruht.

Beim wahrscheinlich annähernd korrekten Lernen fordert man, daß das Lernverfahren nach Eingabe einer polynomiell beschränkten Anzahl von Beispielen und nach polynomieller Zeit die Hypothese bestimmt. Dabei wird in Abschwächung der Forderung der Identifikation im Grenzwert nach einer korrekten Hypothese versucht, mit einer hohen Wahrscheinlichkeit ein Ergebnis mit kleinem Fehler zu finden.

Eine Definition dieses Paradigmas im Rahmen der induktiven logischen Programmierung geben Cohen [Cohen, 1993] und Kietz und Dzeroski [Kietz und Dzeroski, 1993].

Definition 2.5 (wahrscheinlich annähernd korrekt (PAC-) lernbar) *Ein Lernproblem $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ heißt genau dann wahrscheinlich annähernd korrekt (PAC–) lernbar, wenn ein Algorithmus PACLEARN und eine polynomielle Funktion $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_t, n_b)$ existieren, so daß für alle $n_e > 0$, $n_t > 0$, $n_b > 0$, jedes Hintergrundwissen $B \in \mathcal{LB}$ mit Komplexität n_b oder geringer, jeden gesuchten Begriff $C \in \mathcal{LH}$, für alle $\epsilon : 0 < \epsilon < 1$, für alle $\delta : 0 < \delta < 1$, und für jede Wahrscheinlichkeitsverteilung D, für alle Beispielmengen E^+, E^- für C, die als Stichprobe einer Größe von wenigstens $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_t, n_b)$ aus $\mathcal{LE}$ bzgl. D gezogen wurden,*

1. *nach Eingabe von E^+, E^-, ϵ und δ der Algorithmus PACLEARN eine Hypothese $H \in \mathcal{LH}$ berechnet, für die*

$$Prob(D(H \Delta C) > \epsilon) < \delta$$

gilt, wobei $H_1 \Delta H_2$ die symmetrische Differenz der Mengen $\{e \in E \mid H_1 \wedge B \vdash e\}$ und $\{e \in E \mid H_2 \wedge B \vdash e\}$ bezeichnet,

2. *die Laufzeit von PACLEARN polynomiell in* $\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_t, n_b$ *und der Anzahl der Beispiele ist, und*

3. *für alle* $e \in LE$ *der Test* $H, B \vdash e$ *in polynomieller Zeit ausgewertet werden kann.*

Die polynomielle Funktion $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_t, n_b)$ *heißt die Beispielkomplexität des Lernalgorithmus PACLEARN.*

Resultate zum PAC–Lernen lassen sich durch einen Zusammenhang mit der Komplexitätsklasse *RP* mit Ergebnissen für ILP Probleme in Beziehung setzen. Jörg–Uwe Kietz adaptierte einen Satz von Pitt und Valiant [Pitt und Valiant, 1988], der zeigt, wie ein wahrscheinlich annähernd korrektes Lernverfahren benutzt werden kann, um das korrespondierende ILP–Problem[8] wahrscheinlich in polynomieller Zeit zu lösen [Kietz, 1993b].

Satz 2.1 *Ist ein Lernproblem* $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ *PAC–lernbar, dann fällt das Konsistenzproblem für* $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ *in die Klasse RP. Dies bedeutet negativ formuliert: Liegt das Konsistenzproblem nicht in RP, so ist* $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ *nicht PAC–lernbar.*

Danach lassen sich Probleme, die *NP*–schwierig oder *PSPACE*–schwierig sind, nicht PAC–lernen, es sei denn $NP = RP$ oder $PSPACE = RP$.

Pitt beschreibt eine Abschwächung des PAC–Paradigmas. Bei der **polynomiellen Vorhersagbarkeit** (engl. *polynomially predictable*) verzichtet man darauf, daß die Hypothese, die das Lernverfahren ausgibt, in einer bestimmten Sprache formulierbar ist, also auf die Bedingung $H \in \mathcal{LH}$ [Cohen, 1993]. Die Hypothese darf in einer beliebig komplexen Repräsentation dargestellt sein. Das PAC–Lernen ist ein Spezialfall dieses Ansatzes.

2.2 Komplexität des prädikatenlogischen Suchraumes

Das induktive Lernen von Begriffen kann als Suche im Raum aller Ausdrücke der Hypothesensprache aufgefaßt werden [Mitchell, 1982]. Ein einfacher Aufzählungsalgorithmus zählt alle in dieser Sprache enthaltenen Ausdrücke auf und prüft für jeden Ausdruck, ob er die Bedingungen aus der Definition des induktiven Lernproblems erfüllt. Beim ILP–Problem ist der resultierende Suchraum meist nicht endlich. Zudem besitzt ein Aufzählungsalgorithmus keine Möglichkeit die Suche zu beschneiden, da er nacheinander alle Ausdrücke einer aufzählbaren Sprache konstruieren muß.

Um den Suchvorgang effizienter zu organisieren, ist es notwendig, den Suchraum zu strukturieren. Zu diesem Zweck verwendet man beim ILP–Problem

[8] Ohne Zusatz ist im weiteren Verlauf der Arbeit stets die klassische ILP Semantik gemeint, die Verwendung der nicht monotonen Semantik wird ausdrücklich erwähnt.

zumeist eine allgemeiner–als–Relation[9], um den Hypothesenraum hierarchisch anzuordnen. Eine solche Halbordnung hat als speziellste Elemente die positiven Beispiele und als generellstes Element einen sowohl positive wie auch negative Beispiele abdeckenden Ausdruck. Mittels einer solchen Relation kann der Suchraum gezielt durchwandert und Teilräume abgeschnitten werden (engl. *pruning*).

Anhand der Suchstrategie kann man grundsätzlich zwischen zwei Verfahren unterscheiden. Beim *Bottom–up* Verfahren wird ausgehend von den positiven Beispielen innerhalb der Halbordnung solange nach generelleren Ausdrücken gesucht, bis eine Hypothese alle positiven Beispiele abdeckt. Beim *Top–down* Verfahren wird ein allgemeinster Ausdruck solange spezialisiert, bis er kein negatives Beispiel mehr abdeckt.

2.2.1 Generalisierungsmodelle

Der logische Allgemeinheitsbegriff ist modelltheoretisch definiert: Ein Ausdruck $C1$ ist allgemeiner als ein Ausdruck $C2$, wenn jedes Modell von $C1$ ein Modell von $C2$ ist. Dies läßt sich mit Hilfe der logischen Folgerung beschreiben.

Definition 2.6 (logischer Allgemeinheitsbegriff)
Eine Formel $C1$ heißt genau dann **allgemeiner als** *eine Formel $C2$ bezüglich einer Theorie B, wenn gilt:*

$$B \wedge C1 \models C2.$$

In der Literatur zum induktiven logischen Programmieren finden sich verschiedene allgemeiner–als–Relationen über Mengen von Klauseln. Die entsprechenden Definitionen basieren vor allem auf unterschiedlichen Berücksichtigungen von Hintergrundwissen. Alle Generalisierungsmodelle sind Spezialfälle der logischen Folgerung. Sie sind als korrekte, aber nicht vollständige Ableitungsregeln charakterisierbar: Seien C und D Klauseln, $\vdash$ eine allgemeiner–als–Relation und T eine logische Theorie. Dann gilt $T \wedge C \vdash D \Rightarrow T \wedge C \models D$. Die Theorie T entspricht beim Lernproblem dem Hintergrundwissen.

Die jeweiligen Generalisierungsmodelle ordnen Ausdrücke der Hypothesensprache nicht nur unterschiedlich an, sondern definieren auch verschiedene Suchräume, da sie auch zur Überprüfung möglicher Lösungen eines Lernproblems dienen. Eine gültige Hypothese H muß allgemeiner sein als die positiven Beispiele ($B \wedge H \vdash E^+$), darf aber keine negativen Beispiele abdecken ($\forall e \in E^- : B \wedge H \not\vdash E^-$). Da die verwendete allgemeiner–als–Relation $\vdash$ somit Einfluß auf die Berechenbarkeit des zugehörigen ILP–Problems hat, wird ein Lernproblem im folgenden als Tupel $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$ formalisiert.

[9] Diese werden in der Literatur oft als Generalisierungsmodelle bezeichnet.

Für jede Relation $\vdash$ läßt sich eine Äquivalenzrelationen definieren: Zwei Klauseln C und D sind genau dann äquivalent bezüglich $\vdash$ ($C \sim_{\vdash} D$), wenn $C \vdash D$ und $D \vdash C$ gilt[10].

Niblett untersuchte die logische Implikation zur Beschreibung der Generalisierungsbeziehung von Hornklauseln [Niblett, 1988]. Zwischen zwei Hornklauseln ($B = \emptyset$) ist sie entscheidbar[11], mit Hintergrundwissen ($B \neq \emptyset$) ist sie halb–entscheidbar. ILP–Probleme unter Verwendung der logischen Implikation zur Beschreibung der allgemeiner–als–Relationen sind sogar bei sehr eingeschränkten Repräsentationssprachen nicht mehr entscheidbar[12].

Plotkin wies nach, daß die schwächere θ–Subsumtion[13] günstigere Eigenschaften für den Induktionsprozeß hat [Plotkin, 1970].

Definition 2.7 (θ–Subsumtion, $\vdash_\theta$) *Eine Klausel C* **θ–subsumiert** *eine Klausel D, geschrieben als $C \vdash_\theta D$, genau dann, wenn es eine Substitution θ für Termvariablen[14]gibt, für die gilt:*

$$C\theta \subseteq D.$$

Beispiel 2.2

$$\begin{gathered}\{\mathsf{weiblich(X)}, \neg\mathsf{mutter(X,Z)}, \neg\mathsf{mutter(X,Y)}\}\\ \vdash_\theta \\ \{\mathsf{weiblich(X)}, \neg\mathsf{mutter(X,Y)}, \neg\mathsf{tochter(X,Z)}\}\end{gathered}$$

mit $\theta = \{\mathsf{Z/Y}\}$. $\diamond$

Robinson wies nach, daß die θ–Subsumtion zwischen Klauseln entscheidbar ist [Robinson, 1965], $\vdash_\theta$ kann für beliebige Hornklauseln aber nicht effizient getestet werden. Selbst der Test $D \vdash_\theta C$ mit Klauseln C konstanter und geringer Größe ist nicht effizient berechenbar, wie eine Reduktion des *NP*–vollständigen $3SAT$–Problems auf die θ–Subsumtion zwischen entsprechenden Hornklauseln beweist[15]. Außerdem berücksichtigt die

[10]Wird im folgenden von einer Generalisierung (Spezialisierung) bzgl. einer Relation $\vdash$ gesprochen, so schließt das die Äquivalenz stets ein. Der Ausschluß dieses Falles soll als strikte Generalisierung (Spezialisierung) bezeichnet werden.

[11]In [Kietz und Dzeroski, 1993] wird gezeigt, daß dies nur gilt, wenn die allgemeinere Klausel generativ (siehe Abschnitt 2.2.2) ist.

[12]Einige negative Ergebnisse werden in [Kietz und Dzeroski, 1993] wiedergegeben.

[13]Erstmals von Robinson definiert [Robinson, 1965].

[14]Mit Termvariable sei eine Variable für Terme erster Ordnung bezeichnet. Dies dient der Unterscheidung zu Variablen für andere Ausdrücke, die in Kapitel 4 eingeführt werden.

[15]Diese Aussage wurde von Jörg–Uwe Kietz bewiesen und ist mit dem Beweis in [Kietz und Lübbe, 1994] veröffentlicht. Nach [Garey und Johnson, 1979] wurde die *NP*–Vollständigkeit der θ–Subsumtion zuerst in einem unveröffentlichten Papier von Baxter (1977) bewiesen. Im Gegensatz zu anderen veröffentlichten Beweisen der *NP*–Vollständigkeit von $D \vdash_\theta C$ behauptet der Satz von Kietz, daß das Problem auch für konstante Größen von $|C|$ *NP*–schwierig bleibt.

θ–Subsumtion kein Hintergrundwissen, da sie nur zwischen zwei Klauseln definiert ist.

In jeder Äquivalenzklasse $[C]$ einer Klausel C unter der Relation $\sim_\theta$[16] gibt es genau eine ausgezeichnete Klausel. Plotkin nannte diese Vertreter der Äquivalenzklassen *reduzierte* Klauseln, sie sind bis auf Variablenumbennung eindeutig und enthalten keine bezüglich der θ–Subsumtion redundanten Literale.

Definition 2.8 *Eine Klausel C heißt* **reduziert**, *wenn aus $D \subseteq C$ und $C \sim_\theta D$ folgt, daß C und D alphabetische Varianten sind.*

Zur Konstruktion einer reduzierten Klausel C aus einer Klausel E benötigt man ebenfalls einen Test auf θ–Subsumtion [Plotkin, 1970]. Zu testen, ob eine reduzierte Klausel vorliegt, ist *co–NP*–vollständig [Gottlob und Fermüller, 1993] und somit wird es unter der Annahme $P \neq NP$ keinen effizienten Algorithmus zur Reduktion von Hornklauseln geben.

In Abschnitt 4.3.3 wird gezeigt, daß der Test $\vdash_\theta$ für bestimmte Hypothesensprachen dennoch polynomiell berechenbar ist. Die zugrundeliegenden Ideen führen außerdem noch zu verbesserten Algorithmen für die θ–Subsumtion und die Reduktion bei beliebigen Hornklauseln.

Um Hintergrundwissen zu berücksichtigen, erweiterte Plotkin die θ–Subsumtion zur Generalisierung relativ zu einer Theorie [Plotkin, 1971b].

Definition 2.9 (Generalisierung bezüglich einer Theorie, $\vdash_B^{rel}$)
Seien C und D Klauseln, B ein logische Theorie. Klausel C ist genau dann eine **Generalisierung** *von Klausel D* **bezüglich** *B, Notation $C \vdash_B^{rel} D$, wenn es eine Klausel E gibt, so daß $C \vdash_\theta E$, $P \wedge E \models D$ und $P \wedge D \models E$.*

Plotkin wies folgenden Zusammenhang zur Resolution nach.

Satz 2.2 *$C \vdash_B^{rel} D$ gdw. D eine Tautologie ist oder es eine Ableitung von D gibt($B \wedge C \vdash_{SLD} D$), in der C höchstens einmal vorkommt.*

Die relative Generalisierung ist somit ein Spezialfall der logischen Folgerung. Während bei der relativen Generalisierung die generellere Klausel in einer Ableitung der spezielleren nur einmal vorkommen darf, gilt diese Einschränkung für die logische Folgerung nicht.

Beispiel 2.3 *Sei*

$$B = \left\{ \begin{array}{rcl} \mathsf{weiblich(X)} & \leftarrow & \mathsf{mutter(X,Y)}, \\ \mathsf{großmutter(X,Z)} & \leftarrow & \mathsf{elternteil(X,Y)} \ \& \ \mathsf{elternteil(Y,Z)} \\ & & \& \ \mathsf{weiblich(X)} \end{array} \right\}.$$

[16] Zur besseren Lesbarkeit wird die Äquivalenz bzgl. $\vdash_\theta$ entgegen der eingeführten Notation nicht als $\sim_{\vdash_\theta}$ geschrieben.

Dann gilt

$$\begin{array}{c}\mathsf{elternteil(Y,X)} \leftarrow \mathsf{mutter(Y,X)} \\ \vdash_B^{rel} \\ \mathsf{großmutter(X,Z)} \leftarrow \mathsf{mutter(X,Y)}\ \&\ \mathsf{elternteil(Y,Z)}.\end{array}$$

◇

Das Kopfliteral der Generalisierung einer Klausel relativ zu einer Theorie muß im Gegensatz zur θ–Subsumtion nicht mit dem Kopfliteral der spezielleren Klausel unifizierbar sein, d.h. zwei Klauseln, die bezüglich $\vdash_B^{rel}$ in Relation stehen, müssen nicht dieselben Begriffe beschreiben (im obigen Beispiel $\mathsf{großmutter}$ und $\mathsf{elternteil}$). Auch für den Fall $B = \emptyset$ sind die beiden Generalisierungsmodelle nicht völlig äquivalent, aber es gilt: $C \vdash_\emptyset^{rel} D$ genau dann, wenn $C \vdash_\theta D$ oder D eine Tautologie ist [Plotkin, 1971b].

Ein Spezialfall der relativen Generalisierung ist Buntines Definition der generalisierten Subsumtion [Buntine, 1988]. Bei dieser Generalisierungsbeziehung beschreibt eine allgemeinere Klausel wie bei der θ–Subsumtion wieder den gleichen Begriff wie die speziellere.

Definition 2.10 (generalisierte Subsumtion, $\vdash_B$) *Eine Klausel C* **covers** *ein Grundatom A in einer Interpretation I, wenn es eine Substitution θ gibt, so daß gilt:*

$$C_{Kopf}\theta = A \text{ und } \exists(C_{Körper}\theta) \text{ ist wahr in der Interpretation } I.$$

Klausel C **subsumiert** *Klausel D* **bezüglich eines logischen Programmes** *B, wenn für jede Herbrand Interpretation I, in der B wahr ist, und für jedes Atom A gilt:*

$$D \text{ covers } A \Rightarrow C \text{ covers } A$$

Als Schreibweise wird $C \vdash_B D$ benutzt.

Buntine beweist folgende operationale Anleitung für einen generalisierten Subsumtionstest:

Satz 2.3 *Seien C und D Klauseln mit disjunkten Variablen. Sei B ein logisches Programm. Mit θ sei eine Substitution bezeichnet, die für alle Variablen in D neue Konstanten, die nicht in C, D oder B vorkommen, einführt. Dann gilt $C \vdash_B D$, gdw. eine beliebige Substitution σ existiert, so daß gilt:*

$$\begin{array}{c}C_{Kopf}\sigma = D_{Kopf} \quad \text{und} \\ B \wedge D_{Körper}\theta \models \exists(C_{Körper}\sigma\theta).\end{array}$$

Beispiel 2.4 *Sei $B = \{\mathsf{elternteil(X,Y)} \leftarrow \mathsf{tochter(Y,X)}\}$.*
Dann gilt

$$\begin{array}{c}\{\mathsf{mutter(X,Y)}, \neg\mathsf{elternteil(X,Y)}, \neg\mathsf{weiblich(X)}\} \\ \vdash_B, \not\vdash_\theta \\ \{\mathsf{mutter(X,Y)}, \neg\mathsf{tochter(Y,X)}, \neg\mathsf{weiblich(X)}\}.\end{array}$$

Diese beiden Klauseln stehen bezüglich der θ–Subsumtion nicht in Relation.

◇

Buntine zeigt, daß für den Fall $B = \emptyset$ die generalisierte Subsumtion der θ–Subsumtion entspricht [Buntine, 1988]. Plotkin untersuchte bereits vor Buntine einen entsprechenden Spezialfall seiner relativen Generalisierung [Plotkin, 1971b].

Die einzelnen Generalisierungsmodelle sind alle Spezialfälle der logischen Folgerung. Zusammenfassend ergeben sich folgende Implikationen:

$$\begin{array}{ll} C \vdash_\theta D & \Rightarrow \\ C \vdash_B D & \Rightarrow \\ C \vdash_B^{rel} D & \Rightarrow \\ B \wedge C \models D. & \end{array}$$

Der Vergleich in Tabelle 2.1 ist eine Erweiterung einer Übersicht von Jung [Jung, 1993].

Generalisierungsmodell	B	$C_{Head}\theta = D_{Head}$	$\#\, C$ in Beweis für D	Berechenbarkeit
$\vdash_\theta$	nein	ja	0	NP–schwierig
$\vdash_B$	ja	ja	≤ 1	halb–entscheidbar
$\vdash_B^{rel}$	ja	nein	≤ 1	halb–entscheidbar
$\models$	ja	nein	beliebig	halb–entscheidbar

Tabelle 2.1: Vergleich von Generalisierungsmodellen

Weitere allgemeiner–als–Relationen wurden von Nienhuys-Cheng, van der Laag und van der Torre untersucht [Nienhuys-Cheng *et al.*, 1993]. Alle Generalisierungsmodelle enthalten nur Testmethoden, aber keine Konstruktionsvorschriften für eine Generalisierung. Solche Anleitungen geben Generalisierungsoperatoren, wie inverse Resolutionsoperatoren [Muggleton, 1992; Jung, 1993] und speziellste Generalisierungen [Plotkin, 1970; Haussler, 1989].

2.2.2 Generalisierungsmodelle und Hintergrundwissen

Im vorangegangenen Kapitel wurden schon einige Äquivalenzen zwischen den Generalisierungsmodellen aufgezeigt. So gilt für den Fall $B = \emptyset$ die Beziehung $C \vdash_\theta D \Leftrightarrow C \vdash_B D$. Es gibt weitere Einschränkungen unter denen man die halb–entscheidbaren Generalisierungsmodelle auf die θ–Subsumtion zurückführen kann.

Ist das Hintergrundwissen variablen– und funktionsfrei, können die Beispiele so zu erweiterten Klauseln umgeformt werden, daß man das Hintergrundwissen beim Lernen nicht mehr berücksichtigen muß. Zur Generalisierung reicht dann das Modell der θ–Subsumtion, und dieses ist bei funktionsfreien, nicht rekursiven Hornklauseln eine korrekte und vollständige Ableitung: $h \models h'$, gdw. $h\theta \subseteq h'$.

Definition 2.11 (funktionsfrei) *Eine Klausel heißt* **funktionsfrei**, *wenn sie nur Prädikatssymbole, Variablen und Konstanten (nullstellige Funktionssymbole) enthält.*

Liegt kein variablenfreies Hintergrundwissen vor, so kann man dies vor dem eigentlichen Lernprozeß erzeugen. Eine Methode liegt in der Ersetzung aller Variablen in Klauseln aus B durch die in den Beispielen vorkommenden Terme. Dieses Vorgehen findet man bei Robinson unter dem Stichwort Saturierung [Robinson, 1965].

Beschränkt man die Sprache $\mathcal{LB}$ auf generative Klauseln so läßt sich in Verbindung mit einem tiefenbeschränkten Inferenzprozeß ebenfalls variablenfreies Hintergrundwissen herstellen. Dies entspricht der Definition von h–leichten Herbrandmodellen, wie sie Shapiro benutzt [Shapiro, 1981].

Definition 2.12 (generative Hornklauseln) *Eine Hornklausel heißt* **generativ**, *wenn alle Variablen der Konklusion auch in den Prämissen vorkommen.*

Fordert man neben generativen Klauseln eine feste, höchste Stelligkeit der Prädikate des Hintergrundwissens B und eine maximale Anzahl von Literalen in diesen Klauseln, so lassen sich alle Ableitungen aus dem Hintergrundwissen in einer Zeit polynomiell zur Größe von B berechnen [Wrobel, 1987].

Das oben bereits angesprochene Verfahren, das Hintergrundwissen in die Beispiele zu transformieren, findet sich in der Literatur ebenfalls unter dem Begriff Saturierung[17] [Rouveirol, 1991]. Saturierung ist hier aber nicht gleichzusetzen mit der Definition von Robinson.

Definition 2.13 (elementare Saturierung) *Sei $C1$ die Klausel $H_e \leftarrow B_e$ und $C2$ die Klausel $H_s \leftarrow B_s$ mit $B_s \vdash_\theta B_e$. Die Klausel D mit $H_e \leftarrow B_e \wedge H_s\theta$ heißt* **elementare Saturierung** *von C_e durch C_s.*

Definition 2.14 (Saturierung) *Sei B ein logisches Programm und C_e eine Klausel. Die* **Saturierung** *von C_e in B ist die transitive Hülle der elementaren Saturierung von C_e durch Klauseln aus B.*

Jung beweist, daß der Test $H \vdash_B e$ bei generativem Hintergrundwissen B äquivalent zu $H \vdash_\theta e\!\downarrow$ ist, wobei $e\!\downarrow$ die Saturierung von e ist [Jung, 1993, Theorem 4]. Durch die Wahl von funktionsfreien Fakten als Sprache für B und E läßt sich das induktive Lernproblem mit Hintergrundwissen

[17] In der Literatur zum logischen Programmieren taucht der Begriff Saturierung auch noch mit anderen Definitionen auf. So definieren Bossu und Siegel: Eine Menge von geordneten Klauseln C heißt **saturiert** bezüglich der Resolventen des ersten Literals, wenn jeder Resolvent der ersten Literale von zwei Klauseln aus C durch eine Klausel aus C subsumiert wird [Bossu und Siegel, 1985].

B daher auf das Lernen ohne Hintergrundwissen zurückführen. Dazu setzt man $\mathcal{LB}_{neu} = \emptyset$ und $\mathcal{LE}_{neu} = \{e\downarrow \mid e \in E\} = \{e \leftarrow B \mid e \in E\}$. Beispiele sind nun variablenfreie Hornklauseln.

Die ILP Probleme ($\vdash_B$,variablenfreies B,Fakten,$\mathcal{LH}$) und ($\vdash_\theta$,$\emptyset$,$\mathcal{LE}_{neu}$,$\mathcal{LH}$) sind somit äquivalent. Für bestimmte Hypothesensprachen ist es sogar vollständig, die Beispiele e durch Klauseln $e\leftarrow B'$ zu ersetzen, wobei B' eine kleine Teilmenge von B ist und für alle $H \in \mathcal{LH}$ gilt:

$$H \vdash_\theta e \leftarrow B \Leftrightarrow H \vdash_\theta e \leftarrow B'.$$

Beispiel 2.5 *Seien*

$$\begin{aligned} B &= \left\{ \begin{array}{l} \mathsf{tochter(sue, ann)}, \\ \mathsf{elternteil(sue, tom)}, \\ \mathsf{schwester(sue, mary)} \end{array} \right\} \textit{ und} \\ E^+ &= \{\mathsf{mutter(ann, sue)}\}. \end{aligned}$$

Beschränkt man die Hypothesensprache $\mathcal{LH}$ auf Klauseln bei denen alle Variablen im Körper der Klausel auch im Kopf vorkommen müssen (Tiefe 0), dann ist es für jede mögliche Hypothese H vollständig,

$$H \vdash_\theta \mathsf{mutter(ann, sue)} \leftarrow \mathsf{tochter(sue, ann)}$$

zu testen, um H als Lösung des ILP Problems zu identifizieren. ◇

2.2.3 Einschränkung der Repräsentationssprachen

Im vorangegangenen Abschnitt wurden verschiedene Generalisierungsmodelle und Methoden zur Beschränkung des Hintergrundwissens beschrieben, die die Unentscheidbarkeit der Deduktion umgehen und somit zu einer Lösung des ILP–Problems beitragen. Im folgenden sollen nun einige negative Resultate zum Lernen von Hornklauseln im Allgemeinen, sowie positive und negative Ergebnisse für verschiedene Restriktionen der Sprachen $\mathcal{LB}$, $\mathcal{LH}$ und $\mathcal{LE}$ wiedergegeben werden.

Zur besseren Lesbarkeit wird in dieser Arbeit eine Notation für Sprachen von Cohen übernommen [Cohen, 1993]. Sei L eine Menge von Klauseln. Durch Hochstellung wird die Anzahl der erlaubten Klauseln festgelegt. So ist L^1 die Sprache der Programme mit einer Klausel, L^k ist ein Programm mit k Klauseln. Durch ein Subskript wird die Einschränkung der Sprache festgehalten. Sind rekursive Klauseln erlaubt, so wird dieses durch ein ebenfalls hochgestelltes ^{rec} angezeigt.

Kietz wies nach, daß die im folgenden vorzustellenden Lernprobleme von Plotkin und Helft nicht effizient berechenbar sind [Kietz, 1993a].

Plotkins Ansatz war einer der ersten zum ILP–Problem [Plotkin, 1970]. Er benutzte Klausellogik als Hypothesen– und Beispielsprache, kein Hintergrundwissen und θ–Subsumtion als Generalitätsrelation. Helft schränkt Plotkins Problem auf verbundene (engl. *linked*), funktionsfreie Hornklauseln als Sprache $\mathcal{LH}$ ein. Beispiele sind Grundklauseln aus $\mathcal{LH}$ [Helft, 1987].

Definition 2.15 (verbunden, $\mathcal{LH}_{LINKED}$) *Eine Hornklausel heißt* **verbunden**, *wenn alle ihre Literale verbunden sind. Ein Literal ist verbunden, wenn wenigstens einer seiner Terme verbunden ist. Ein Term ist verbunden mit einer* **Verbindungskette** *der Länge 0, wenn er im Kopf einer Klausel vorkommt. Ein Term in einem Literal ist mit der Verbindungskette der Länge $d+1$ verbunden, wenn ein anderer Term in demselben Literal mit der Verbindungskette der Länge d verbunden ist. Die* **Tiefe eines Terms** *ist die minimale Länge seiner Verbindungsketten.*

Als Hintergrundwissen benutzt Helft eine Ungleichheitsbeziehung, die die Verschiedenheit von Objekten beschreibt. Das Hintergrundwissen wird durch die generalisierte Subsumtion in den Lernprozeß einbezogen. Wie im letzten Abschnitt beschrieben, ist dies äquivalent zur θ–Subsumtion von saturierten Klauseln. Plotkins Problem ist also allgemeiner als Helfts Problem.

Kietz reduziert eine *NP*–vollständige Problemstellung von Haussler zum Lernen von speziellsten Generalisierungen auf beide Probleme. Dadurch läßt sich zeigen, daß beide Probleme *NP*–schwierig sind, also das Lernen von funktionsfreien Hornklauseln ohne Hintergrundwissen nicht effizient berechenbar ist. Wie in Kapitel 2.1 geschildert sind sie unter der Annahme $RP \neq NP$ somit auch nicht PAC–lernbar.

Page und Frisch konnten dagegen beweisen, daß eine Hypothese mit einer einzigen *constrained* Klausel als Hypothesensprache $\mathcal{LH}^1_{CONSTRAINED}$ in polynomieller Zeit wahrscheinlich annähernd korrekt lernbar ist [Page und Frisch, 1992].

Definition 2.16 (constrained Klauseln, $\mathcal{LH}_{CONSTRAINED}$)
Eine Klausel heißt **constrained**, *wenn alle Variablen des Klauselkörpers auch im Kopf vorkommen.*

Muggleton und Feng benutzen *ij–determinierende* Klauseln zur Restriktion der Sprache $\mathcal{LH}$ [Muggleton und Feng, 1992]. Determinierende Klauseln setzen eine Ordnung über den Literalen der Klausel voraus und erlauben nur solche Literale im Klauselkörper, die nur dann neue Variablen einführen, wenn diese bezüglich der Ordnung und dem Hintergrundwissen in einer funktionalen Abhängigkeit zu den Termen niedriger Ordnung stehen.

Definition 2.17 (ij–determinierende Hornklauseln, $\mathcal{LH}_{ij-DET}$)
Eine Hornklausel h heißt **determinierend** *bezüglich der Beispiele E und dem variablenfreien Hintergrundwissen B, wenn jeder Term aus h durch eine determinierende Verbindungskette verbunden ist. Ein Term, der im Klauselkopf vorkommt, ist durch eine determinierende Verbindungskette der Länge 0 verbunden.*
Sei $h = \{A, \neg B_1, \ldots, \neg B_m, \neg B_{m+1}, \ldots, \neg B_n\}$ eine geordnete Menge. Der Term t aus dem Literal $\neg B_{m+1}$ ist genau dann mit einer determinierenden Verbindungskette der Länge $d+1$ verbunden, wenn alle Terme der Menge

$\{A, \neg B_1, \ldots, \neg B_m\}$ *mit einer determinierenden Verbindungskette von höchstens der Länge d verbunden sind und es für alle Substitutionen* θ *mit der Eigenschaft* $A\theta \in E^+$ *und* $\{\{B_1\}, \ldots, \{B_m\}\}\theta \subseteq B$ *nur eine einzige Substitution* δ *gibt, für die* $\{B_{m+1}\}\theta\delta \in B$ *gilt. Die* **determinierende Tiefe eines Terms** *ist die minimale Länge seiner determinierenden Verbindungsketten.*
Eine Klausel mit maximaler determiniernder Tiefe i ihrer Terme und maximaler Stelligkeit der Literale j heißt **ij–determinierend**.

Beispiel 2.6 *Die Klausel*

$$\mathsf{großmutter(X,Z)} \leftarrow \mathsf{mutter(Y,Z)} \;\; \& \; \mathsf{mutter(X,Y)}$$

ist 12*–determinierend, die Klausel*

$$\mathsf{großmutter(X,Z)} \leftarrow \mathsf{mutter(X,Y)} \;\; \& \; \mathsf{elternteil(Y,Z)}$$

dagegen nicht, da jede Mutter X *mehrere Kinder* Y *haben kann und die Variable* Y *nicht im Kopf der Klausel vorkommt.* ◇

Muggleton und Feng beweisen, daß sie mit Grundhintergrundwissen und ij–determinierenden Klauseln als Hypothesensprache für feste Werte i und j Hypothesen mit höchstens $O((tfm)^{i^j})$ Literalen im Klauselkörper konstruieren können. Dabei ist t die Anzahl der Terme in E, m ist die Anzahl der verschiedenen Prädikate aus B und f die maximale Anzahl der Subterme in einem Prädikat aus B, bei funktionsfreien Klauseln also das Maximum von j und der Stelligkeit des Klauselkopfes. Die Länge der Generalisierung ist also im Gegensatz zum Generalisierungsmodell von Plotkin nicht exponentiell in der Anzahl der Beispiele aus E, da sie in der Tiefe beschränkt ist.

Dzeroski, Muggleton und Russel gelang der Beweis, daß eine k–Disjunktion von ij–determinierenden, funktionsfreien, nicht rekursiven Hornklauseln (die Sprache $\mathcal{LH}^k_{ij-DET}$) PAC–lernbar bei einfachen Verteilungen ist [Dzeroski *et al.*, 1992]. Zum Beweis konstruierten sie eine Abbildung dieses ILP–Problems auf eine höchstens polynomiell längere Darstellung in der Aussagenlogik. Damit wurde aber auch gezeigt, daß das Lernen von funktionsfreien ij–determinierenden Hornklauseln aus Grundbeispielen und variablenfreiem Hintergrundwissen bei festen Werten für i und j nicht stärker als Lernen in Aussagenlogik ist.

Hebt man die Tiefenbeschränkung bei ij–determinierenden Hornklauseln auf, wird die Komplexität des ILP–Problems entscheidend vergrößert [Kietz, 1993b]. So sind determinierende verbundene Hornklauseln ohne Tiefenbeschränkung $\mathcal{LH}^1_{2-DET}$ nicht polynomiell PAC–lernbar. Zum Beweis reduziert Kietz ein *PSPACE*–vollständiges Problem aus der Automatentheorie auf das Konsistenzproblem zum Lernen einer einzigen 2–deterministischen Hornklausel ohne Tiefenbeschränkung. Dadurch zeigt er, daß dieses Problem *PSPACE*–schwierig ist.

Auch deterministische Klauseln mit einer Tiefenbeschränkung logarithmisch zur Größe der Beispiele $\mathcal{LH}^1_{(log|E|)j-DET}$ lassen sich nicht polynomiell vorhersagen und sind somit auch nicht PAC–lernbar [Cohen, 1993].

Bei nicht determinierenden Klauseln kommt man selbst mit einer Tiefenbeschränkung nicht auf positive Resultate.

Definition 2.18 (ij–nicht determinierende (ij–verbundene) Klauseln, $\mathcal{LH}_{ij-notDET}$)
Verbundene Klauseln mit maximaler Tiefe i ihrer Terme und maximaler Stelligkeit j ihrer Literale heißen **ij–nicht determinierend (ij–verbunden)**.

So ist bereits das Problem, eine einzige 12–nicht determinierende Hornklausel, also die Sprache $\mathcal{LH}^1_{12-notDET}$, zu lernen, nicht PAC–lernbar, da sich das *NP*–vollständige SAT–Problem auf das entsprechende Konsistenzproblem reduzieren läßt [Kietz, 1993b].

Im übrigen entsprechen constrained Klauseln den $0j$–verbundenen (und somit auch den $0j$–determinierenden) Klauseln.

Erlaubt man Rekursion in den obigen Sprachen, sind alle Resultate nicht mehr gültig.

Einschränkungen anderer Art machen Hypothesensprachen auf Grundlage von syntaktischen Einschränkungen durch Schemata. Zu den Schema–basierten Ansätzen gehören die in dieser Arbeit untersuchten Regelmodelle [Kietz und Wrobel, 1991]. Regelmodelle definieren eine Sprache mit Klauseln, die gewissen syntaktischen Vorgaben folgen (Für Einzelheiten sei auf Kapitel 3 verwiesen). Verwandte Ansätze sind die Programmierschemata in CAN [Tausend, 1992] und die relationalen Klischees von FOCL [Pazzani und Kibler, 1992].

2.3 Speziellste Generalisierungen

Auf Grundlage der allgemeiner–als–Relation kann eine ausgezeichnete Hypothese für den Induktionsprozeß definiert werden. Eine *speziellste Generalisierung H* von mehreren Klauseln ist dadurch festgelegt, daß alle anderen Generalisierungen dieser Klauseln allgemeiner sind als H. Die Konstruktion einer speziellsten Generalisierung erlaubt somit das Steigen mit der geringsten Schrittweite in der durch die allgemeiner–als–Relation festgelegten Klauselhierarchie.

Außerdem besitzen speziellste Generalisierungen interressante theoretische Eigenschaften. Sie bilden die unteren Grenzen von Mitchells Versionenraum [Mitchell, 1982], so daß einige der dort vorgestellten Resultate auch für speziellste Generalisierungen gültig sind. Weiterhin ist die Konstruktion von speziellsten Generalisierungen die duale Operation zur Bildung von generellsten Unifikatoren (engl. *Most General Unifiers*).

Die Aufgabe, eine speziellste Generalisierung von Beispielen zu lernen, ist folgender Sonderfall des induktiven Lernproblems [Kietz, 1993a]:

Definition 2.19 (speziellste Generalisierung, MSG) *Gegeben sei ein ILP Problem* $(\vdash, \mathcal{LB}, \mathcal{LE}, \mathcal{LH})$*, wobei* $\vdash$ *eine allgemeiner–als–Relation ist.*

Gesucht ist eine Hypothese $H \subseteq \mathcal{LH}$*, so daß* $\forall h \in H$ *gilt: h ist Lösung des ILP Problems und*

$$\forall h' \in H : (h' \vdash h \Rightarrow h \vdash h').$$

Jedes $h \in H$ *heißt* **speziellste Generalisierung (Most Specific Generalization)** *der Beispiele bzgl. der Sprache* $\mathcal{LH}$*, dem Hintergrundwissen und der Relation* $\vdash$*.*

Die Existenz einer eindeutigen speziellsten Generalisierung ist abhängig vom Generalisierungsmodell $\vdash$. Sei $H_{\vdash}$ die Menge der Äquivalenzklassen bezüglich der von $\vdash$ induzierten Äquivalenzrelation $\sim_{\vdash}$.

Bei der logischen Implikation mit Hintergrundwissen ist die Existenz einer speziellsten Generalisierung nicht sicher ($H_{\models} = \emptyset$), ohne Hintergrundwissen gibt es nicht immer genau eine speziellste Generalisierung ($|H_{\models}| \geq 1$) von zwei Hornklauseln.

Plotkin wies nach, daß es für die θ–Subsumtion immer genau eine reduzierte speziellste Generalisierung gibt, die bis auf Variablenumbenennung eindeutig ist ($|H_{\vdash_\theta}| = 1$). H ist die leere Klausel, wenn keine Selektionen von E^+ existieren.

Definition 2.20 (Selektion) *Eine* **Selektion** *einer Menge von Klauseln* $S = \{C_i \mid 1 \leq i \leq n\}$ *ist die Menge der Literale* $K = \{L_i = \phi(t_{1_i}, \ldots, t_{m_i}) \mid 1 \leq i \leq n, L_i \in C_i\}$*.*

Gibt es keine speziellste Generalisierung der positiven Beispiele E^+ bzgl. $\vdash_\theta$, aus der sich zugleich keine negativen Beispiele E^- folgern lassen, so existiert überhaupt keine Generalisierung für die θ–Subsumtion.

Plotkin führt die Konstruktion einer speziellsten Generalisierungen zweier Klauseln bzgl. der θ–Subsumtion auf die speziellste Generalisierung von Wörtern zurück. Algorithmus 2.1 konstruiert speziellste Generalisierungen zweier Klauseln bzgl. der θ–Subsumtion nach Plotkin [Plotkin, 1970]. Plotkin wies auch nach, daß die speziellste Generalisierung einer Menge von Klauseln $\{C_1, \ldots, C_n\}$ auf diesen Fall zurückgeführt werden kann, da $msg(C_1, \ldots, C_n) = msg(\ldots msg(msg(C_1, C_2), C_3) \ldots, C_n)$ gilt.

Beispiel 2.7 *Seien*

e1: großmutter(ann,bob) $\leftarrow$ weiblich(ann) & vater(tom,bob)
& mutter(ann,tom) & vater(zak,tom) & tochter(sue,tom)

und

e2: großmutter(liz,john) $\leftarrow$ vater(ben,john) & mutter(liz,ben)
& mutter(liz,paul) & tochter(mary,ben).

Dann ist die speziellste Generalisierung von e1 und e2 bzgl. der θ–Subsumtion die Klausel

msg(C,D)
    konstruiere Liste $S := [[L_1^C - L_1^D], \ldots, [L_n^C - L_n^D]]$ der Selektionen
        von C und D, (d.h., $L_i^C \in C$ und $L_i^D \in D$ sind Literale
        mit gleichem Vorzeichen und Prädikatssymbol)
    transformiere Liste S in zwei Listen $M_C := [L_1^C, \ldots, L_n^C]$ und
        $M_D := [L_1^D, \ldots, L_n^D]$
    for $i := 1$ **to** n **do**
        repeat
            suche zwei Terme t_c und t_d in den Literalen L_i^C und L_i^D,
                so daß $t_c \neq t_d$ und sie entweder mit unterschiedlichen
                Funktionssymbolen beginnen oder mindestens einer
                der beiden Terme eine Variable ist
            ersetze t_c und t_d in M_C und M_D überall dort durch eine
                neue Variable X, wo sie in M_C und M_D an derselben
                Stelle auftreten
        until keine neuen Terme t_c und t_d mehr gefunden
    return M_C in Klauselnotation

Algorithmus 2.1: Konstruktion speziellster Generalisierungen nach Plotkin

msg(e1,e2): großmutter(X,Y) ← vater(Z,Y) & mutter(X,Z)
& mutter(X,U) & vater(R,S) & tochter(T,Z).

Als Reduktion ergibt sich

großmutter(X,Y) ← vater(Z,Y) & mutter(X,Z) & tochter(T,Z).

◇

Sowohl die Länge der speziellsten Generalisierungen, als auch ihre Berechnungszeit sind begrenzt durch die Anzahl der Selektionen von E^+. Sei m die Länge der längsten Klausel in E^+ und n die Anzahl der Klauseln von E^+, dann ist die Anzahl der Selektionen begrenzt durch m^n, also exponentiell in der Anzahl der Beispiele. Ein Teil der Literale in einer MSG wird durch Bildung der Reduktion wieder entfernt: Ein Literal L ist redundant in einer speziellsten Generalisierung C, wenn $C\theta \subseteq C \setminus L$.

Im letzten Abschnitt wurde bereits erwähnt, daß ij–determinierende MSGs nicht exponentiell wachsen, da sie in der Tiefe beschränkt sind. Ohne feste Werte für i und j können MSGs exponentiell länger werden. Kietz zeigt, daß die Länge einer determinierenden, reduzierten speziellsten Generalisierung von n Beispielen durch $\Omega(\frac{2^n}{p(n)})$ für eine polynomielle Funktion p begrenzt ist [Kietz, 1993a].

Die speziellste Generalisierung zweier Beispiele bezüglich der generalisierten Subsumtion $\vdash_B$ ist äquivalent zur MSG der saturierten Beispiele bezüglich der θ–Subsumtion $\vdash_\theta$ [Kietz, 1993a], [Jung, 1993, Folgerung 3]. Damit kann Plotkins Verfahren auch mit Hintergrundwissen benutzt werden.

Neben den Ansätzen zum Lernen von speziellsten Generalisierungen in prädikatenlogischen Formalismen existieren Verfahren zum Lernen von MSGs auf Basis von Graph Matching Verfahren [Haussler, 1989]. Während die logischen Verfahren die Literale abgleichen (Bildung von Selektionen) und die MSGs so viele Variablenbindungen enthalten, wie kompatibel mit den Selektionen, werden beim Graph Matching Ansatz die Objekte der Klauseln verglichen und entsprechend viele Relationen sind in den MSGs enthalten. Dadurch wird die Identität der einzelnen Objekte gewahrt.

Hausslers Graph Matching Ansatz wurde von Kietz in Plotkins Rahmen eingeordnet [Kietz, 1993a]. Bezüglich der θ–Subsumtion sind Hausslers MSGs zu generell beziehungsweise unvollständig. Sie lassen sich als MSGs bezüglich einer schwächeren allgemeiner–als–Relation definieren:

Definition 2.21 (θ–Subsumtion mit Objektidentität, $\vdash_{\theta_{inj}}$)
Eine Hornklausel C θ–subsumiert eine Hornklausel D genau dann unter Beachtung der Identität der Objekte, notiert als $C \vdash_{\theta_{inj}} D$, wenn $C \vdash_\theta D$ und θ eine injektive Abbildung ist.

Eine speziellste Generalisierung unter Beachtung der Objektidentität ist demnach eine MSG bezüglich der Relation $\vdash_{\theta_{inj}}$.

Beispiel 2.8 *Die speziellsten Generalisierungen bzgl. $\vdash_{\theta_{inj}}$ im obigen Beispiel sind:*

großmutter(X,Y) $\leftarrow$ vater(Z,Y) & mutter(X,Z) & tochter(T,Z)
großmutter(X,Y) $\leftarrow$ mutter(X,U)
großmutter(X,Y) $\leftarrow$ vater(R,S) $\diamond$

Im letzten Abschnitt wurde bereits erwähnt, daß Hausslers Problemstellung auf Plotkins Problem reduzierbar ist. Während es bei Plotkin genau eine speziellste Generalisierung gibt, die exponentiell in der Anzahl der Beispiele wachsen kann, ist die speziellste Generalisierung bei Haussler nicht eindeutig. Im schlimmsten Fall ist die Anzahl der MSGs exponentiell in der Anzahl der Objekte der Beispiele.

2.4 Zusammenfassung

Die erweiterte Ausdrucksstärke von Lernverfahren in prädikatenlogischen Formalismen führt zu negativen Resultaten bezüglich der Entscheidbarkeit und Komplexität der zugrundeliegenden Lernprobleme. Die Unentscheidbarkeit der logischen Folgerung wird durch korrekte, aber nicht vollständige Ableitungsregeln umgangen, die zugleich den Suchraum in einer Allgemeinheitsordnung strukturieren. Die θ–Subsumtion ist in Verbindung mit der Saturierung der Beispiele eine entscheidbare und für den Großteil der Hypothesensprachen vollständige Ableitungsregel, mit der auch Hintergrundwissen berücksichtigt werden kann. Das Lernen einer Begriffsbeschreibung unter Verwendung der θ–Subsumtion ist für tiefenbeschränkte, determinierende Klauseln in polynomieller Zeit möglich, Obermengen dieser Sprachen sind jedoch nicht effizient lernbar. Für die nicht monotone Semantik existieren in der Lerntheorie keine Resultate, die nicht monotonen Lernprobleme sind aber zum Teil einfacher. Speziellste Generalisierungen erlauben das bottom-up Steigen im Hypothesenraum, aber sowohl deren logische Vertreter (Plotkin) als auch die Graph Matching Ansätze (Haussler) sind exponentiell in der Anzahl der Beispiele.

Kapitel 3

Modellbasiertes Lernen in MOBAL

Zur Reduktion der Komplexität des induktiven Lernens in prädikatenlogischen Formalismen ist es notwendig, die Beschreibungssprachen für Hypothesen und Hintergrundwissen, sowie die Beispielsprache einzuschränken. Insbesondere muß der Raum aller möglichen Hypothesen kontrolliert und effizient durchsucht werden können.

Das Werkzeug RDT [Kietz und Wrobel, 1991] der Werkbank MOBAL [Morik *et al.*, 1993] ist ein *modellbasiertes* Lernverfahren mit einer speziellen Hypothesensprache, die eine vollständige Suche im Hypothesenraum in polynomieller Zeit ermöglicht.

Dieses Lernverfahren durchsucht, wie MIS [Shapiro, 1981] oder FOIL [Quinlan, 1990], den Suchraum von generelleren zu spezielleren Hypothesen. Anders als bei den zuletzt genannten Verfahren, benutzt RDT dabei keinen speziellen Verfeinerungsoperator, sondern legt die Menge der lernbaren Hypothesen durch *Modellwissen* in Form von syntaktischen Vorgaben an diese Hypothesen, sogenannten *Regelmodellen* oder *Regelschemata*, fest. Dies sind Formelschemata mit Prädikatsvariablen an Stelle von Prädikatssymbolen. Regelmodelle repräsentieren also Mengen syntaktisch gleicher Formeln.

Zu Beginn dieses Kapitels werden die Repräsentationssprache der Werkbank MOBAL und das Zusammenwirken ihrer einzelnen Werkzeuge beschrieben.

Abschnitt 3.2 diskutiert den Einsatz von Regelmodellen zum induktiven Lernen. Insbesondere wird eine neue Generalisierungsbeziehung zwischen Regelmodellen als Erweiterung der θ–Subsumtion von prädikatenlogischen Klauseln auf Mengen von Klauseln definiert. Das Lernverfahren RDT nutzt diese allgemeiner–als–Relation über Regelmodellen zur Strukturierung des Suchraumes. Um das Lernen redundanter Hypothesen zu verhindern, wird ein geeigneter Test bereitgestellt, der auf dieser Generalisierungsbeziehung beruht.

Die Semantik des Werkzeuges RDT wird in Abschnitt 3.3 definiert, da sie zur Formalisierung der eigentlichen Zielsetzung dieser Arbeit relevant ist.

Abschließend werden verwandte modellbasierte Ansätze zum induktiven Lernen skizziert (Abschnitt 3.4).

3.1 Die Werkbank Mobal

Das System Mobal [Morik *et al.*, 1993] ist eine modulare Werkbank mit einem eingeschränkten logischen Repräsentationsformalismus höherer Ordnung, die den Aufbau und die Wartung einer Wissensbasis unterstützt. Mobal verfolgt beim Modellierungsprozeß das Paradigma der ausbalancierten Kooperation (engl. *balanced cooperation*) [Morik, 1993] zwischen dem Benutzer und den Lernwerkzeugen des Systems. Alle im Repräsentationsformalismus von Mobal formulierbaren Ausdrücke können direkt vom Benutzer zum Modell des Sachbereiches hinzugefügt werden. Für jeden Bestandteil der Sprache existiert aber auch ein Werkzeug, daß entsprechene Ausdrücke auf Wunsch oder wann immer möglich, ausgehend vom aktuellen Modell, automatisch erzeugt.

3.1.1 Mobals Repräsentationsformalismus

Die Werkbank Mobal verwendet eine parakonsistente erweiterte Hornklausellogik ohne Funktionssymbole mit negierten Literalen im Körper der Hornklausel. Solche Formeln finden sich in der Literatur auch als generalisierte Hornklauseln (engl. *generalized Horn clauses*) [Blair und Subrahmanian, 1989]. In diesem Abschnitt werden die Bestandteile des Repräsentationsformalismus beschrieben, die dem Benutzer beim Aufbau und der Wartung einer Sachbereichstheorie über die Mensch–Maschine Schnittstelle (engl. *Human–Computer Interface*) zugänglich sind und für diese Arbeit relevant sind. Die Bedeutung der einzelnen Ausdrücke wird nur informell erläutert, eine formale Semantik des in Mobal verwendeten Formalismus wird in [Morik *et al.*, 1993, Kapitel 2.2] angegeben.

Fakten

Fakten beschreiben Relationen zwischen und Eigenschaften von Objekten, sowie die Zugehörigkeit zu einem Begriff. Ein positives Faktum ist ein funktions- und variablenfreies Literal der Form

$$\boxed{\mathsf{p}(\mathsf{t}_1,\ldots,\mathsf{t}_n)}$$

wobei p ein n–stelliges Prädikat und t_j ein konstanter Term sind. Mobals Semantik beruht auf einer vierwertigen Logik. Ein positives Faktum hat den Wahrheitswert wahr, negierte Fakten der Form $\mathsf{not}(\mathsf{p}(\mathsf{t}_1,\ldots,\mathsf{t}_n))$ haben den Wahrheitswert falsch. Beispiele für Fakten sind

mutter(ann,sue),
not(weiblich(tom)).

MOBAL kann zusätzlich widersprüchliche Fakten repräsentieren, diese haben den Wahrheitswert beides und werden als both(p($t_1, \ldots, t_n$)) notiert. Fehlende Informationen werden als unbekannt interpretiert, d.h. die Welt wird nicht als abgeschlossen betrachtet.

Prädikatsdeklarationen und Sorten

Den Argumenten eines Prädikates kann eine bestimmte Sorte zugewiesen werden. Diese erhalten in einer Prädikatsdeklaration einen symbolischen Namen:

p/n: <sorte_1>,...,<sorte_n>

Beziehungen zwischen Sorten, wie Inklusionen, können mit

sorte_1 :< sorte_2

spezifiziert werden.

Beispiele für Sorten sind

mutter/2: <frau>,<person>,
frau :< person.

Das Werkzeug STT verwaltet diese vom Benutzer definierten Sorten in einem dynamischen Sortenverband. Es kann außerdem ohne Angabe von Benutzersorten einen Sortenverband automatisch erzeugen[1]. Diese vom System definierten Sorten finden in dieser Arbeit aber keine Verwendung.

Einige ILP Lernverfahren setzen funktionale Abhängigkeiten (engl. *mode declarations*) zwischen den Argumenten eines Prädikates zur Reduzierung des Suchraumes ein. Argumente werden in *Ein*– und *Ausgabe*argumente unterteilt. Hypothesen, die an Eingabestellen eines Prädikates neue Variablen einführen, sind unsinnig. Eingabeargumente werden in MOBAL durch Kennzeichnung einer Sorte mit dem Symbol ! definiert. Nicht gekennzeichnete Argumente sind entweder Ausgabe– oder Eingabeargumente. Nach Deklaration von

alter/2: !<person>,<zahl>

muß die Variable X in dem Literal alter(X,Y) einer Hypothese bereits an anderer Stelle der Hypothese beschrieben sein, die Variable Y kann dagegen entweder alt oder neu sein.

[1] Das System STT wird in [Morik *et al.*, 1993, Kapitel 4] beschrieben.

Regeln

Regeln in MOBAL sind Implikationen und enthalten neben einem ausgezeichneten Kopf mindestens ein weiteres Literal. Als generalisierte Hornklauseln können sie beliebig viele negative und positive Literale enthalten:

$$\mathsf{Kopf} \leftarrow \mathsf{L}_1 \ \& \ldots \ \& \ \mathsf{L}_n$$

Ein Beispiel ist die Regel

mutter(X,Y) ← tochter(Y,X) & not(männlich(X)).

Regeln können in der Inferenzmaschine eingebaute Prädikate verwenden. Diese unterteilen sich in arithmetische Funktionen und *autoepistemische Operatoren*, deren Auswertung vom augenblicklichen Zustand der Wissensbasis abhängt.

Prädikatstopologie

Die Prädikatstopologie beschreibt semantische Beziehungen zwischen den Prädikaten der Sachbereichstheorie. Ein Topologieknoten besteht aus einer Menge von Prädikaten, die in einer semantischen Beziehung stehen, und der Angabe seiner Kindesknoten in der Topologie.

geburtstag – [ort_der_feier,anzahl_der_gäste,musikrichtung]
<– [verwandtschaftsbeziehungen,freundschaften,persönliches]

Der Topologieknoten geburtstag enthält die Prädikate ort_der_feier, anzahl_der_gäste und musikrichtung, die Eigenschaften einer Geburtstagsfeier beschreiben. Zu diesem Knoten gibt es Kanten von den Knoten verwandtschaftsbeziehungen, freundschaften und persönliches. Zwischen allen Prädikaten in diesen Knoten und den Prädikaten des Knotens geburtstag werden inferentielle Beziehungen in Betracht gezogen. Für ein Lernverfahren bedeutet dies, daß nur solche Prädikate im Körper einer Hypothese auftreten dürfen, die sich im Topologieknoten des Kopfes der Hypothese befinden oder in einem seiner Kindesknoten. Die allgemeine Definition eines Topologieknotens ist:

$$\mathsf{Knoten} - [\mathsf{P}_1,\ldots,\mathsf{P}_n] <\!- [\mathsf{T}_1,\ldots,\mathsf{T}_r]$$

wobei die P_i Prädikate und die T_j Kindesknoten des Topologieknotens Knoten sind.

Metaprädikate

Metaprädikate sind die Bestandteile der Sprache, die zur Repräsentation der Regelmodelle dienen. Sie bestehen aus einem Metaprädikatskopf und dem eigentlichen Regelschema, welches voll instanziiert zu einer Regel der Sprache wird:

$$\boxed{\mathsf{m}(\mathsf{P}_1,\ldots,\mathsf{P}_n,\mathsf{C}_1,\ldots,\mathsf{C}_r)\text{: } \mathsf{P}_1(\vec{X}_1),\ldots,\mathsf{P}_n(\vec{X}_n) \text{ -> } \mathsf{P}_{n+1}(\vec{X}_{n+1})}$$

Dabei sind m der Name des Metaprädikates und die $\vec{X}_i$ sind die Argumentsvektoren der Literale. Im Kopf des Metaprädikates werden außer den Prädikatsvariablen $\mathsf{P}_1,\ldots,\mathsf{P}_n$ noch diejenigen Argumente des Regelmodells notiert, die in einer Regel als Instanz dieses Modells durch einen konstanten Term ersetzt werden müssen ($\mathsf{C}_1,\ldots,\mathsf{C}_r$). Eine Instanz des Metaprädikates

m1(P,Q,C): P(C,X) → Q(X,C)

ist beispielsweise die Regel

mutter(ann,X) → kind(X,ann).

MOBALs Formalismus befähigt darüber hinaus zur Überwachung von *Integritätsbedingungen* [Morik *et al.*, 1993, Kapitel 2.1.5] und zur Definition von *Metaregeln, Metametaprädikaten* und *Metametafakten* [Morik *et al.*, 1993, Kapitel 2.1.8]. Diese Bestandteile der Sprache sind für die Problemstellung dieser Arbeit aber nicht relevant und werden daher nicht dargestellt. Die Meta-Formalismen der Sprache können zudem beim Beschreiben eines Sachbereichmodells außer acht gelassen werden, da sie falls notwendig automatisch generiert werden.

3.1.2 MOBALs Werkzeuge

Die Eingabe von Fakten durch eine Benutzer ist Voraussetzung für den Aufbau einer Sachbereichstheorie. Die Inferenzmaschine IM–2 [Morik *et al.*, 1993, Kapitel 3] des Systems benutzt eingegebene oder von einer der beiden induktiven Komponenten RDT und CLT[2] (engl. *concept learning tool*) [Morik *et al.*, 1993, Kapitel 8] entdeckte Regeln zur Inferenz weiterer Fakten. Zugleich verwaltet sie alle anderen Bestandteile der Repräsentationssprache und die Zurücknahme von Teilen des operationalen Modells durch den Benutzer oder das Wissensrevisionswerkzeug KRT (engl. *knowledge revision tool*) [Morik *et al.*, 1993, Kapitel 7].

Der dynamische Sortenverband wird vom Werkzeug STT (engl. *sort taxonomy tool*) [Morik *et al.*, 1993, Kapitel 4] verwaltet. Neben den vom Benutzer definierten Sorten können mit diesem Werkzeug automatisch Argumentssorten generiert werden, die das Vorkommen der Objekte in den Fakten der Sachbereichstheorie wiederspiegeln.

Die Prädikatstopologie und deren Einsatz beim Lernen überwacht das Werkzeug PST (engl. *predicate structuring tool*) [Morik *et al.*, 1993, Kapitel 5]. Analog zum Sortenverband besteht auch hier die Möglichkeit, neben der vom

[2] CLT ist das Werkzeug des Systems MOBAL zum Lernen einer Begriffsbeschreibung.

Benutzer definierten Topologie, eine Systemtopologie zu erstellen, die dann die Beziehungen der Prädikate bezüglich der Regelmenge der momentanen Sachbereichstheorie reflektiert.

3.2 Modellwissen in Form von Regelmodellen

Regelmodelle (oder Regelschemata[3]) mit Prädikatsvariablen stellen eine Erweiterung der Hornklausellogik dar. Sie erlauben es, Prädikate als Datenobjekte zu betrachten. Warren diskutiert entsprechende Erweiterungen der Programmiersprache PROLOG [Warren, 1982]. In Analogie zu der Fähigkeit funktionaler Sprachen (wie LISP) Funktionen als Datenobjekte zu behandeln, schildert er, daß in Prologausdrücken Prädikatssymbole als Variablen betrachtet und weiterhin *Lambda*–Ausdrücke zum anonymen Zugriff auf solche Objekte eingeführt werden können.

Warren zeigt, daß diese Erweiterungen die Ausdrucksstärke von Prolog nicht vergrößert, indem er eine Abbildung auf die Prädikatenlogik erster Stufe angibt. Diese Abbildung hat keinen unbegrenzten Anstieg der Größe oder der Anzahl der Berechnungen in Prolog Programmen zur Folge. Auch wären von der Einführung von Prädikatsvariablen keine Effizienzsteigerungen oder eine bessere Lesbarkeit von Prolog Programmen zu erwarten.

Im Bereich des maschinellen Lernens dagegen wurden Prädikatsvariablen durch die Werkbank MOBAL und seine Vorläufer (METAXA [Emde *et al.*, 1983]) in die Literatur eingeführt und ihre Eignung zur Reduzierung und Strukturierung des Suchraumes beim Lernen in eingeschränkter Prädikatenlogik nachgewiesen [Morik *et al.*, 1993; Sommer *et al.*, 1994].

Die folgenden Definitionen sind angelehnt an die bei der Prädikatenlogik erster Stufe üblichen Definitionen von Prädikaten, Atomen, Literalen und Regeln [Lloyd, 1987] und entsprechen im wesentlichen den Bezeichnungen von Kietz und Wrobel [Kietz und Wrobel, 1991][4].

Definition 3.1 (Prädikatsvariable) *Eine* **Prädikatsvariable** *ist eine Variable, die nur mit einem Prädikatssymbol bestimmter Stelligkeit instanziiert werden kann. Als Bezeichner für Prädikatsvariablen werden P, Q oder andere Großbuchstaben verwendet.*

Außer Prädikatsvariablen enthalten Regelmodelle noch Ausdrücke der Form $\ddot{C}_i$. Dies sind Termvariablen die nur durch Konstanten instanziiert oder mit anderen Ausdrücken $\ddot{C}_j$ unifiziert werden dürfen. Sie sind Platzhalter für beliebige Konstanten, die an ihrer Stelle in der zu lernenden Regel auftreten. Formal werden sie als Terme der Repräsentationssprache eingeführt und die Substitution auf Termen entsprechend erweitert.

[3] In dieser Arbeit werden die Begriffe Regelmodell und Regelschema gleichwertig verwendet.

[4] Abweichend von [Kietz und Wrobel, 1991] wird hier der Begriff des Literalschemas eingeführt, und der Begriff des Regelmodelles umfaßt auch Hornklauseln.

Definition 3.2 (Konstantenvariablen, $\ddot{C}$) *Seien c eine beliebige Konstante und V und C beliebige Variablen. Dann sind c, V und $\ddot{C}$ Terme. Terme der Form $\ddot{C}$ heißen* **Konstantenvariablen**. *Eine* **erweiterte Substitution** $\ddot{\sigma}$ *ist eine Abbildung auf der Menge der Variablen, die nur endlich viele Variablen durch Variablen, Konstantenvariablen oder Konstanten ersetzt.*

Konstantenvariablen sind somit als Erweiterung der Terme der Prädikatenlogik erster Stufe definiert. Sie wurden im letzten Abschnitt bereits als Bestandteile des Metaprädikatkopfes eingeführt. Es sind die Argumente des Kopfes, die nicht als Prädikatsvariable im Regelmodell vorkommen und beim Lernen durch konstante Terme zu ersetzen sind. Die Termsubstitution σ über Variablen und Konstanten bleibt unverändert, eine erweiterte Substitution $\ddot{\sigma}$ bildet zusätzlich Variablen auf Konstantenvariablen ab. Beispielsweise gilt

$$\mathsf{mutter(X,Y)}\{\mathsf{Y}/\ddot{\mathsf{C}}\} = \mathsf{mutter(X,}\ddot{\mathsf{C}}\mathsf{)}.$$

Definition 3.3 (Literalschema) *Seien $t_1, \ldots, t_n$ Terme, p ein Prädikatssymbol und P eine Prädikatsvariable, so sind die Formeln $p(t_1, \ldots, t_n)$ und $P(t_1, \ldots, t_n)$* **atomare Schemata**. *Ein* **Literalschema** *ist ein atomares Schema oder dessen Negation. Ein Literalschema heißt* **prädikatsvariablenfrei** *oder* **Literal**, *wenn es keine Prädikatsvariablen und keine Konstantenvariablen enthält.*

Definition 3.4 (Regelschema, Regelmodell)
Ein **Regelschema** *oder* **Regelmodell** *besteht aus genau einem Kopfliteralschema und endlich vielen Literalschemata im Körper des Regelmodells. Regelschemata werden als Implikationen*

$$C \leftarrow L_1, \ldots, L_n \text{ , } \textit{wobei } C, L_1, \ldots, L_n \textit{ Literalschemata und } n \geq 0,$$

notiert. Übliche Bezeichner für Regelschemata sind die Großbuchstaben R oder RS. Prädikatsvariablenfreie Regelschemata entsprechen den Regeln in MOBAL.

Die Anwendung einer Termsubstitution auf Regelmodelle verändert keine Prädikatsvariablen. Da insbesondere die θ–Subsumtion nur für Termsubstitutionen definiert ist, stehen Regelmodelle mit unterschiedlichen Prädikatsvariablen bzgl. $\vdash_\theta$ nicht in Relation[5]. Beispielsweise sind

$$\mathsf{Q(X)} \leftarrow \mathsf{P1(X,Y)} \text{ und}$$
$$\mathsf{Q(X)} \leftarrow \mathsf{P2(X,Y)}$$

bzgl. $\vdash_\theta$ nicht äquivalent.

[5]Analog zur Klauselnotation läßt sich ein Regelmodell auch als Menge von Literalschemata schreiben. Da aber beliebig viele negative und positive Literalschemata auftreten können, ist der Kopf C der Implikation in der Mengennotation besonders zu kennzeichnen, etwa kopf(C).

Kietz und Wrobel führen analog zur Termsubstitution eine Substitution Σ angewandt auf Prädikatsvariablen ein [Kietz und Wrobel, 1991]. Diese soll Prädikatsvariablen umbenennen oder instanziieren[6]. Sie wird hier erweitert um die Substitution von Konstantenvariablen und die Anwendung auf Regelmodelle:

Definition 3.5 (Schemasubstitution Σ) *Eine* **Schemasubstitution** Σ *ist eine Abbildung auf der Menge der Prädikats– und Konstantenvariablen, die nur endlich viele dieser Terme verändert. Seien $X_1, \ldots, X_n$ Prädikatsvariablen der Stelligkeit $a_1, \ldots, a_n$ und $P_1, \ldots, P_n$ Prädikatsvariablen oder Prädikatssymbole der Stelligkeit $a_1, \ldots, a_n$. Seien $\ddot{C}_1, \ldots, \ddot{C}_r$ Konstantenvariablen und $D_1, \ldots, D_r$ Konstanten oder Konstantenvariablen. Zur Notation wird eine endliche Menge von Paaren $\Sigma = \{X_1/P_1, \ldots, X_n/P_n, \ddot{C}_1/D_1, \ldots, \ddot{C}_r/D_r\}$ verwendet. Mithilfe der rekursiven Definition*

$$\Sigma(F) = \left\{ \begin{array}{lll} D_j & ,\ \mathit{falls} & F = \ddot{C}_j \wedge\ C_j/D_j \in \Sigma \\ P_i(\Sigma(t_1), \ldots, \Sigma(t_n)) & ,\ \mathit{falls} & F = X_i(t_1, \ldots, t_n) \\ & & \wedge\ X_i/P_i \in \Sigma \\ \Sigma(\mathsf{L}_1), \ldots, \Sigma(\mathsf{L}_n) \rightarrow \Sigma(\mathsf{C}) & ,\ \mathit{falls} & F = \mathsf{L}_1, \ldots, \mathsf{L}_n \rightarrow \mathsf{C} \\ F & ,\ \mathit{sonst} & \end{array} \right\}$$

wird diese Abbildung zu einem Homomorphismus über Literalschemata und Regelmodellen erweitert.

Das Ergebnis des gleichzeitigen Einsetzens der $P_1, \ldots, P_n$ für die Prädikatsvariablen $X_1, \ldots, X_n$ und der $D_1, \ldots, D_r$ für die Konstantenvariablen $\ddot{C}_1, \ldots, \ddot{C}_r$ heißt **Instanz** *$R\Sigma$ von R. Die Menge $\{X_1, \ldots, X_n, \ddot{C}_1, \ldots, \ddot{C}_r\}$ heißt* **Bereich**, *die Menge $\{P_1, \ldots, P_n, D_1, \ldots, D_r\}$* **Bild** *von Σ.*

Zur Veranschaulichung dieser Definition dient ein kurzes Beispiel. So wird das Regelmodell

$$\mathsf{Q(X)} \leftarrow \mathsf{P(X,\ddot{C})}$$

durch die Schemasubstitution $\Sigma = \{\mathsf{P/mutter,Q/weiblich,\ddot{C}/mary}\}$ auf die Regel

$$\mathsf{weiblich(X)} \leftarrow \mathsf{mutter(X,mary)}$$

abgebildet.

3.2.1 Ordnungsrelation über Regelmodellen

Das modellbasierte Lernverfahren RDT benutzt Regelmodelle zur Reduktion des Hypothesensuchraumes. Unter Vorgabe einer Menge $\mathcal{R}$ von Regelmodellen und Mengen $\mathcal{P}$ von Prädikaten und $\mathcal{C}$ von Objekten des Sachbereichs reduziert sich dieser Raum auf die Menge aller Grundinstanzen von Regelschemata aus $\mathcal{R}$:

[6] Im Gegensatz zu [Kietz und Wrobel, 1991] wird hier keine Unterscheidung zwischen der Instanziierung und der Substitution von Regelschemata vorgenommen.

Bezeichnung 3.1 ($\mathcal{LH}_{\mathcal{R}}$) *Der Hypothesenraum einer Menge von Regelmodellen $\mathcal{R}$ und Mengen $\mathcal{P}$ von Prädikaten und $\mathcal{C}$ von Objekten des Sachbereichs wird als Menge*

$$\mathcal{LH}_{\mathcal{R}} := \{R\Sigma \mid R \in \mathcal{R} \wedge \mathit{Bild}(\Sigma) \subseteq \mathcal{P} \cup \mathcal{C} \wedge R\Sigma \textit{ ist eine Regel}\}$$

bezeichnet.

Durch zusätliche Auflagen an die Schemasubstitution Σ sind weitere Einschränkungen dieses Suchraumes möglich. Diese haben außerdem Auswirkungen auf eine Generalisierungsrelation über den Regelmodellen, die das modellbasierte Lernverfahren RDT zu Strukturierung des Suchraumes einsetzt.

Verbietet man die Unifikation von verschiedenen Prädikatsvariablen, wird die Substitution Σ zu einer injektiven Abbildung.

Bezeichnung 3.2 (Σ^{inj}) *Eine Schemasubstitution Σ, die Prädikatsvariablen injektiv abbildet, wird mit der Hochstellung Σ^{inj} gekennzeichnet.*

Mit dieser Einschränkung ergibt sich folgender Suchraum:

Bezeichnung 3.3 ($\mathcal{LH}_{\mathcal{R}}^{inj}$) *Sei $\mathcal{P}$ die Menge der Prädikate und $\mathcal{C}$ die Menge der Objekte des Sachbereiches. Der Hypothesenraum einer Menge von Regelmodellen $\mathcal{R}$ mit der Restriktion auf Instanzen von injektiven Substitutionen wird als Menge*

$$\mathcal{LH}_{\mathcal{R}}^{inj} := \{R\Sigma^{inj} \mid R \in \mathcal{R} \wedge \mathit{Bild}(\Sigma^{inj}) \subseteq \mathcal{P} \cup \mathcal{C} \wedge R\Sigma^{inj} \textit{ ist eine Regel}\}$$

bezeichnet.

Nicht so stark ist die Restriktion, nur solche Schemasubstitutionen nicht zu erlauben, die bzgl. θ nicht redundante Literalschemata auf redundante Literalschemata abbilden:

Bezeichnung 3.4 ($\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$) *Sei $\mathcal{P}$ die Menge der Prädikate und $\mathcal{C}$ die Menge der Objekte des Sachbereiches. Der Hypothesenraum einer Menge von Regelmodellen $\mathcal{R}$ mit der Restriktion auf Instanzen von Schemasubstitutionen, die keine bzgl. θ nicht redundante Literalschemata auf redundante Literalschemata abbilden, wird als Menge*

$$\begin{aligned}\mathcal{LH}_{\mathcal{R}}^{lit\_inj} := \{R\Sigma \mid\ & R \in \mathcal{R} \ \wedge \mathit{Bild}(\Sigma) \subseteq \mathcal{P} \cup \mathcal{C} \ \wedge \ R\Sigma \textit{ ist eine Regel} \\ & \wedge \forall L \in R, \theta : (R\theta = R \setminus \{L\} \vee R\Sigma\theta \neq R \setminus \{L\}\Sigma\theta)\}\end{aligned}$$

bezeichnet.

Die Unterschiede zwischen diesen Suchräumen verdeutlicht ein kurzes Beispiel.

Beispiel 3.1 *Die Menge der Regelmodelle sei*

$$\mathcal{R} = \left\{ \begin{array}{lll} \mathsf{Q(X,Z)} & \leftarrow & \mathsf{P1(X,Y)} \ \& \ \mathsf{P2(Y,Z)}, \\ \mathsf{Q(X,Y)} & \leftarrow & \mathsf{P1(X,Y)} \ \& \ \mathsf{P2(X,Z)} \end{array} \right\}$$

und die gegebenen Prädikatssymbole seien $\mathcal{P} = \{\mathsf{kleiner/2}\}$.
Dann gelten

$$\mathcal{LH}_{\mathcal{R}} = \left\{ \begin{array}{lll} \mathsf{kleiner(X,Z)} & \leftarrow & \mathsf{kleiner(X,Y)} \ \& \ \mathsf{kleiner(Y,Z)}, \\ \mathsf{kleiner(X,Y)} & \leftarrow & \mathsf{kleiner(X,Y)} \ \& \ \mathsf{kleiner(X,Z)} \end{array} \right\}$$

und ohne nicht redundante Literalschemata auf redundante abzubilden

$$\mathcal{LH}_{\mathcal{R}}^{lit\_inj} = \{\mathsf{kleiner(X,Z)} \leftarrow \mathsf{kleiner(X,Y)} \ \& \ \mathsf{kleiner(Y,Z)}\}$$

während sich dieser Raum, durch das Verbot Prädikatsvariablen zu unifizieren, auf

$$\mathcal{LH}_{\mathcal{R}}^{inj} = \{\}$$

reduziert. ◇

Zur Strukturierung dieser Suchräume dient eine allgemeiner–als–Relation über Regelmodellen. Das Lernverfahren RDT sucht die bezüglich der θ–Subsumtion allgemeinsten Regeln innerhalb seines Suchraumes, die eine Lösung gemäß der Semantik dieses Werkzeuges darstellen. Die Generalisierung über Regelmodellen muß daher der θ–Subsumtion von Klauseln entsprechen. Sie ist somit als θ–Subsumtion von Klauselmengen zu definieren.

Ein Generalisierungsbegriff $\vdash_{RS}$ über Regelmodellen sollte folgende Anforderungen erfüllen:

1. Gilt R genereller als R', dann gibt es zu jeder Grundinstanz C' von R' eine Regel C als Instanz von R mit $C \vdash_\theta C'$. Dadurch wird einem modellbasierten Lernverfahren ein Abschneiden von Teilen des Suchraumes ermöglicht.

2. Für Grundinstanzen C, C' von Regelmodellen, also Regeln, gilt: $C \vdash_{RS} C' \Leftrightarrow C \vdash_\theta C'$.

3. Analog zur θ–Subsumtion soll eine Schemasubstitution oder das Hinzufügen eines Literalschemas zu einer Spezialisierung bezüglich $\vdash_{RS}$ führen.

In der Literatur zum automatischen Beweisen findet sich folgender Subsumtionsbegriff über Mengen von Hornklauseln [Hofbauer und Kutsche, 1989]:

Definition 3.6 (θ–Subsumtion auf Klauselmengen)
*Eine Klauselmenge X θ–***subsumiert eine Klauselmenge** *Y genau dann, wenn für alle Klauseln $D \in Y$ eine Klausel $C \in X$ existiert, so daß Klausel C die Klausel D θ–subsumiert.*

Übertragen auf Regelmodelle als Repräsentanten solcher Mengen von Klauseln gilt es, die Grundinstanzen von Regelmodellen zu betrachten. Daher sind für die drei vorgestellten Suchräume unterschiedliche allgemeiner–als–Relationen zu definieren.

Definition 3.7 (allgemeiner–als–Relation $\vdash_{RS}$ bzgl. $\mathcal{LH}_{\mathcal{R}}$)
Ein **Regelmodell** *R heißt genau dann* **genereller als** *ein Regelmodell R' im Suchraum $\mathcal{LH}_{\{R\}}$, wenn gilt: Für alle Regeln $D \in \mathcal{LH}_{\{R'\}}$ gibt es eine Regel $C \in \mathcal{LH}_{\{R\}}$ mit $C \vdash_\theta D$. Die Schreibweise ist $R \vdash_{RS} R'$.*

Ist bei der Instanziierung der Regelmodelle das Abbilden von bzgl. $\vdash_\theta$ nicht redundanten Literalschemata auf äquivalente Schemata nicht erlaubt, so ist folgende Definition sinnvoll:

Definition 3.8 (allgemeiner–als–Relation $\vdash_{RS}^{lit\_inj}$ bzgl. $\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$)
Ein **Regelmodell** *R heißt genau dann* **genereller als** *ein Regelmodell R' im Suchraum $\mathcal{LH}_{\{R\}}^{lit\_inj}$, wenn gilt: Für alle Regeln $D \in \mathcal{LH}_{\{R'\}}^{lit\_inj}$ gibt es eine Regel $C \in \mathcal{LH}_{\{R\}}^{lit\_inj}$ mit $C \vdash_\theta D$. Die Schreibweise ist $R \vdash_{RS}^{lit\_inj} R'$.*

Ist gar die Unifikation von Prädikatsvariablen verboten, wird die Relation noch schwächer:

Definition 3.9 (allgemeiner–als–Relation $\vdash_{RS}^{inj}$ bzgl. $\mathcal{LH}_{\mathcal{R}}^{inj}$)
Ein **Regelmodell** *R heißt genau dann* **genereller als** *ein Regelmodell R' im Suchraum $\mathcal{LH}_{\{R\}}^{inj}$, wenn gilt: Für alle Regeln $D \in \mathcal{LH}_{\{R'\}}^{inj}$ gibt es eine Regel $C \in \mathcal{LH}_{\{R\}}^{inj}$ mit $C \vdash_\theta D$. Die Schreibweise ist $R \vdash_{RS}^{inj} R'$.*

Es gilt: R $\vdash_{RS}^{inj} R' \quad \Rightarrow \quad R \vdash_{RS}^{lit\_inj} R' \quad \Rightarrow \quad R \vdash_{RS} R'$.

Alle drei Definitionen sind eine direkte Umsetzung der ersten Forderung für die einzelnen Suchräume und erfüllen auch Punkt zwei.

Die letzte Forderung ist wichtig für eine Operationalisierung dieser Definitionen. Das Hinzufügen von Literalschemata führt bei allen Definitionen zu einer Spezialisierung. Dagegen kann es durch eine Schemasubstitution Σ von Prädikatsvariablen im Gegensatz zu einer Substitution σ von Termvariablen durch Unifikation verschiedener Literalschemata zu einer Generalisierung eines Regelmodells bezüglich der Relation $\vdash_{RS}$ aus Definition 3.7 kommen (d.h. $\forall\sigma : R \vdash_{RS} R\sigma$, aber $\exists\Sigma : R\Sigma \vdash_{RS} R$).

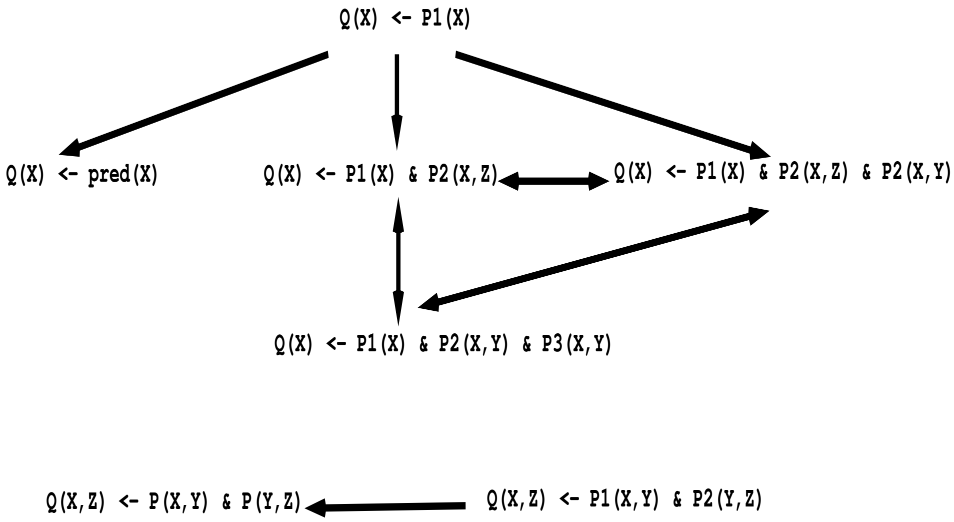

Abbildung 3.1: Generalisierungshierachie $\vdash_{RS}$ von Regelmodellen

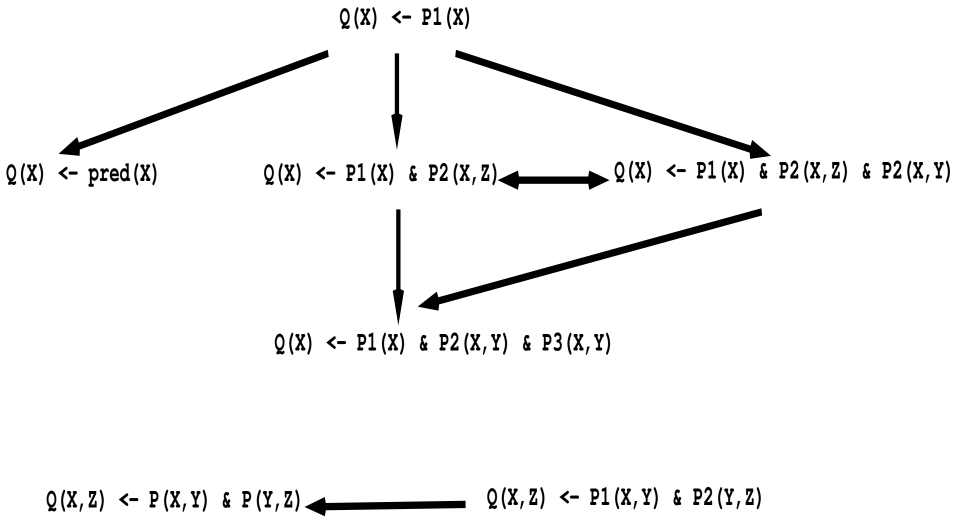

Abbildung 3.2: Generalisierungshierachie $\vdash_{RS}^{lit\_inj}$ von Regelmodellen ohne Unifikation nicht redundanter Literalschemata

Beispiel 3.2 *Sei* $R = \mathsf{P}_1(\mathsf{X})\ \&\ \mathsf{P}_2(\mathsf{X},\mathsf{Y})\ \&\ \mathsf{P}_3(\mathsf{X},\mathsf{Y}) \rightarrow \mathsf{Q}(\mathsf{X})$.
Regelmodell R *ist zwar genereller als* $R\sigma$ *mit* $\sigma = \{\mathsf{Y}/\mathsf{X}\}$*:*

$$\mathsf{P}_1(\mathsf{X})\ \&\ \mathsf{P}_2(\mathsf{X},\mathsf{Y})\ \&\ \mathsf{P}_3(\mathsf{X},\mathsf{Y}) \rightarrow \mathsf{Q}(\mathsf{X})$$
$$\vdash_{RS}$$
$$\mathsf{P}_1(\mathsf{X})\ \&\ \mathsf{P}_2(\mathsf{X},\mathsf{X})\ \&\ \mathsf{P}_3(\mathsf{X},\mathsf{X}) \rightarrow \mathsf{Q}(\mathsf{X});$$

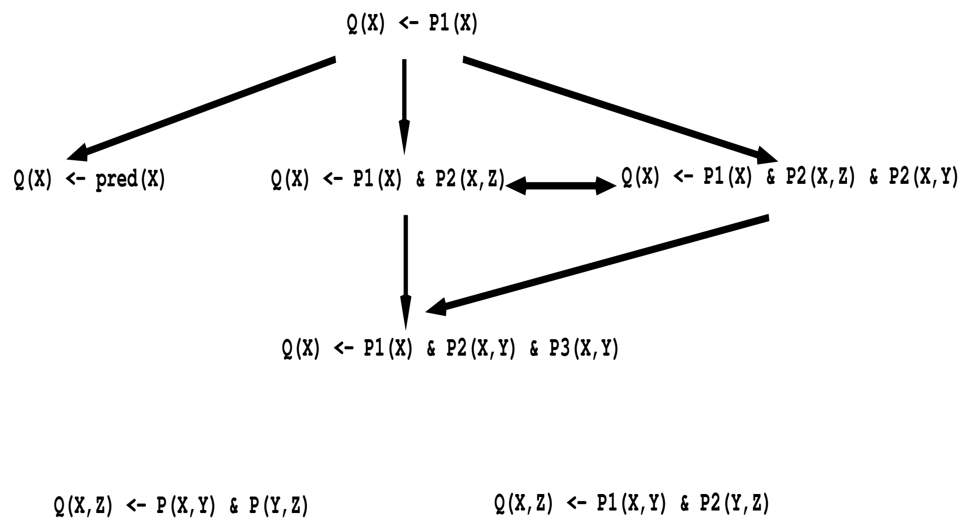

Abbildung 3.3: Generalisierungshierachie $\vdash_{RS}^{inj}$ von Regelmodellen mit injektiver Substitution von Prädikatsvariablen

es gilt aber ebenso $R\Sigma \vdash_{RS} R$ *mit* $\Sigma = \{\mathsf{P}_3/\mathsf{P}_2\}$ *:*

$$\mathsf{P}_1(\mathsf{X})\ \&\ \mathsf{P}_2(\mathsf{X},\mathsf{Y}) \rightarrow \mathsf{Q}(\mathsf{X})$$
$$\vdash_{RS}$$
$$\mathsf{P}_1(\mathsf{X})\ \&\ \mathsf{P}_2(\mathsf{X},\mathsf{Y})\ \&\ \mathsf{P}_3(\mathsf{X},\mathsf{Y}) \rightarrow \mathsf{Q}(\mathsf{X}).$$

◇

Aufgrund dieser unerwünschten Eigenschaft ist die Definition 3.7 einer allgemeiner–als–Relation zwischen Regelmodellen schlecht zu operationalisieren. Solche Schemasubstitutionen Σ verbieten aber gerade die Suchräume $\mathcal{LH}_{\{R\}}^{inj}$ und $\mathcal{LH}_{\{R\}}^{lit\_inj}$. Daher kommen nur noch diese Suchräume und die korrespondierenden allgemeiner–als–Relationen für ein modellbasiertes Lernverfahren in Frage.

Der Suchraum $\mathcal{LH}^{inj}$, der die Unifikation von Prädikatsvariablen verbietet, ist aber zu restriktiv. Daß die Hornklausel

$$\mathsf{kleiner(X,Z)} \leftarrow \mathsf{kleiner(X,Y)}\ \&\ \mathsf{kleiner(Y,Z)}$$

nicht im Suchraum des Regelmodells

$$\mathsf{Q(X,Z)} \leftarrow \mathsf{P1(X,Y)}\ \&\ \mathsf{P2(Y,Z)}$$

liegt, entspricht nicht der Intuition. Ebenso ist nicht einzusehen, daß die Regelmodelle

$$\mathsf{Q(X,Z)} \leftarrow \mathsf{P(X,Y)}\ \&\ \mathsf{P(Y,Z)}$$

und

$$\mathsf{Q(X,Z)} \leftarrow \mathsf{P1(X,Y)}\ \&\ \mathsf{P2(Y,Z)}$$

in Abbildung 3.3 nicht in Relation zueinander stehen, da im Suchraum $\mathcal{LH}^{lit\_inj}$ sogar alle Grundinstanzen des ersten Regelmodelles auch im zweiten Modell enthalten sind.

Daher erscheint nur der Suchraum $\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ sinnvoll, zusammen mit der Relation $\vdash_{RS}^{lit\_inj}$ auf der Menge der Regelmodelle $\mathcal{R}$.

Die Abbildungen 3.1, 3.2 und 3.3 verdeutlichen noch einmal die Unterschiede der so definierten Generalisierungsrelationen.

Eine Operationalisierung der Relation $\vdash_{RS}^{lit\_inj}$ kann jetzt mit Hilfe einer Schemasubstitution vorgenommen werden, die keine nicht redundanten Literalschemata unifizierbar macht.

Satz 3.1 *Seien R und R' Regelmodelle und Σ eine Schemasubstitution. Seien R_{red} und R'_{red} die Reduktionen von R und R' bezüglich der θ–Subsumtion. Dann gilt $R \vdash_{RS}^{lit\_inj} R'$ genau dann, wenn eine erweiterte Substitution $\ddot{\sigma}$ für Termvariablen existiert, so daß $R_{red}\Sigma\ddot{\sigma} \subseteq R'_{red}$ gilt und Σ eine injektive Abbildung über der Menge der Literalschemata ist.*

Beweis

" $\Rightarrow$ "

Nach Definition 3.8 gilt: $\forall$ Regeln $R'\Sigma'$: $\exists$ Regel $R\Sigma$ mit $R\Sigma \vdash_\theta R'\Sigma'$ und somit auch für die reduzierten Regelmodelle: $R_{red}\Sigma \vdash_\theta R'_{red}\Sigma'$,

$\Rightarrow$ Σ und Σ' sind über den reduzierten Regelmodellen injektive Abbildungen von Literalschemata auf Literale und $\exists R'_{sub} \subseteq R'_{red} \wedge \Sigma'_{sub} \subseteq \Sigma'$ mit: $R_{red}\Sigma\theta = R'_{sub}\Sigma'_{sub}$,

$\Rightarrow$ $R_{red}\Sigma\theta\Sigma'^{-1}_{sub} = R'_{sub} \subseteq R'_{red}$, wobei Σ'^{-1}_{sub} als inverse Abbildung von Σ'_{sub} eine injektive Abbildung von Literalen auf Literalschemata ist, d.h. $\Sigma\Sigma'^{-1}_{sub}$ ist eine Abbildung, die Literalschemata injektiv auf Literalschemata abbildet.

" $\Leftarrow$ "

$R_{red}\Sigma\ddot{\sigma} \subseteq R'_{red}$

$\Rightarrow$ $R_{red}\Sigma\ddot{\sigma}\Sigma_{ground} \subseteq R'_{red}\Sigma_{ground}$ und Σ_{ground} ist eine beliebige Schemasubstitution, die keine Literalschemata unifiziert,
so daß $R_{red}\Sigma\ddot{\sigma}\Sigma_{ground}$ und $R'_{red}\Sigma_{ground}$ Regeln sind,

$\Rightarrow$ $R_{red} \vdash_{RS}^{lit\_inj} R'_{red}$ und somit auch $R \vdash_{RS}^{lit\_inj} R'$ □

Die Relation $\vdash_{RS}^{lit\_inj}$ entspricht bei prädikatsvariablenfreien Regelmodellen, also Regeln, der θ–Subsumtion. Somit ist die θ–Subsumtion ein Spezialfall dieser Relation und $\vdash_{RS}^{lit\_inj}$ ist folglich nicht in polynomieller Zeit berechenbar.

Folgerung 3.1 *Der Test* $\vdash_{RS}^{lit\_inj}$ *ist NP–schwierig.*

In Kapitel 4.3.5 wird ein Algorithmus für den Test $\vdash_{RS}^{lit\_inj}$ zwischen Regelmodellen beschrieben, der auf Satz 3.1 beruht.

3.2.2 Redundanz von Regelmodellen

Um unnötigen Aufwand bei der Suche nach Hypothesen und beim Lernen von Regelmodellen zu vermeiden, ist es wichtig, redundante Regelmodelle aus einer Menge $\mathcal{R}$ von Regelmodellen zu entfernen.

Maher diskutiert Äquivalenzen von logischen Programmen und führt aus, daß zwei Klauseln C und D genau dann logisch äquivalent sind, wenn sie sich gegenseitig θ–subsumieren, also $C \sim_\theta D$ gilt [Maher, 1986]. Solche Klauseln fallen in dieselbe Äquivalenzklasse unter der Relation $\sim_\theta$. Vertreter dieser Klassen sollten die reduzierten Klauseln sein, da alle anderen redundante Literale enthalten.

Eine Menge $P1$ von Klauseln ist daher redundant in einer Menge $P2$, wenn es zu jeder Klausel $C1 \in P1$ eine Klausel $C2 \in P2$ mit $C1 \sim_\theta C2$ gibt.

Wie bereits beim Generalisierungsbegriff kann die Redundanz von Regelmodellen auf die Redundanz von Regeln zurückgeführt werden, indem Regelmodelle als Repräsentanten einer Menge von Regeln betrachtet werden. Ein Regelmodell R ist redundant in einer Menge von Regelmodellen $\mathcal{R}$, wenn es für alle Grundinstanzen von R logisch äquivalente Regeln aus $\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ gibt:

Definition 3.10 (Redundanz von Regelmodellen) *Ein Regelmodell R heißt* **redundant** *in einer Menge von Regelmodellen $\mathcal{R}$, gdw. für alle Regeln $C \in \mathcal{LH}_{\{R\}}^{lit\_inj}$ gilt:*

$$\exists \textit{ Regel } C' \in \mathcal{LH}_{\mathcal{R}}^{lit\_inj} : C \sim_\theta C'.$$

Redundanz kann auch innerhalb eines Regelmodells auftreten. Analog zu Klauseln erster Ordnung heißt ein Literalschema $L \in R$ genau dann redundant in einem Regelmodell R, wenn $R \vdash_{RS}^{lit\_inj} R \setminus \{L\}$. Dies bedeutet aber, daß in R zwei Literalschemata unifiziert werden, die θ–subsumtionsäquivalent sind, da im Suchraum $\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ andere Unifikationen von Literalschemata nicht möglich sind. Regelmodelle können also nur mit $\Sigma = \emptyset$ bzgl. $\vdash_{RS}^{lit\_inj}$ reduziert werden.

Definition 3.11 (Reduktion eines Regelmodelles) *Ein Regelmodell R heißt genau dann* **reduziert**, *wenn für alle Regelmodelle $R' \subseteq R$ gilt: Aus $R' \sim_{\vdash_{RS}^{lit\_inj}} R$ folgt, daß R und R' alphabetische Varianten sind.*

Beispiel 3.3 *Das Regelmodell*

$$\mathsf{Q(X)} \leftarrow \mathsf{P(X,Y)}\ \ \&\ \mathsf{P(X,Z)}$$

läßt sich zu

$$\mathsf{Q(X)} \leftarrow \mathsf{P(X,Z)}$$

reduzieren und

$$\mathsf{Q(X,Z)} \leftarrow \mathsf{P(X,Y)}\ \ \&\ \mathsf{P(Y,Z)}$$

ist redundant, falls das Regelschema

$$\mathsf{Q(X,Z)} \leftarrow \mathsf{P1(X,Y)}\ \ \&\ \mathsf{P2(Y,Z)}$$

vorhanden ist. ◇

Die Redundanz von Literalschemata in Regelmodellen führt zu einem weiteren Einwand gegen die Verwendung der Relation $\vdash_{RS}^{inj}$. So ist die Reduktion von

$$\mathsf{Q(X)} \leftarrow \mathsf{P(X,Y)}\ \ \&\ \mathsf{foo(X,Z)}$$

bzgl. $\vdash_{RS}^{inj}$ das Regelmodell

$$\mathsf{Q(X)} \leftarrow \mathsf{foo(X,Z)},$$

da mit $\Sigma = \{P/foo\}$ keine Prädikatsvariablen unifiziert werden und das erste Regelmodell nach dieser Substitution äquivalent zum zweiten Modell ist. Im Suchraum $\mathcal{LH}^{inj}_{\{\mathsf{P(X,Y)}\ \&\ \mathsf{foo(X,Z)} \rightarrow\ \mathsf{Q(X)}\}}$ liegt aber auch die nicht äquivalente Klausel

$$\mathsf{Q(X)} \leftarrow \mathsf{pred(X,Y)}\ \ \&\ \mathsf{foo(X,Z)},$$

und somit ist die Relation $\vdash_{RS}^{inj}$ bezüglich der Redundanz von Literalschemata nicht wohldefiniert. Dagegen läßt sich obiges Regelmodell nicht bzgl. $\vdash_{RS}^{lit\_inj}$ reduzieren, da diese Relation die Unifikation nicht θ–subsumtionsäquivalenter Literalschemata ($\mathsf{P(X,Y)}$ und $\mathsf{foo(X,Z)}$) verbietet.

Die Definition der Redundanz von Regelmodellen läßt sich mit Hilfe der Reduktion eines Regelmodelles operationalisieren. Dabei wird aus Komplexitätsgründen nur die Redundanz eines Regelmodells aufgrund eines einzelnen anderen Regelmodells erkannt, nicht aber ob dieses Regelmodell durch eine Menge anderer Regelmodelle überflüssig wird.

Satz 3.2 *Ein Regelmodell R mit der Reduktion R_{red} ist genau dann* **redundant** *aufgrund des Regelmodells R', wenn*

1. $|R'_{red}| = |R_{red}|$*, wobei R'_{red} die Reduktion von R' ist und*

2. $R'_{red} \vdash_{RS}^{lit\_inj} R_{red}$ *mit $R'_{red}\Sigma\ddot{\theta} \subseteq R_{red}$, wobei Bild($\Sigma$) und Bild($\ddot{\theta}$) keine Konstanten enthalten und $\ddot{\theta}$ eine injektive Abbildung ist.*

Beweis

„ $\Rightarrow$ “

R ist redundant in $\{R'\}$, nach Definition 3.11 und Definition 3.8 folgt: $R' \vdash_{RS}^{lit\_inj} R$ und somit auch $R'_{red} \vdash_{RS}^{lit\_inj} R_{red}$. Da alle Regeln $C \in \mathcal{LH}_{\{R\}}^{lit\_inj}$ außerdem auch θ–subsumtionsäquivalent zu Grundinstanzen von R' sind, gelten die Punkte 1. und 2. des obigen Satzes.

„ $\Leftarrow$ “

Aus $R'_{red} \vdash_{RS}^{lit\_inj} R_{red}$ folgt, daß für alle Regeln $C \in \mathcal{LH}_{\{R\}}^{lit\_inj}$ gilt:

$$\exists \text{ Regel } C' \in \mathcal{LH}_{\{R'\}}^{lit\_inj} : C' \vdash_\theta C.$$

Nach Punkt 1. und den Anforderungen an Bild(Σ), Bild($\ddot{\theta}$) und $\ddot{\theta}$ in Punkt 2. des Satzes sind die Regeln C' aber nicht echt allgemeiner als die Regeln C, d.h. es gilt

$$C' \sim_\theta C.$$

□

Die Reduktion von Regelmodellen läßt sich, wie schon die Subsumtion, auf einen modifizierten Algorithmus zur Reduktion von Hornklauseln reduzieren (siehe Abschnitt 4.3.5). In Kombination mit dem Algorithmus für $\vdash_{RS}^{lit\_inj}$ kann ein Redundanztest direkt auf obigem Satz basieren.

3.3 Lernen mit Regelmodellen in der Werkbank MOBAL

Das Lernverfahren RDT entdeckt Regelhaftigkeiten im operationalen Modell einer Sachbereichstheorie. Eingabe dieses Lernverfahrens sind alle Fakten der Wissensbasis. Die Inferenzmaschine IM–2 nutzt gelernte oder eingegebene Regeln vor dem Lernvorgang zur Inferenz weiterer Fakten. Der Suchraum wird durch Modellwissen in Form von Regelmodellen und der Prädikatstopologie definiert, d.h. er umfaßt alle Regeln, die im Raum $\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ einer Menge $\mathcal{R}$ von Regelmodellen liegen und kompatibel zur Prädikatstopologie sind. Ausgabe des Verfahrens ist eine Menge von Regeln, die von der Inferenzmaschine zur Inferenz weiterer Fakten benutzt werden (*closed loop learning*). Durch Vorgabe einer Liste von Zielrelationen beschränkt RDT sich auf die Entdeckung von Regeln, die diese Relationen in der Konklusion beschreiben.

Die Semantik dieses Werkzeuges ist abhängig von mehreren Parametern. Diese Parameter legen fest, ob eine Regel als Hypothese akzeptiert wird, insbesondere müssen die gelernten Regeln nicht notwendigerweise einen Begriff definieren. Sei H eine mögliche Hypothese:

$$H = p(X_1, \ldots, X_m) \rightarrow q(X_1, \ldots, X_n),$$

wobei $m \geq n$ und $p(X_1, \ldots, X_m)$ die Konjunktion aller Prämissen von H über den Variablen $X_1, \ldots, X_m$ ist. Die Parameter werden mit Hilfe folgender primitiver Funktionen ausgewertet[7]:

$pos(H) := \{(c_1, \ldots, c_n) | p(c_1, \ldots, c_m) \mathbin{\&} q(c_1, \ldots, c_n)\}$, die positiven Instanzen von H.

$neg(H) := \{(c_1, \ldots, c_n) | p(c_1, \ldots, c_m) \mathbin{\&} not(q(c_1, \ldots, c_n))\}$, die negativen Instanzen von H.

$pred(H) := \{(c_1, \ldots, c_n) | p(c_1, \ldots, c_m) \mathbin{\&} unknown(q(c_1, \ldots, c_n))\}$, die unbekannten Instanzen der Konklusion, die von H vorhergesagt werden.

$total(H) := pos(H) \cup neg(H) \cup pred(H)$.

Die zum weiteren Verständnis notwendigen Parameter dieses Werkzeuges sind[8]:

min_no_of_pos_examples Dieser Wert bestimmt die Mindestanzahl der positiven Instanzen, die eine Hypothese haben soll, d.h. für den aktuellen Wert **Wert** dieses Parameters muß gelten: $|pos(\mathsf{H})| \geq \mathsf{Wert}$. Setzt man diesen Wert auf **unrestricted**, so wird dieser Parameter nicht ausgewertet.

inductive_leap_percentage Diese ganze Zahl zwischen 0 und 100 bestimmt die prozentuale Vorhersage der Hypothese. Bei kleinen Werten wird eine Hypothese auch dann akzeptiert, wenn die Anzahl der positiven Instanzen klein ist im Verhältnis zu allen möglichen, d.h. hier wird der Ausdruck $|pos(\mathsf{H})| \geq \mathsf{Wert}/100 * |total(\mathsf{H})|$ ausgewertet. Auch hier ist der Wert **unrestricted** erlaubt.

closed_world_assumption Hat dieser Parameter den Wert **yes**, so werden alle nicht als wahr bekannten Instanzen der Konklusion als negative Instanz betrachtet, d.h. der Wert **Negative**, der in dem nächsten Parameter benutzt wird, beträgt: $\mathsf{Negative} = |pred(H)| + |neg(H)|$. Ist der Wert **no**, so gilt: $\mathsf{Negative} = |neg(H)|$.

max_no_of_exceptions Dieser Wert bestimmt die maximale Anzahl der negativen Instanzen einer Regel, d.h. für den aktuellen Wert **Wert** dieses Parameters muß gelten: $\mathsf{Negative} \leq \mathsf{Wert}$. Setzt man diesen Wert auf **unrestricted**, so wird dieser Parameter nicht ausgewertet.

topology_restrictions Dieser Parameter bestimmt, ob eine Hypothese kompatibel zur Prädikatstopologie sein muß, d.h. seine möglichen Werte sind **yes** und **no**.

---

[7] Diese Definitionen stammen aus [Morik *et al.*, 1993, Kapitel 6, Seite 185]. Dort werden weitere in dieser Arbeit nicht benötigte Primitive definiert.

[8] Eine vollständige Beschreibung der Parameter von RDT findet sich in [Sommer *et al.*, 1993].

Die aktuellen Werte dieser Parameter bilden das Akzeptanzkriterium für eine Hypothese. RDT lernt die bezüglich der θ–Subsumtion allgemeinsten Regeln im Suchraum, die dieses Akzeptanzkriterium erfüllen. Damit läßt sich die Semantik dieses Verfahrens wie folgt formalisieren:

Definition 3.12 (akzeptierende Semantik von RDT, RDT($\mathcal{A}, \mathcal{R}, \mathcal{T}$)) *Seien B eine Menge von Fakten und $\mathcal{R}$ eine Menge von Regelmodellen. Sei weiterhin $\mathcal{A}$ das aktuelle Akzeptanzkriterium und $\mathcal{T}$ eine Menge von Zielrelationen. Die* **akzeptierende Semantik** RDT($\mathcal{A}, \mathcal{R}, \mathcal{T}$) *des Werkzeuges* RDT *besteht im Entdecken einer Menge von Regeln H, so daß $\forall h \in H$ gilt:*

- *$h \in \mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ und die Kopfrelation p von h ist Element von $\mathcal{T}$,*
- *h erfüllt das Akzeptanzkriterium $\mathcal{A}$ und*
- *es gibt keine Regel $h' \neq h$ mit $h' \vdash_\theta h$, die diese Punkte erfüllt.*

Mit $\mathcal{A} =$ {closed_world_assumption = yes, min_no_of_pos_examples = unrestricted, max_no_of_exceptions = 0, inductive_leap_percentage = unrestricted} und einer beliebigen Menge von Zielrelationen ist eine Lösung H dieser Semantik auch eine Lösung für die nicht monotone ILP Semantik mit der Hypothesensprache $\mathcal{LH}_{\mathcal{R}}^{lit\_inj}$.

Gibt es nur eine Zielrelation, dann kann RDT zum Lernen einer Begriffsbeschreibung eingesetzt werden: Sei $E^+ \subseteq B$ die Menge der positiven Fakten einer Zielrelation $\mathcal{T}$. Mit $\mathcal{A} =$ {closed_world_assumption = no, min_no_of_pos_examples = $|E^+|$, max_no_of_exceptions = 0, inductive_leap_percentage=unrestricted} lernt RDT Begriffsbeschreibungen in einer Hypothesensprache, die nur Hypothesen bestehend aus einer einzigen Regel $h \in \mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ zuläßt, oder formal RDT($\mathcal{A}, \mathcal{R}, \mathcal{T}$) $= ILP(\vdash_\theta, Fakten, Fakten, \mathcal{LH}_{\mathcal{R}}^{lit\_inj1})$.

Das Lernverfahren RDT versucht vorgegebene Regelmodelle sukzessiv mit Prädikaten des Hintergrundwissens zu instanziieren und zu testen, ob die resultierenden Regeln gemäß des vom Benutzer wählbaren Kriteriums akzeptabel sind. Mit Hilfe der Erweiterung der θ–Subsumtion auf Regelmodelle läßt sich der Suchraum dabei so strukturieren, daß von generelleren Regelmodellen zu spezielleren gesucht werden kann und Teile des Suchraumes während eines Lernlaufes abgeschnitten werden können[9].

3.4 Verwandte Ansätze

Regelmodelle als Bestandteil des Repräsentationsformalismus wurden schon im System METAXA [Emde *et al.*, 1983] eingeführt, einem Vorgänger der Werkbank MOBAL. Die in diesem Kapitel definierte Generalisierungrelation

[9] Für Einzelheiten sei auf [Kietz und Wrobel, 1991], [Morik *et al.*, 1993, S. 184ff] verwiesen. Dort wird der Suchraum $\mathcal{LH}_{\mathcal{R}}$ zusammen mit der Generalisierungsrelation $\vdash_{RS}^{inj}$ verwendet, die in diesem Kapitel als zu restriktiv beschrieben wurde.

$\vdash_{RS}^{lit\_inj}$ verbessert das Lernverfahren RDT, da die bisher dort benutzte Relation $\vdash_{RS}^{inj}$ nicht der θ–Subsumtion von Klauselmengen entspricht und daher ein optimales Abschneiden des Suchraumes während des Lernvorganges verhindert. Zusätzlich wird erstmals ein Redundanzbegriff für Regelschemata definiert, der auf der Äquivalenz der Instanzen bzgl. der Relation $\vdash_\theta$ beruht. Damit werden alle nicht zum Lernen benötigten Regelschemata erfaßt. Der existierende Redundanztest des Systems MAT [Thieme, 1989] erkennt nur Schemata, die bis auf Permutation und Umbenennung von Variablen gleich sind.

Einige andere modellbasierte Ansätze verwenden ebenfalls Regelmodelle in ihren Lernverfahren. De Raedt und Bruynooghe benutzen diese im System CLINT–CIA zur Bildung neuer Begriffsbeschreibungen durch Analogieschlüsse [De Raedt, 1992]. Sehr verwandt sind auch die relationalen Klischees von FOCL [Pazzani und Kibler, 1992], die dort in einem *greedy top–down* Verfahren eingesetzt werden. Klingspor definiert für das Lernverfahren GRDT kontextfreie Grammatiken zur Beschreibung von Regelmodellen [Klingspor, 1994]. Dadurch können Schemata beliebiger Länge kompakt formuliert werden. Ähnliche Grammatiken finden sich auch im System GRENDEL, nur wird dort nicht von den Prädikatsnamen abstrahiert [Cohen, 1994]. Ebenfalls nicht von den Prädikatsnamen abstrahieren die Literalmengen, die die induktiven Lernverfahren FILP und TRACY zur Beschreibung des Hypothesenraumes einsetzen [Bergadano und Gunetti, 1994].

Das Verfahren CAN schränkt den Suchraum durch Programmierschemata ein, die während der Induktion durch Analogieschlüsse angepasst werden können [Tausend, 1992]. Die Abhängigkeitsgraphen des Systems SIERES abstrahieren nicht nur von den Prädikatsnamen der Literale, sondern auch von den Argumenten [Wirth und O'Rorke, 1991].

Kapitel 4

Datengesteuertes Lernen von Regelmodellen

Die Festlegung geeigneter Regelmodelle zum Entdecken von Regelhaftigkeiten für bestimmte Zielrelationen ist im System MOBAL momentan Aufgabe des Benutzers. Da Regelmodelle nur von den Prädikatssymbolen und von konstanten Termen in den Argumenten der Literale abstrahieren, sind weitgehende Kenntnisse über die Sachbereichstheorie und die zu entdeckenden Regeln notwendig, um den Hypothesenraum mit Hilfe dieser Schemata so zu definieren, daß die gesuchten Regeln im Suchraum liegen.

Wie für alle Bestandteile des Repräsentationsformalismus im System MOBAL existiert auch für Regelmodelle ein Werkzeug zur automatischen Gewinnung entsprechender Ausdrücke. Das Werkzeug MAT (engl. *model aquisition tool*) unterstützt den Benutzer aber nur insofern, daß anstelle von Regelmodellen Regeln eingegeben werden können, die durch Umwandlung von Prädikatssymbolen in Prädikatsvariablen zu Regelmodellen abstrahiert werden [Thieme, 1989]. Das abstrahierte Regelmodell wird dann auf Redundanz in der Menge der bereits vorhandenen Regelmodelle getestet.

Eine ideale Ergänzung zum Werkzeug RDT des Systems MOBAL wäre daher eine Methode zum Erwerb von Regelmodellen für eine Zielrelation. Die Eingabe eines derartigen Verfahrens bestände aus dem Hintergrundwissen und den Beispielen der Zielrelation. Auszugeben wären die zur gesuchten Menge von Regeln korrespondierenden Regelmodelle beziehungsweise eine Obermenge davon.

In Abschnitt 4.1 soll zunächst diese Zielsetzung formalisiert und auf ihre Komplexität untersucht werden. Daraufhin wird in Abschnitt 4.2 ein heuristischer Ansatz vorgestellt, der Methoden der induktiven logischen Programmierung benutzt. Abschnitt 4.3 stellt die aus diesem Ansatz resultierende Erweiterung des bisherigen Werkzeuges MAT vor. Den Abschluß dieses Kapitels bildet die Schilderung verwandter Ansätze und eine Zusammenfassung des Verfahrens.

4.1 Die Komplexität des Lernens von Regelmodellen

Die Bestimmung der Komplexität des Problems, Regelmodelle zu lernen, setzt die Formalisierung dieser Aufgabe voraus. Entscheidend für die Formalisierung ist die Verknüpfung dieses Problems mit der akzeptierenden Semantik des Werkzeuges RDT. Die Aufgabenstellung kann nicht im Erwerb von Regelmodellen für ein konkretes Akzeptanzkriterium liegen. Eine solche Zielsetzung würde ein erneutes Aufrufen dieses Verfahrens nach jedem Ändern des Akzeptanzkriteriums zu Folge haben und außerdem eher ein direktes Lernen von Regeln, die das konkrete Kriterium erfüllen, motivieren.

Definition 4.1 (Lernen von Regelmodellen, RM**)** *Gegeben seien eine Zielrelation* $\mathcal{T}$ *und eine Sachbereichstheorie* B*. Ein Algorithmus löst das Problem, Regelmodelle für ein beliebiges Akzeptanzkriterium* $\mathcal{A}$ *von* RDT *zu lernen, wenn er bei Eingabe von* B *und* $\mathcal{T}$ *eine Menge* $\mathcal{R}$ *von Regelmodellen berechnet, so daß für alle* H *als Lösung von* $\mathrm{RDT}(\mathcal{A}, \mathcal{R}^\star, \mathcal{T})$ *gilt:* $H \in \mathcal{LH}_{\mathcal{R}}^{lit\_inj}$*, wobei* $\mathcal{R}^\star$ *eine beliebige Menge von Regelmodellen ist. Existiert für keine Menge* $\mathcal{R}^\star$ *von Regelmodellen und kein Akzeptanzkriterium* $\mathcal{A}$ *eine solche Lösung* H*, so antwortet der Algorithmus mit „Fehler“.*

Um die Komplexität dieser Aufgabe zu bestimmen, betrachte man die in Abschnitt 3.3 dargestellte Zielsetzung von RDT, unter der dieses Werkzeug eine Begriffsbeschreibung im Sinne von Definition 2.1 lernt: Es gibt nur eine Zielrelation $\mathcal{T}$ und $E^+ \subseteq B$ sei die Menge der positiven Fakten dieser Zielrelation. Mit $\mathcal{A} = \{$closed_world_assumption = no, min_no_of_pos_ examples = $|E^+|$, max_no_of_exceptions = 0, inductive_leap_ percentage=unrestricted$\}$ lernt RDT Begriffsbeschreibungen in der Hypothesensprache $\mathcal{LH}_{\mathcal{R}^\star}^{lit\_inj1}$. Sei $\mathcal{R}$ eine Lösung für RM bei Eingabe von $\mathcal{T}$ und B. Nach Definition gibt es dann auch eine Lösung $H \in \mathcal{LH}_{\mathcal{R}}^{lit\_inj}$ für obige Zielsetzung $\mathrm{RDT}(\mathcal{A}, \mathcal{R}^\star, \mathcal{T})$. Die Kombination des Lernens von Regelmodellen mit dem Lernverfahren RDT kann nach diesem Ablauf zur Lösung von $ILP(\vdash_\theta, \emptyset, \{e \leftarrow B \mid e \in E\}, \mathcal{LH}^1)$ für beliebige Sprachen $\mathcal{LH}^1$ eingesetzt werden. Wie in Abschnitt 2.2.3 geschildert, ist dieses Problem aber für bestimmte Hypothesensprachen nicht in polynomieller Zeit lernbar. Eines der beiden Teilprobleme RM und RDT kann daher nicht zugleich effizient und vollständig sein. Welches dieser Probleme aber schwierig im Sinne der Komplexitätstheorie ist, kann nicht allgemein angegeben werden, sondern ist vom Einzelfall abhängig.

So wird im Beweis von Kietz zur *NP*–Schwierigkeit des Konsistenzproblems für $ILP(\vdash_\theta, \emptyset, \{e \leftarrow B \mid e \in E\}, \mathcal{LH}^1_{12-notDET})$ das SAT Problem auf ein entsprechendes Lernproblem reduziert und bewiesen, daß alle Lösungen dieses Problems Hypothesen der Form $p(X) \leftarrow p(X, Y_k), a_1(Y_k), \ldots, a_{2n}(Y_k)$ sind [Kietz, 1993b, Lemma 3]. Dabei ist n die Anzahl der boolschen Variablen des SAT Problems und $1 \leq k \leq 2^n$. Natürlich läßt sich für diese Hypothesen leicht ein Regelmodell angeben, das entsprechende RDT Problem bleibt aber schwierig. Bei determinierenden Hypothesensprachen dagegen ist das Problem RM schwierig und das Problem RDT wird einfach.

Daher wird sich kein effizientes und zugleich vollständiges Verfahren zum Lernen von Regelmodellen für beliebige Akzeptanzkriterien entwickeln lassen. Im folgenden Abschnitt wird daher ein heuristischer Ansatz beschrieben.

4.2 Ein heuristischer Ansatz zum Lernen von Regelmodellen

Einen ersten intuitiven Ansatzpunkt für ein heuristisches Verfahren zum Lernen von Regelmodellen liefert ein weiteres Werkzeug von MOBAL zur Inspektion des Sachbereichswissen: Der Fakten–Graph ist die inkrementell erweiterbare Darstellung der Wissensbasis als Graph. Die Knoten des Graphen sind die Argumente der Fakten (die Objekte des Sachbereichs) und die Kanten entsprechen den Prädikaten, in denen diese Objekte vorkommen. Abbildung 4.1 zeigt einen Ausschnitt eines solchen Fakten–Graphen.

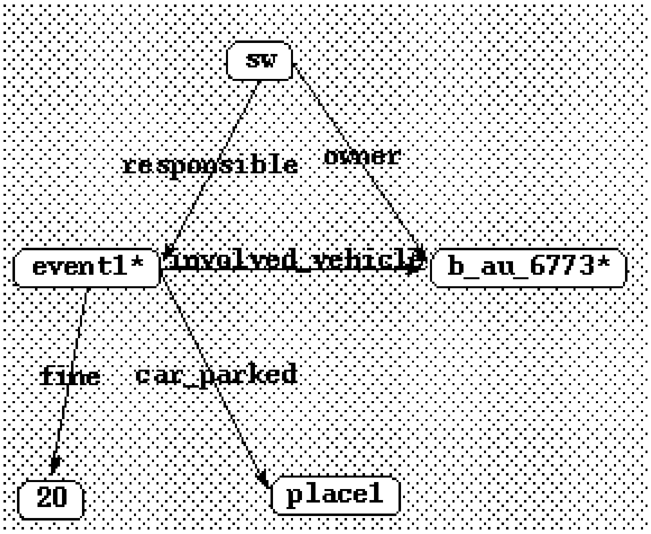

Abbildung 4.1: Ausschnitt eines Fakten–Graphen des Objektes event1

Das Objekt sw ist über eine mit responsible gewichtete Kante mit dem Objekt event1 verbunden. Die beiden Objekte stehen somit bezüglich responsible in Relation zueinander, d.h. es existiert ein Faktum responsible(sw,event1) in der Sachbereichstheorie, die durch diesen Ausschnitt eines Fakten–Graphen zum Teil repräsentiert wird.

Der resultierende Graph der Wissensbasis spiegelt also wieder, wie die Prädikate über ihre Argumente verbunden sind. Ebenso legen Regelmodelle fest, wie Objekte (repräsentiert durch Termvariablen) über Prädikate (repräsentiert durch Literalschemata) miteinander verbunden sind.

Die grundlegende Idee eines heuristischen Ansatzes zum Erwerb von Regelmodellen liegt nun darin, daß ein solcher Fakten–Graph in irgendeiner Art und Weise sowohl die zu entdeckenden Regelhaftigkeiten als auch die korrespondierenden Regelmodelle beinhalten muß. Auf einer Idee von Jörg–Uwe Kietz und Katharina Morik basiert das in Abbildung 4.2 skizzierte Gerüst für einen entsprechenden Algorithmus.

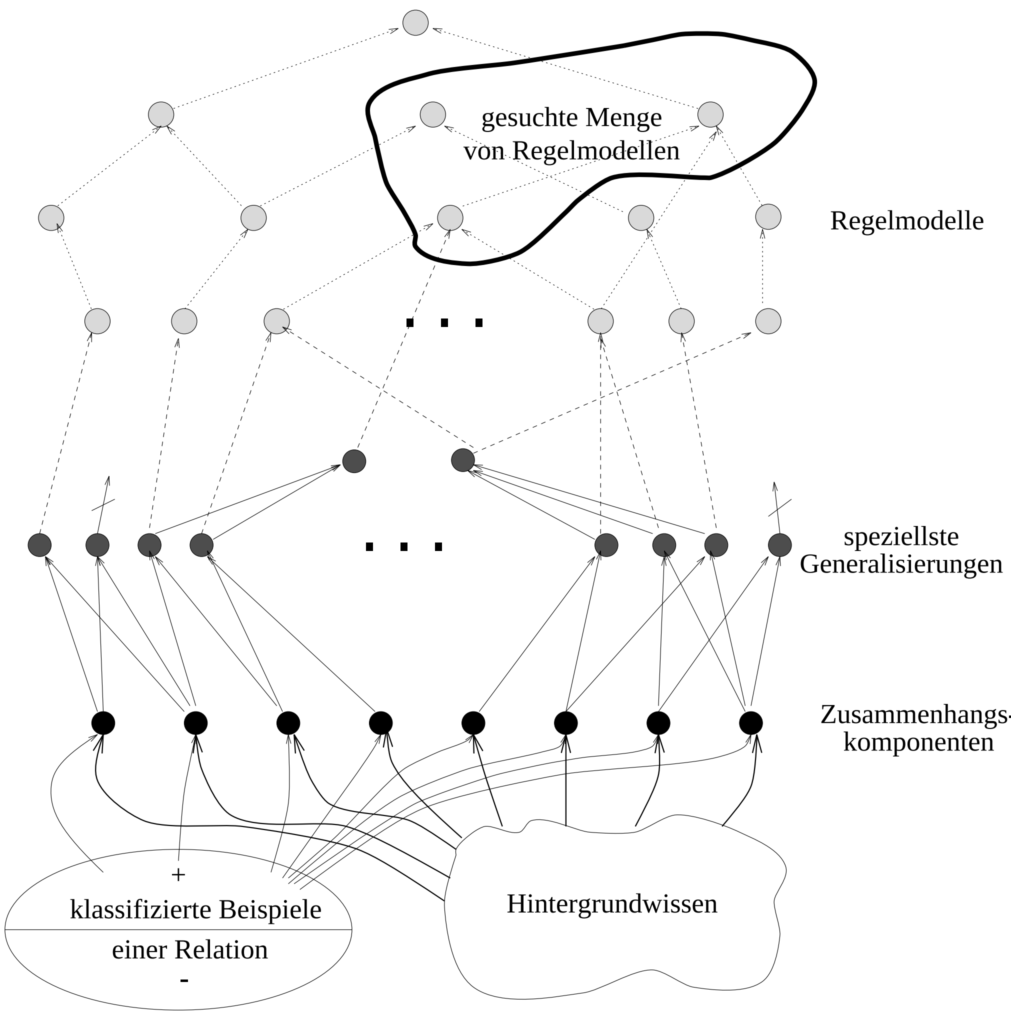

Abbildung 4.2: Skizze eines Verfahrens zum Erwerb von Regelmodellen

Das Hintergrundwissen B und die Beispiele $E \subseteq B$ für die betrachtete Zielrelation werden als Graph repräsentiert. Die Knoten des Graphen bilden die einzelnen Fakten aus B, Kanten existieren zwischen Fakten (Knoten) mit gemeinsamen Termen. Für jedes positive Beispiel $e \in E^+$ bestimmt man die Zusammenhangskomponente im Graphen.

Notiert man die Zusammenhangskomponente als variablenfreie Hornklausel mit e als Konklusion und allen anderen Literalen als Prämisse, kann man die speziellste Generalisierung von je zwei oder auch mehreren Zusammenhangskomponenten konstruieren.

Die aus der Konstruktion der speziellsten Generalisierungen resultierenden Klauseln lassen sich entweder direkt oder zuvor abstrahiert (analog zum Vorgehen im Werkzeug MAT) als Regelmodelle formulieren. Diese Modelle bilden die unteren Schranken in einem Suchraum, dessen Struktur durch die Relation $\vdash_{RS}^{lit\_inj}$ als Erweiterung der θ–Subsumtion auf Regelmodelle festgelegt wird. Innerhalb dieser Ordnung gilt es nun, mit einer geeigneten Strategie nach einer Menge von Regelmodellen zu suchen, die unter ihren Instanzen die gesuchten Regeln enthalten. Dieser Algorithmus konstruiert die Regelmodelle also *bottom–up*, d.h. von den spezielleren zu den generelleren Modellen hin.

Das skizzierte Verfahren besteht im wesentlichen aus drei Schritten: Das Finden von Zusammenhangskomponenten (im folgenden Faktenketten) im skizzierten Graphen der Wissensbasis ähnelt der Verkettung oder Saturierung der Beispiele mit dem Hintergrundwissen, wie sie bereits in Kapitel 2.2.3 beschrieben wurde. Für diesen Schritt werden Sprachen vorgestellt, die die Verkettung einschränken und es wird untersucht, für welche Lernprobleme diese Beispielsprachen vollständig sind. Die Konstruktion von speziellsten Generalisierungen dieser Faktenketten folgt den in Abschnitt 2.3 geschilderten Ansätzen. Der letzte Schritt besteht in der weiteren Generalisierung und Abstraktion dieser Zwischenergebnisse zu den gesuchten Regelmodellen.

4.2.1 Saturierung der Beispiele

Der zu Beginn dieses Kapitels vorgestellte Ansatz zum Erwerb von Regelmodellen basiert auf der Verallgemeinerung von Zusammenhangskomponenten eines Graphen, der die Verbindung von Fakten des Hintergrundwissens und der Beispiele repräsentiert. Die Bildung der Faktenketten ist erforderlich, da die θ–Subsumtion zwar eine entscheidbare Ableitungsregel ist, dafür aber kein Hintergrundwissen berücksichtigt. Die Konstruktion speziellster Generalisierungen bezüglich der θ–Subsumtion setzt daher ein Hineinrechnen des Hintergrundwissens in die Beispiele voraus. In Abschnitt 2.2.2 wurde bereits die Konstruktion der Menge $\{e \leftarrow B' \mid e \in E \wedge B' \subseteq B\}$ als für bestimmte Hypothesensprachen vollständige Saturierung der Beispiele bei Verwendung der θ–Subsumtion charakterisiert. Die Bildung von Faktenketten entspricht somit obiger Sprache, falls in B' alle mit der Konklusion verbundenen Fakten aus B enthalten sind. Sie ist somit vollständig zum Lernen verbundener Hornklauseln, einer sehr oft gemachten und sinnvollen Einschränkung der Hypothesensprache. Unter Verwendung des Repräsentationsformalismus von MOBAL lassen sich Faktenketten wie folgt definieren:

Definition 4.2 (Faktenkette, $factchain(e, B)$) *Gegeben seien eine Sachbereichstheorie B und Beispiele $E \subseteq B$ einer Relation. Die bezüglich $\vdash_\theta$ speziellsten Regeln der Menge $\{e \leftarrow b_1, \ldots, b_n \mid e \in E \wedge \{b_1, \ldots, b_n\} \subseteq$*

$B \setminus e$ $\wedge$ alle Literale in $e \leftarrow b_1, \ldots, b_n$ sind verbunden$\}$ heißen **Faktenketten**, *$factchain(e, B)$, der Beispiele E.*

Gibt es nur eine Zusammenhangskomponente im Graph der Sachbereichstheorie so ist die Anzahl der Literale in den Faktenketten gleich $|B|$. Nach Kapitel 2.2.3 liegt die Anzahl der Literale einer speziellsten Generalisierung von zwei Faktenketten dann in $O(|B|^2)$. In Kapitel 5 werden Lernprobleme vorgestellt, bei denen alle Fakten derart verknüpft sind, daß sie eine einzige Zusammenhangskomponente bilden. Zudem besteht die Wissensbasis in diesen Fällen aus über tausend Literalen. Die aus den Faktenketten zu konstruierenden speziellsten Generalisierungen enthalten dann zu viele Literale. Da die Anzahl der Literale einer Generalisierung durch ein Polynom in der Größe der Faktenketten begrenzt ist, gilt es, die Faktenketten auf relevante Teilmengen ihrer Literale zu begrenzen.

Im folgenden sollen nun verschiedene Einschränkungen von Faktenketten hinsichtlich ihrer Auswirkung auf das Lernergebnis des anvisierten Verfahrens zum Lernen von Regelmodellen charakterisiert werden. Dazu wird untersucht für welche Hypothesensprachen das Lernen aus diesen reduzierten Faktenketten noch vollständig ist.

Besteht der Suchraum z.B. aus nicht rekursiven Hornklauseln, so enthält die Faktenkette $factchain(e, B \setminus E)$ bereits alle zu betrachtenden Literale. Andere Sprachen $\mathcal{LH}$ lassen aber weitere Effizienzsteigerungen zu. Die Grundlage dazu bilden Übertragungen der aus Kapitel 2.2.3 bekannten Einschränkungen der Hypothesensprachen auf die Konstruktion von Faktenketten.

ij–nicht determinierende Faktenkette

Nach Definition 4.2 sind Faktenketten immer verbunden (siehe Definition 2.15). Analog zur Definition 2.18 von ij–nicht determinierenden Klauseln lassen sich ij–nicht determinierende Faktenketten definieren:

Definition 4.3 (ij–nicht determinierende Faktenkette) *Eine Faktenkette, deren Literale die maximale Stelligkeit j und deren Terme die maximale Tiefe i besitzen, heißt* **ij–nicht determinierende Faktenkette**.

Beispiel 4.1 *In der nachstehenden Faktenkette haben die Terme* ann *und* bob *die Tiefe* 0*, da sie im Kopfliteral der Regel vorkommen. Die Terme* tom *und* jim *haben Tiefe* 1*, die Terme* zak *und* sue *Tiefe* 2 *und der Term* eve *die Tiefe* 3*.* liz *und* pat *sind dagegen nicht mit dem Beispiel verbunden, d.h. das Literal* mutter(liz,pat) *gehört nicht zur Faktenkette des Beispiels* großmutter(ann,bob)*:*

großmutter(ann,bob) $\leftarrow$ weiblich(ann) & vater(tom,bob)
& mutter(ann,tom) & mutter(ann,jim)
& vater(tom,sue) & tochter(sue,tom)
& vater(zak,tom) & mutter(eve,sue)
[& mutter(liz,pat)]. $\diamond$

Unterliegen die Hypothesen eines Suchraumes einer Tiefenbeschränkung, so ist es ausreichend die ij–nicht determinierenden Faktenketten der Beispiele zu berechnen.

Satz 4.1 *Seien B eine Sachbereichstheorie und $E \subseteq B$ eine Beispielmenge einer Relation. Dann sind die ILP–Probleme ($\vdash_\theta$,$\{\}$,$\{e \leftarrow B \setminus \{e\} \mid e \in E\}$, ij–nicht determinierende Hornklauseln) und ($\vdash_\theta$,$\{\}$,ij–nicht determinierende Faktenketten, ij–nicht determinierende Hornklauseln) äquivalent. Ebenso ist es für die akzeptierende Semantik von* RDT *vollständig, ij–nicht determinierende Faktenketten zur Auswertung des Akzeptanzkriteriums zu benutzen, falls nur ij–nicht determinierende Regeln im Suchraum der Regelmodelle liegen.*

Zum Beweis dieses Satzes wird eine Aussage über die Veränderung der Tiefe einer Regel durch die Anwendung einer Grundsubstitution benötigt.

Hilfsatz 4.1 *Seien h eine ij–nicht determinierende Regel und θ eine Substitution, die alle Variablen aus h durch Konstanten ersetzt. Dann ist $h\theta$ eine dj–nicht determinierende Faktenkette mit $d \leq i$.*

Beweis (des Hilfsatzes) Der Beweis erfolgt durch Induktion über die Tiefe der Terme. Alle Variablen der Tiefe 0 aus h sind Terme des Konklusionsliterals, die substituierenden Konstanten dieser Variablen in $h\theta$ haben also auch Tiefe 0. Sei V eine Variable der Tiefe $s \leq i$ in der Klausel h, d.h. es gibt ein Literal l und einen Term t, so daß $t, V \in arg(l)$ gilt und der Term t die Tiefe $s - 1$ hat. Nach Induktionsvoraussetzung ist der Term $t\theta$ in $h\theta$ verbunden und hat die Tiefe $s' \leq s - 1$. Der Term $V\theta$ ist daher ebenfalls verbunden und hat höchstens die Tiefe s. □

Beweis Die Lernprobleme in Satz 4.1 sind äquivalent, wenn eine ij–nicht determinierende Hypothese $H \in \mathcal{LH}_{ij-notDET}$ bezüglich der θ–Subsumtion genau dann in Relation zu einem Beispiel $e \leftarrow B \setminus \{e\}$ steht, wenn sie auch die ij–nicht determinierende Faktenkette dieses Beispiels subsumiert, also gilt:

$$H \vdash_\theta e \leftarrow B \setminus \{e\} \Leftrightarrow H \vdash_\theta ij\text{–non determinate } factchain(e, B).$$

Sei $e \leftarrow S := ij$–non determinate $factchain(e, B)$.

" $\Rightarrow$ "

Es gilt $\exists \theta : H\theta \subseteq e \leftarrow B \setminus \{e\}$. Nach Satz 4.1 hat auch $H\theta$ höchstens Tiefe i und alle Literale sind verbunden, d.h. es gilt $H\theta = e \leftarrow S'$ mit $S' \subseteq B \setminus \{e\}$ und alle Literale aus S' sind mit einer Verbindungskette der maximalen Tiefe i mit dem Beispiel e verbunden. Da die Faktenkette $e \leftarrow S$ per Definition 4.3 alle Literale aus $B \setminus \{e\}$ mit dieser Eigenschaft enthält, gilt auch $S' \subseteq S$ beziehungsweise $H\theta \subseteq e \leftarrow S$.

" $\Leftarrow$ "

Natürlich gilt $S \subseteq B \setminus \{e\}$ und somit auch $H \vdash_\theta e \leftarrow B \setminus \{e\}$. □

ij–(schwach) determinierende Faktenkette

Die Tiefenbeschränkung für neue Variablen führt, wie in Kapitel 2.2.3 erläutert, erst in Verbindung mit dem Begriff der determinierenden Terme zu effizient lernbaren Hypothesensprachen. Die Definition 2.17 von ij–determinierenden Klauseln läßt sich aber nicht direkt auf Faktenketten übertragen, da hier alle Terme variablenfrei sind und daher jeder Term einer Faktenkette determinierend ist.

Die Idee der determinierenden Terme besteht darin, bei der Konstruktion von Hypothesen nur solche neuen Variablen einzufügen, deren Belegungen durch die anderen Terme eindeutig bestimmt sind.

Ein Beispiel hierfür ist die Klausel

$$\mathsf{großmutter(X,Y)} \leftarrow \mathsf{vater(U,Y)}\ \ \&\ \mathsf{mutter(X,U)}.$$

Zu jedem Kind Y gibt es genau einen Vater U und ebenso genau eine Mutter X von U. Dagegen ist die Klausel

$$\mathsf{großmutter(X,Y)} \leftarrow \mathsf{elternteil(X,Z)}\ \ \&\ \mathsf{tochter(Y,Z)}$$

nicht determinierend, da es zu einem Elternteil X mehrere Kinder Z geben kann, die Instanzen der Variable Z also nicht durch die Instanziierung der Variable X eindeutig festgelegt werden.

Bei der Übertragung auf Faktenketten sind zwei Definitionen sinnvoll:

Definition 4.4 (**ij–(schwach) determinierende Faktenkette**) *Seien B eine Sachbereichstheorie und $E \subseteq B$ eine Beispielmenge einer Relation. Sei h die Klausel, die durch gleichzeitiges Ersetzen aller Terme einer Faktenkette eines Beispiels $e \in E$ durch verschiedene Variablen entsteht.*
Eine Faktenkette heißt **ij–(schwach) determinierend** *bzgl. der Beispiele E und dem Hintergrundwissen B, wenn die so konstruierte Klausel h ij–determiniernd bzgl. der Beispielmenge E ($\{e\}$) und dem Hintergrundwissen B ist.*

Beispiel 4.2 *Die schwach determinierende Faktenkette von*

$$\mathsf{großmutter(ann,bob)} \leftarrow \mathsf{vater(tom,bob)}\ \ \&\ \mathsf{mutter(ann,tom)}$$
$$\&\ \mathsf{mutter(ann,jim)}\ \ \&\ \mathsf{tochter(sue,tom)}$$

ist somit die Klausel

$$\mathsf{großmutter(ann,bob)} \leftarrow \mathsf{vater(tom,bob)}\ \ \&\ \mathsf{mutter(ann,tom)}$$
$$\&\ \mathsf{tochter(sue,tom)},$$

da im obigen Beispiel der Vater tom *nur eine Tochter* sue *hat. Die determinierende Faktenkette dagegen ist die Klausel*

$$\mathsf{großmutter(ann,bob)} \leftarrow \mathsf{vater(tom,bob)}\ \ \&\ \mathsf{mutter(ann,tom)},$$

da jeder Sohn nur einen Vater hat und tom *dann auch in* mutter(ann,tom) *ein determinierender Term ist, aber nicht alle Väter nur eine Tochter haben und somit Instanzen von* großmutter(X,Z) *existieren, die die Tochter* T *in der Regel*

$$\mathsf{großmutter(X,Z)} \leftarrow \mathsf{vater(Y,Z)}\ \ \&\ \mathsf{mutter(X,Y)}\ \ \&\ \mathsf{tochter(T,Y)}$$

nicht determinieren. ◇

Während in der Definition der determinierenden Faktenketten gefordert wird, daß neu einzuführende Terme für alle Instanziierungen des Klauselkopfes durch die anderen Terme eindeutig festgelegt werden, werden bei schwach determinierenden Faktenketten nur mögliche Instanzen mit dem zu verkettenden Beispiel im Klauselkopf betrachtet. Wird ein neuer Term in einem Literal einer Faktenkette eingeführt, so ist es ausreichend, diesen in eine Variable umzuwandeln und die möglichen Instanzen dieses so erhaltenen Literals zu betrachten. Die Definition der determinierenden Faktenketten basiert auf der Definition der entsprechenden Klauseln und setzt somit eine Ordnung über den Literalen voraus. Diese hat aber keine Auswirkung auf die Konstruktion der determinierenden Faktenketten, da die Definition der Faktenketten jede Permutation umfaßt.

Analog zu Satz 4.1 ist für das Lernen von ij–determinierenden Klauseln die Konstruktion von determinierenden Faktenketten ausreichend. Der folgende Satz gilt sinngemäß auch wieder für die akzeptierende Semantik von RDT:

Satz 4.2 *Seien B eine Sachbereichstheorie und $E \subseteq B$ eine Beispielmenge einer Relation. Dann sind die ILP–Probleme* ($\vdash$,$\{\}$,$\{e \leftarrow B\backslash\{e\} \mid e \in E\}$,*$ij$–determinierende Klauseln*) *und* ($\vdash$,$\{\}$,*ij–determinierende Faktenketten, ij–determinierende Klauseln*) *äquivalent.*

Beweis Die beiden ILP–Probleme sind äquivalent, wenn eine ij–determinierende Hypothese $H \in \mathcal{LH}_{ij-DET}$ bezüglich der θ–Subsumtion genau dann in Relation zu einem Beispiel $e \leftarrow B \setminus \{e\}$ steht, wenn sie auch die ij–determinierende Faktenkette dieses Beispiels subsumiert, also gilt:

$$H \vdash_\theta e \leftarrow B \setminus \{e\} \Leftrightarrow H \vdash_\theta ij\text{–determinate } factchain(e, B).$$

Sei $e \leftarrow S := ij$–determinate $factchain(e, B)$.

" $\Rightarrow$ "

Es gilt $\exists\theta$: $H\theta = e \leftarrow S'$ mit $S' \subseteq B \setminus \{e\}$. Sei $arg(H\theta) = \{t_1, \ldots, t_n\}$ die Menge aller Terme aus $H\theta$. Weiterhin sei $\sigma^{-1} = \{t_1/V1, \ldots, t_n/V_n\}$ ($\forall i, j,\ 1 \leq i, j \leq n : V_i \neq V_j$) eine inverse Substitution, die alle Terme $H\theta$ durch unterschiedliche Variablen ersetzt. Dann gilt: $H\theta\sigma^{-1}$ ist bis auf Umbenennung von Variablen eine Teilmenge von H und somit determinierend bzgl. E und $B \setminus \{e\}$ und hat nach Satz 4.1 höchstens Tiefe i. Da die Faktenkette $e \leftarrow S$ per Definition 4.4 alle Literale aus $B \cup E \setminus \{e\}$ mit dieser Eigenschaft enthält, gilt auch $S' \subseteq S$ beziehungsweise $H\theta \subseteq e \leftarrow S$.

" $\Leftarrow$ "

Natürlich gilt $S \subseteq B \setminus \{e\}$ und somit auch $H \vdash_\theta e \leftarrow B \setminus \{e\}$. □

Jede determinierende Faktenketten ist Teilmenge der schwach determinierenden Faktenketten desselben Beispiels:

Folgerung 4.1 *Die ILP–Probleme* ($\vdash$,{},$\{e \leftarrow B \setminus \{e\} \mid e \in E\}$,*ij–determinierende Klauseln) und* ($\vdash$,{},*ij–schwach determinierende Faktenketten,ij–determinierende Klauseln) sind äquivalent.*

Obwohl das Lernen von determinierenden Hornklauseln auf Grundlage von determinierenden Faktenketten vollständig ist, macht es Sinn, sich auf die Konstruktion schwach determinierender Faktenketten zu beschränken. Dies liegt an der höheren Komplexität der Berechnung determinierender Faktenketten. Um zu testen, ob eine Variable determinierend bzgl. der Beispiele E und des Hintergrundwissens B ist, benötigt man $O(|E| * |B|)$ Substitutionen, die schwache Determiniertheit neuer Terme dagegen kann mit linearem Aufwand bzgl. $|B|$ getestet werden.

Beschränkung durch Prädikatstopologie

Die Beachtung einer vom Benutzer vorgegebenen Prädikatstopologie kann zur Reduzierung auf relevante Teile einer Faktenkette beitragen: Wurden die Prädikate großmutter und freundin der Faktenkette

großmutter(ann,bob) $\leftarrow$ vater(tom,bob) & mutter(ann,tom)
& freundin(ann,mary)

in zwei nicht kompatible Topologieknoten verwandschaft und freundschaft eingeordnet, so ist die Verkettung mit dem Literal freundin(ann,mary) zum Lernen von Regeln über die Relation großmutter nicht relevant. Wendet man diese Reduzierung bereits bei der Konstruktion der Faktenketten an, so erspart man unnötigen Aufwand bei der späteren Berechnung der Generalisierungen.

i–kompatible Faktenkette

Zwei weitere Bestandteile des Repräsentationsformalismus in MOBAL können die Verknüpfung mit nicht relevanten Literalen verhindern:

Beispiel 4.3 *Die folgenden Prädikatsdeklarationen stammen aus einer Sachbereichstheorie zum Lernen von Wahrnehmungsmerkmalen in der Robotik [Klingspor und Morik, 1995]:*

s_concave/5: !<trace>, !<sensor>, !<time>, <time>, !<orientation>
increasing/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
straight_away/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
no_measurement/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
decreasing/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
something_happened/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>.

Sowohl Sorten als auch funktionale Abhängigkeiten durch Angabe der Eingabeterme sind für die Argumente der Prädikate definiert. Die aktuellen Fakten der Theorie seien

straight_away(t31,236,s17,3,29,48)
decreasing(t31,296,s22,48,56,-28)
increasing(t30,212,s17,37,44,29)
no_measurement(t31,236,s17,29,57,999)
decreasing(t31,236,s17,57,86,-29)
something_happened(t31,236,s17,86,100,16) .

Gesucht ist die Faktenkette zum Beispiel

s_concave(t31,s17,3,86,diagonal)

der Relation s_concave. *Alle Fakten der Theorie sind mit dem Beispiel über gemeinsame Terme verbunden. Eine zu lernende Regel ist aber nur dann korrekt in bezug auf die Prädikatsdeklarationen, wenn identische Variablen nur an Argumentspositionen mit gleichen oder im Sortenverband in Beziehung stehenden Sorten auftreten. Zusätzlich müssen alle Eingabeargumente im Literal einer Regel stets bereits an anderer Stelle der Regel als Ausgabeargument oder als Eingabe im Kopf der Regel eingeführt worden seien, also mit dem Kopf verbunden seien. Dies gilt genauso für Faktenketten.*

Somit ist das zweite Faktum nicht kompatibel zu den definierten Eingabeargumenten, da mit s22 *ein Eingabeargument dieses Faktums nicht mit dem Beispiel verbunden ist. Der Eingabeterm* 48 *ist zwar mit dem ersten Faktum verbunden, ist dort aber ein Argument der Sorte* <grad>, *während es im Faktum* decreasing(t31,296,s22,48,56,-28) *ein Argument der Sorte* <time> *ist. Ebenso ist das dritte Faktum nicht kompatibel bezüglich der funktionalen Abhängigkeiten, da die Terme* t30 *und* 37 *der Eingabeargumente nicht verbunden sind.*

Die Fakten

straight_away(t31,236,s17,3,29,48)
no_measurement(t31,236,s17,29,57,999)
decreasing(t31,236,s17,57,86,-29)
something_happened(t31,236,s17,86,100,16)

sind dagegen auch unter Berücksichtigung ihrer Argumentssorten und funktionalen Abhängigkeiten kompatibel zum Beispiel. Das Beispiel hat als einzigen Ausgabeterm den Zeitpunkt 86. *Dieser wird bereits im vorletzten Faktum als Ausgabeterm gefunden und somit ist das letzte Faktum nicht relevant zur Beschreibung von* s_concave(t31,s17,3,86,diagonal).

Die mit der Prädikatsdeklaration kompatible Faktenkette dieses Beispiels ist die Regel

s_concave(t31,s17,3,86,diagonal) ←
    straight_away(t31,236,s17,3,29,48) &
    no_measurement(t31,236,s17,29,57,999) &
    decreasing(t31,236,s17,57,86,-29). ◇

Die Auswertung der Ausgabeargumente des Kopfes einer Faktenkette zur Reduzierung derselben ist nicht für alle Sachbereiche sinnvoll und ist daher über einen Parameter zu steuern. Sind für ein Prädikat keine Eingabeargumente definiert, muß nur die Sortenverträglichkeit einer beliebigen Verbindung gewährleistet seien. Die Tiefe eines Literals ist hier durch die Verbindung über Eingabeargumente zu definieren. Insbesondere haben die Ausgabeterme des Kopfliterals nicht mehr per Definition die Tiefe 0.

Definition 4.5 (i–kompatible Faktenkette)
Eine Faktenkette heißt **kompatibel** *zur Prädikatsdeklaration, wenn alle ihre Literale kompatibel sind. Ein Literal ist kompatibel, wenn alle Terme seiner Eingabeargumente kompatibel sind. Die Terme der Eingabeargumente des Kopfes der Faktenkette sind kompatibel mit einer Verbindungskette der Länge* 0*. Sind keine funktionalen Abhängigkeiten für den Kopf definiert, so sind alle Argumente des Kopfes Eingabeargumente. Ein Term in einem Literal l ist kompatibel mit einer Verbindungskette der Länge $d+1$, wenn für alle Terme $t_r (r \geq 1)$ der Eingabeargumente desselben Literals gilt: Es existiert ein Literal k in dem t_r kompatibel mit einer Verbindungskette der Länge d ist und die Sorte von t_r im Literal k ist bezüglich des Sortenverbandes kompatibel zu der Sorte von t_r im Literal l. Besitzt das Literal l keine Eingabeargumente, so muß dies für einen einzigen beliebigen Term aus l gelten. Die kompatible Tiefe eines Terms ist die minimale Länge seiner kompatiblen Verbindungsketten. Eine Faktenkette mit maximaler kompatibler Tiefe i ihrer Terme heißt* **i–kompatibel***.*

Im Beispiel 4.3 hat der Term 86 somit die kompatible Tiefe drei und die Faktenkette ist 3–kompatibel. Wie im Beispiel 4.3 angewandt, ist es oft sinnvoll, die i–kompatible Faktenkette mit minimalem Wert i zu konstruieren, so daß alle Ausgabeterme des Kopfes kompatibel mit einer Tiefe $j \leq i$ sind.

beispieldisjunkte Faktenkette

Oft kann man bei der Modellierung eines Sachbereichs die Fakten in zwei Gruppen trennen. Zu der einen Gruppe gehören neben den Beispielen der Zielrelation noch bestimmte Fakten die jeweils einem Beispiel direkt zugeordnet werden können. Es gibt dann einen Term in diesen Fakten, der nur in dem zugehörigen Beispiel vorkommt, nicht aber in anderen Beispielen. Zur zweiten Gruppe gehören alle Fakten, bei denen ein solcher Term nicht vorkommt.

Das folgende Beispiel ist ein Teil der Modellierung einer Endspielsituation beim Schach [Quinlan, 1983].

Beispiel 4.4 *Sei* $E = \{\mathsf{illegal}(457356), \mathsf{illegal}(502520)\}$ *und*

$$\begin{aligned} B = \{ \ & \mathsf{white\_king}(4,5,457356), \mathsf{white\_rook}(7,3,457356), \\ & \mathsf{black\_king}(5,6,457356), \mathsf{white\_king}(5,0,502520), \end{aligned}$$

$$\mathsf{white\_rook}(2,5,502520), \mathsf{black\_king}(2,0,502520),$$
$$\mathsf{adjacent}(0,1), \mathsf{adjacent}(2,3), \mathsf{adjacent}(3,4),$$
$$\mathsf{adjacent}(4,5), \mathsf{adjacent}(5,6), \mathsf{adjacent}(6,7)\}.$$

Die Fakten mit den Termen 457356 *und* 502520 *lassen sich eindeutig den Beispielen zuordnen. Dies gilt nicht für die Fakten mit dem Prädikatssymbol* adjacent. *Das Argument der Beispiele trennt hier also die Menge B in disjunkte Teilmengen.* $\diamond$

Lassen sich derartige Beobachtungen in einer Beispielmenge eines zu lernenden Begriffes anstellen, so ist eine weitere Einschränkung der Faktenketten sinnvoll.

Definition 4.6 (beispieldisjunkte Faktenkette)
Gegeben seien eine Sachbereichstheorie B und Beispiele $E \subseteq B$ der Zielrelation. Seien $arg(l)$ die Menge aller Terme eines Literals l und $ext(j, M)$ die Menge aller Terme einer Faktenmenge M für die gilt: $ext(j, M) = \{t \mid t \text{ ist ein Term an Stelle } j \text{ in } l \wedge l \in M\}$.
Eine Faktenkette $e \leftarrow l_1, \ldots, l_n$ heißt genau dann **beispieldisjunkt**, *wenn ein Index j existiert, für den $|E| = |ext(j, E)|$ gilt, und für alle Literale l_i, $1 \leq i \leq n$, gilt: $arg(l_i) \cap ext(j, E \setminus \{e\}) = \emptyset$.*

Der Effekt dieser Einschränkung ist unabhängig von einer bestimmten Hypothesensprache. Entsprechende Sachbereiche sind beispielsweise beim Lernen aus Datenbanken anzutreffen, da dort oft bestimmte Relationen ein Schlüsselattribut mit genau der in Definition 4.6 geforderten Eigenschaft besitzen.

Die beispieldisjunkten Faktenketten im obigen Beispiel sind dann:

$$\begin{aligned}
\mathsf{illegal}(457356) \leftarrow\ & \mathsf{white_king}(4,5,457356), \mathsf{white_rook}(7,3,457356),\\
& \mathsf{black_king}(5,6,457356), \mathsf{adjacent}(2,3), \mathsf{adjacent}(3,4),\\
& \mathsf{adjacent}(4,5), \mathsf{adjacent}(5,6), \mathsf{adjacent}(6,7)\\
\mathsf{illegal}(502520) \leftarrow\ & \mathsf{white_king}(5,0,502520), \mathsf{white_rook}(2,5,502520),\\
& \mathsf{black_king}(2,0,502520), \mathsf{adjacent}(0,1), \mathsf{adjacent}(2,3),\\
& \mathsf{adjacent}(3,4), \mathsf{adjacent}(4,5), \mathsf{adjacent}(5,6), \mathsf{adjacent}(6,7)
\end{aligned}$$

Darüberhinaus sind Situationen vorstellbar, in denen ein Index j mit der Eigenschaft $|E| = |ext(j, E)|$ aus Definition 4.6 nicht existiert, aber trotzdem nur solche Literale l_i einer Faktenkette $e \leftarrow l_1, \ldots, l_n$ relevant sind, die die Bedingung $arg(l_i) \cap ext(j, E \setminus \{e\}) = \emptyset$ für bestimmte Argumentstellen j von e erfüllen. Diese sind dann vom Benutzer vorzugeben. Die Beschränkung auf beispieldisjunkte Faktenketten muß auf jeden Fall über einen Parameter steuerbar bleiben.

In diesem Kapitel wurden verschiedene Einschränkungen zur Verknüpfung der Beispiele mit dem Hintergrundwissen vorgestellt. Diese Verknüpfung ist aufgrund der Verwendung der θ–Subsumtion als allgemeiner–als Relation in den weiteren Phasen dieses Ansatzes notwendig. Alle definierten Sprachen für Faktenketten und Reduzierungen auf relevante Teile der Sachbereichstheorie sind miteinander kombinierbar. Für verbundene, tiefenbeschränkte und determinierende Hypothesensprachen wurde bewiesen, daß sich diese Verknüpfung ohne Verlust der Vollständigkeit stark reduzieren läßt. Weitere Einschränkungen können bei Angabe von Argumentssorten und funktionalen Abhängigkeiten, Definition einer Prädikatstopologie und Auftreten von Schlüsselargumenten in den Beispielen vorgenommen werden.

4.2.2 Speziellste Generalisierung von Faktenketten

Der zweite Schritt des zu Beginn dieses Kapitels formulierten Ansatzes besteht in der Konstruktion der speziellsten Generalisierungen der Faktenketten, die durch die unterschiedlichen Einschränkungen aus dem vorherigen Abschnitt auf relevante Teilmengen reduziert worden sind.

Die Konstruktion speziellster Generalisierungen stellt eine Abstraktion der Faktenketten dar. Diese Generalisierungen abstrahieren von den Objekten der Faktenketten und ersetzen diese durch Termvariablen, wenn sie nicht in denselben Literalen der jeweiligen Faktenketten auftauchen. Diese Abstraktion soll auf alle $\frac{n(n-1)}{2}$ Paare von n Faktenketten oder als Heuristik auf eine bestimmte Anzahl x von Paaren für jede Faktenkette (insgesamt nx Paare) angewandt werden (siehe Abschnitt 4.3.6 zu den Parametern des neuen Werkzeuges MAT).

In Abschnitt 2.3 wurden bereits die beiden wichtigsten Verfahren zur Bildung speziellster Generalisierungen vorgestellt. Die logischen Verfahren nach Plotkin bilden die speziellsten Generalisierungen bezüglich der θ–Subsumtion und können exponentiell in der Anzahl der Beispiele wachsen. Die MSG zweier Faktenketten besitzt eine eindeutige Reduktion und ist im schlimmsten Fall quadratisch in der Größe der Faktenketten.

Die MSGs nach dem Graph Matching Verfahren von Haussler sind bezüglich $\vdash_\theta$ unvollständig und lassen sich als speziellste Generalisierungen unter Beachtung der Objektidentität (Definition 2.21) charakterisieren. Die speziellste Generalisierung ist bei diesem Ansatz nicht eindeutig. Im schlimmsten Fall ist die Anzahl der MSGs exponentiell in der Anzahl der Objekte der Beispiele. Dagegen sind diese MSGs maximal so groß wie die zu abstrahierenden Faktenketten.

Beide Methoden werden in diesem Ansatz alternativ verwendet. Als weitere Heuristik wird neben der Berechnung aller MSGs unter Beachtung der Objektidentität ein Algorithmus angeboten, der genau eine dieser speziellsten Generalisierungen berechnet, ohne die gesamte Menge aller MSGs als Zwischenergebnis zu konstruieren und dadurch sehr effizient ist (siehe auch Abschnitt 4.3.2 und Abschnitt 4.3.6).

Als weitere unvollständige MSG im Sinne von Plotkin ist folgender Ansatz zu betrachten: Die Definition von schwach determinierenden Faktenketten kann auf die speziellste Generalisierung von zwei termvariablenfreien Regeln, also auch Faktenketten, erweitert werden.

Definition 4.7 (schwach determinierende speziellste Generalisierung) *Sei h die speziellste Generalisierung zweier Grundklauseln $C1$ und $C2$. Seien θ_1, θ_2 zwei Substitutionen für die gilt: $h\theta_1 = C1 \wedge h\theta_2 = C2$. Eine* **speziellste Generalisierung** *h zweier Grundklauseln $C1$ und $C2$ heißt* **schwach determinierend** *bzgl. der Beispiele E und der Sachbereichstheorie B, wenn jeder Term aus h durch eine schwach determinierende Verbindungskette verbunden ist und h keine Grundliterale enthält. Ein Term, der im Klauselkopf vorkommt, ist durch eine schwach determinierende Verbindungskette der Länge 0 verbunden.*
Sei $h = A \leftarrow B_1 \,\&\, \ldots \,\&\, B_m \,\&\, B_{m+1} \,\&\, \ldots \,\&\, \neg B_n$ eine Regel mit einer Ordnung. Seien $\theta'_i \subseteq \theta_i \quad (i = 1, 2)$ die kleinsten Teilmengen von θ_i für die gilt: $\{A, \ldots, B_m\}\theta'_i \in B \cup E$. Ein Term t im Literal B_{m+1} ist mit einer schwach determinierenden Verbindungskette der Länge $d{+}1$ verbunden, gdw. alle Terme der Menge $\{A, B_1, \ldots, B_m\}$ mit einer schwach determinierenden Verbindungskette von höchstens der Länge d verbunden sind und es nur genau zwei Substitutionen $\sigma_i \quad (i = 1, 2)$ gibt, für die $\{B'_{m+1}\}\theta'_i\sigma_i \in B \cup E$ gilt. Die **schwach determinierende Tiefe eines Terms** *ist die minimale Länge seiner schwach determinierenden Verbindungsketten.*
Eine speziellste Generalisierung mit maximaler schwach determinierender Tiefe i ihrer Terme und maximaler Stelligkeit der Literale j heißt **ij–schwach determinierend**.

Beispiel 4.5 *Gegeben seien die Faktenketten*

$$\mathsf{c(a,b)} \leftarrow \mathsf{c(a,c)} \;\&\; \mathsf{p(b,d)} \;\&\; \mathsf{p(c,d)}$$
$$\mathsf{c(a,c)} \leftarrow \mathsf{c(a,b)} \;\&\; \mathsf{p(c,d)} \;\&\; \mathsf{p(b,d)}$$

Dann ist

$$\mathsf{c(a,X)} \leftarrow \mathsf{c(a,Y)} \;\&\; \mathsf{p(X,d)} \;\&\; \mathsf{p(b,d)} \;\&\; \mathsf{p(c,d)} \;\&\; \mathsf{p(Y,d)}$$

eine speziellste Generalisierung, aber die schwach determinierende Generalisierung ist

$$\mathsf{p(X,d)} \leftarrow \mathsf{c(a,X)}. \qquad \diamond$$

Damit steht eine weitere Alternative zur Bildung der speziellsten Generalisierungen der Faktenketten zur Verfügung. Plotkins Verfahren zur Konstruktion dieser Generalisierungen dient als Grundlage der in Abschnitt 4.3.2 vorzustellenden Implementierung dieser Ansätze und wurde bereits in Kapitel 2.3 beschrieben.

Reduktion der MSGs

Die so konstruierten MSGs von je zwei Faktenketten lassen sich weiter reduzieren. Analog zur Bildung der i–kompatiblen Faktenketten können Literale, die nicht zu den definierten funktionalen Abhängigkeiten kompatibel sind oder eine zu große Tiefe haben, entfernt werden. Dieses Vorgehen wird auch in anderen ILP Verfahren praktiziert (z.B. in GOLEM [Muggleton und Feng, 1992]).

Nicht zur Prädikatstopologie kompatible Literale wurden bereits in den Faktenketten entfernt. Durch einen Parameter kontrollierbar ist das Streichen von Grundliteralen in einer speziellsten Generalisierung. Dahinter steckt die Idee, daß konstante Terme in einer Regel nicht durch weitere Eigenschaften beschrieben werden müssen. So haben solche Grundliterale in einer Regel beispielsweise keine Auswirkung auf die Auswertung des Akzeptanzkriteriums von RDT.

Im Gegensatz zu diesen heuristischen Methoden ist die Entfernung redundanter Literale eine äquivalente Umformung. In Abschnitt 2.2.1 wurde bereits angeführt, daß diese Reduktionen nicht in polynomieller Zeit möglich sind. In Abschnitt 4.3.4 wird ein korrektes und oft effizientes Verfahren zur Berechnung der Reduktion einer Regel vorgestellt.

Eine weitere Möglichkeit den Suchraum zu beschneiden, bietet das Ableiten der klassifizierten Beispiele aus den berechneten Generalisierungen. Werden bezüglich des Akzeptanskriteriums von RDT zu viele Faktenketten negativer Beispiele von einer speziellsten Generalisierung H θ–subsumiert, braucht diese Regel H und ihre Generalisierungen nicht weiter betrachtet werden. Dieses Vorgehen ist aber auch heuristisch unter Beachtung der Aufgabenstellung, Regelmodelle für eine beliebiges Akzeptanzkriterium zu lernen. Zudem führt beim Test $H \vdash_\theta C$ auch eine Beschränkung der Größe der Faktenketten (also von $|C|$) auf kleinste Werte nicht zu einem in polynomieller Zeit berechenbaren θ–Subsumtionstest (siehe Abschnitt 2.2.1). Abschnitt 4.3.3 stellt aber einen für relevante Eingaben effizienten Algorithmus zur θ–Subsumtion vor, auf dem auch der bereits angesprochene verbesserte Reduktionsalgorithmus beruht [Kietz und Lübbe, 1994].

4.2.3 Abstraktion und Generalisierung der Regelmodelle

Nach der Konstruktion und Generalisierung der Faktenketten in den ersten beiden Schritten dieses Ansatzes erfolgt nun die Berechnung der gesuchten Regelmodelle auf Basis dieser speziellsten Generalisierungen.

Diese Regeln sind sehr speziell und decken nur eine geringe Anzahl von Beispielen ab. Da die akzeptierende Semantik von RDT aber wie in Abschnitt 3.3 beschrieben die allgemeinsten akzeptierbaren Regeln sucht, müssen die gesuchten Regelmodelle auch alle generelleren Regeln abdecken.

Eine zumindest für diesen letzten Schritt vollständige Methode basiert auf der Generalisierung durch Weglassen eines Literals: Bei allen Regeln wandelt man die Prädikatssymbole in Prädikatsvariablen um und bildet alle Teilmengen des Regelkörpers. Zusätzlich ist für jede Konstante in den Literalschemata dieser Regelmodelle dasselbe Schema mit einer Variable anstelle der Konstanten zu notieren.

Beispiel 4.6 *Aus der Generalisierung*

großmutter(X,Z) $\leftarrow$ mutter(X,Y) & mutter(Y,Z) & tochter(Z,Y)

erhält man so unter anderem die Schemata

Q(X,Z) $\leftarrow$ P(X,Y) & P(Y,Z) & R(Z,Y)
Q(X,Z) $\leftarrow$ P(X,Y) & P(Y,Z)
Q(X,Z) $\leftarrow$ P(Y,Z) & R(Z,Y)
Q(X,Z) $\leftarrow$ P(X,Y)
... $\diamond$

Für jede Generalisierung mit n Literalen gibt es somit 2^n verschiedene Regelmodelle allein durch Bildung der Teilmengen des Regelkörpers. Diese Zahl reduziert sich dadurch, daß ein Teil dieser Modelle nicht verbundene Literalschemata enthält oder redundant sind. Dennoch bleibt der durch die Ordnung über den Regelmodellen strukturierte Suchraum sehr groß und dieser Ansatz ist daher zu komplex.

Folgendes induktiv definiertes Verfahren versucht die Menge $\mathcal{R}$ der relevanten Regelmodelle für eine Zielrelation zu konstruieren (**Algorithmus 4.1**):

> Sei $\mathcal{G}_1$ die Menge der aus den Faktenketten berechneten speziellsten Generalisierungen. Diese Menge $\mathcal{G}_1$ wird als die Teilmenge der gesuchten Menge von Regelmodellen definiert, die die speziellsten Regelmodelle aus $\mathcal{R}$ enthält[1]. Sei $\mathcal{G}_{red_1}$ die Menge der nicht redundanten Regelmodelle aus $\mathcal{G}_1$ und $\mathcal{R}_1 = \mathcal{G}_{red_1}$ enthalte das momentane Zwischenergebnis.
>
> Analog zur Generalisierung der Faktenketten wird nun für eine Menge $\mathcal{G}_{red_{i-1}}$ die Menge $\mathcal{G}_i$ der speziellsten Generalisierungen von Paaren von Regelmodellen aus $\mathcal{G}_{red_{i-1}}$ berechnet. Anschließend wird die Menge $\mathcal{G}_{red_i}$ der bezüglich $\mathcal{R}_{i-1} \cup \mathcal{G}_i$ nicht redundanten Regelmodelle aus $\mathcal{G}_i$ konstruiert. Weiterhin sei $\mathcal{R}_i$ die Menge aller nicht redundanten Regelmodelle aus $\mathcal{R}_{i-1} \cup \mathcal{G}_i$.
>
> Dieses Vorgehen wird wiederholt, bis ein Index r mit $\mathcal{G}_{red_r} = \{\}$ existiert. Dann wird die Lösung $\mathcal{R}$ als $\mathcal{R} := \mathcal{R}_{r-1}$ definiert.

Kern dieses Verfahrens ist die Berechnung der speziellsten Generalisierungen von Regelmodellen bzgl. der allgemeiner–als–Relation $\vdash_{RS}^{lit\_inj}$ über diesen Modellen. Dazu wird die Konstruktion der MSGs unter Beachtung der Objektidentität eingesetzt. Zur Reduktion dieser Regelmodelle können sowohl nach Definition 3.11 redundante als auch Literalschemata mit zu großer Tiefe entfernt werden.

Die Heuristik dieses Verfahrens besteht darin, daß die Regelmodelle aus $\mathcal{G}_{red_i}$ mit wachsendem i immer genereller werden, durch die Konstruktion von speziellsten Generalisierungen aber innerhalb des durch die Relation $\vdash_{RS}^{lit\_inj}$ strukturierten Suchraumes ausgehend von den Generalisierungen der

[1] Dies ist möglich, da die Definition von Regelmodellen auch Regeln umfaßt.

Faktenketten mit geringer Schrittweite zu den generelleren Regelmodellen übergegangen wird.

Zur Veranschaulichung dieses Verfahrens diene Beispiel 4.7:

Beispiel 4.7 *Seien*

r_1:großmutter(X,Y) $\leftarrow$ vater(Z,Y) & mutter(X,Z)
r_2:großmutter(ann,Y) $\leftarrow$ vater(tom,Y) & mutter(ann,tom)
r_3:großmutter(ann,Y) $\leftarrow$ mutter(Z,Y) & mutter(ann,Z)

die speziellsten Generalisierungen $\mathcal{G}_1 = \mathcal{G}_{red_1} = \mathcal{R}_1$ einer Menge von Faktenketten. Dann bilden die Regelmodelle

$msg(r_1, r_2)$:großmutter(X,Y) $\leftarrow$ vater(Z,Y) & mutter(X,Z)
$msg(r_1, r_3)$:großmutter(X,Y) $\leftarrow$ P(Z,Y) & mutter(X,Z)
$msg(r_2, r_3)$:großmutter(ann,Y) $\leftarrow$ P(Z,Y) & mutter(ann,Z)

die Menge $\mathcal{G}_2$ der reduzierten Generalisierungen von $\mathcal{G}_{red_1}$ und $\mathcal{G}_{red_2} = \{msg(r_1, r_3), msg(r_2, r_3)\}$, da das Regelmodell $msg(r_1, r_2)$ redundant ist (wegen $msg(r_1, r_3)$). Das Zwischenergebnis ist: $\mathcal{R}_2 = \{r_2, msg(r_1, r_3), msg(r_2, r_3)\}$ (r_1 ist redundant wegen $msg(r_1, r_2)$, r_3 wegen $msg(r_2, r_3)$). In Stufe 3 wird dann aus $\mathcal{G}_{red_2}$ nur noch

großmutter(X,Y) $\leftarrow$ P(Z,Y) & mutter(X,Z)

berechnet. Dies Schema ist aber redundant in $\mathcal{R}_2$, so daß $\mathcal{G}_{red_3} = \{\}$. Somit ist $r = 3$ und $\mathcal{R} := \mathcal{R}_2$ mit den Modellen

großmutter(X,Y) $\leftarrow$ P(Z,Y) & mutter(X,Z)
großmutter(ann,Y) $\leftarrow$ P(Z,Y) & mutter(ann,Z)

nach Algorithmus 4.1 die Lösung für das Problem, Regelmodelle für die Relation großmutter *zu lernen.* $\diamond$

Wie obiges Beispiel deutlich macht, wird bei diesem Steigen in der durch $\vdash_{RS}^{lit\_inj}$ definierten Halbordnung nicht automatisch von den Prädikatssymbolen abstrahiert, sondern bleiben Objekt– und Prädikatsbindungen zum Teil erhalten.

Die Mengen $\mathcal{G}_{red_i}$ werden mit steigenden Werten i zunächst im Umfang wachsen und somit auch die Anzahl der zu konstruierenden Generalisierungen aller Paare dieser Mengen (diese Anzahl wächst sogar quadratisch zur Größe von $\mathcal{G}_{red_i}$). Da die Berechnung der MSGs aber aufwendig sein kann und insbesondere die anschließende Entfernung redundanter Regelmodelle auf einem *NP*–schwierigen Test beruht (siehe Abschnitt 3.2.2), steuert ein Parameter, wieviele Paare zu jedem Schema aus $\mathcal{G}_{red_i}$ zur Konstruktion neuer Generalisierungen herangezogen werden. Außerdem ist es möglich die Anzahl der Iterationen i nach oben zu begrenzen.

4.3 Das neue MAT

Der in Abschnitt 4.2 beschriebene heuristische Ansatz zum Lernen von Regelmodellen wurde als Erweiterung des schon beschriebenen Werkzeuges

MAT [Morik *et al.*, 1993, Abschnitt 6.5.3] von MOBAL implementiert. MOBAL ist in QUINTUS PROLOG geschrieben und läuft auf SUN Workstations unter dem Betriebssystem SUNOS4.1.

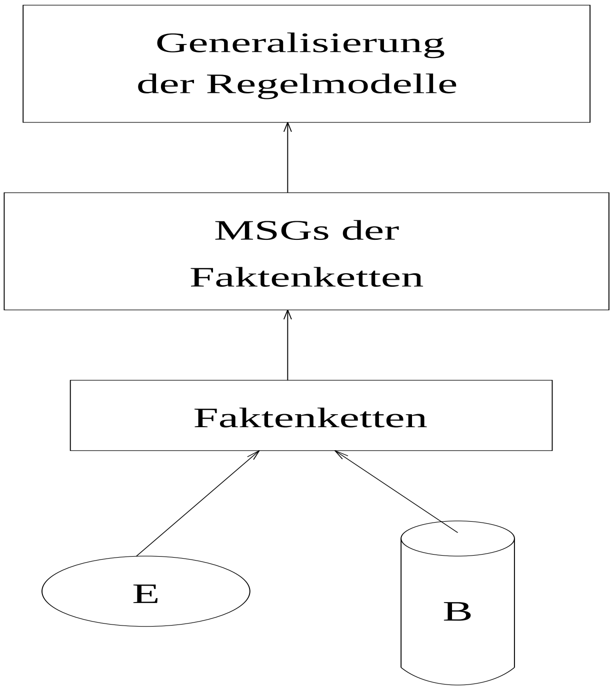

Abbildung 4.3: Der MAT Algorithmus im Überblick

Algorithmus 4.3 stellt den Ansatz im Überblick dar. Eingabe des Verfahrens sind die Sachbereichstheorie B und die Beispiele $E \subseteq B$ der Zielrelation. Aus diesen werden die Faktenketten der Beispiele konstruiert (siehe Abschnitt 4.3.1). Die Parameter **percentage_of_pos_examples** und **percentage_of_neg_examples** dieses Sytems bestimmen dabei den Prozentsatz der Beispiele, die MAT zum Lernen benutzt.

Im nächsten Schritt werden die speziellsten Generalisierungen der Faktenketten berechnet. Welche der alternativen Methoden zur Konstruktion dieser MSGs eingesetzt werden bestimmt der Parameter **factchain_lgg**. Algorithmen zur Konstruktion der MSGs werden in Abschnitt 4.3.2 und zur Reduktion in Abschnitt 4.3.4 dargestellt. Wie in Abschnitt 4.2.2 beschrieben, werden MSGs, die zu viele Faktenketten negativer Beispiele θ–subsumieren nicht weiter betrachtet. Ein effizienter Algorithmus zur θ–Subsumtion wird in Abschnitt 4.3.3 wiedergegeben.

Schließlich wird Algorithmus 4.1 zur iterativen Berechnung speziellster Generalisierungen von Regelmodellen ausgeführt. Die Parameter **max_number_of_pairs** und **max_number_of_mpred_generalisation_layers** steuern dabei die in Abschnitt 4.2.3 erwähnten Einschränkungen der Suchtiefe und –breite. Die Implementierungen der speziellsten Generalisierung, der Reduktion und der allgemeiner–als Test zwischen Regelmodellen werden in Abschnitt 4.3.5 beschrieben. Auf eine Darstellung der Implementierung von Algorithmus 4.1 unter Berücksichtigung der Parameter soll hier verzichtet werden.

Weitere Parameter des Systems regeln die Anzeige der Faktenketten und der MSGs während des Lernlaufes. Eine Übersicht über alle Parameter gibt Abschnitt 4.3.6.

4.3.1 Konstruktion der Faktenketten

Das Werkzeug MAT bildet die i–kompatiblen Faktenketten der Zielrelation in Kombination mit jeder beliebigen zusätzlichen Einschränkung mit Zeitaufwand polynomiell zur Größe der Sachbereichstheorie.

Satz 4.3 *Gegeben seien eine Sachbereichstheorie B und Beispiele E einer Zielrelation. Sei $t(h)$ das Maximum der Anzahl der Literale und der Anzahl aller Terme einer Faktenkette h. Die i–kompatible Faktenkette h eines Beispiels $e \in E$ kann mit Zeitaufwand in $O(i * |B| * t(h))$ berechnet werden.*

Beweis

Der Algorithmus 4.2 berechnet die ij–nicht determinierenden Faktenketten. Der Test < restriction > ist als die jeweils gewünschte zusätzliche Einschränkung der Verkettung zu implementieren (Algorithmus 4. 3). Für i–kompatible Faktenketten sind in < restriction > darüberhinaus die Sorten und funktionalen Abhängigkeiten von arg($Fakt$) zu überprüfen. Dies erfordert aber die Übergabe weiterer Argumente beim Aufruf von rec_fact_chain und soll daher zum besseren Verständnis hier nicht dargestellt werden.

Zum Beweis der Korrekheit für ij–nicht determinierende Faktenketten ist < restriction > durch **true** zu ersetzen. $ArgList$ enthält stets alle Terme, die in der Faktenkette eines Beispiels der Tiefe $i - Depth$ vorkommen. Beim ersten Aufruf von rec_fact_chain ist $Depth = i$ und $ArgList$ enthält die Terme der Tiefe 0, die Argumente des Beispiels $Example$. $NewArgList$ ist die Menge aller Terme der Tiefe $i - Depth + 1$. In der **forall** Schleife werden alle jene Fakten aus dem Hintergrundwissen B zur Liste $FactList$ hinzugefügt, die Terme aus $ArgList$ enthalten, also mit den Fakten der Tiefe $i - Depth$ verbunden sind. Dabei wird geprüft, ob diese Fakten nicht schon in $FactList$ vorhanden sind. Ist $Depth = 0$, also bei Tiefe i, so dürfen nur Fakten hinzugefügt werden, die keine neuen Terme einführen, daher ist der Schnittmengentest nun durch einen Teilmengentest zu ersetzen. Abbruchbedingung ist das Erreichen der maximalen Tiefe ($Depth = 0$) oder die Fakten der gerade betrachteten Tiefe führen keine neuen Terme mehr ein ($NewArgList = \emptyset$).

Für i–kompatible Faktenketten kommt als weiteres Abbruchkriterium das Finden aller Ausgabeargumente von $Example$ hinzu (ausschaltbar durch den Parameter **use_head_out_args**).

fact_chain($Depth, B, E, Example$)
  arg($Example$) := $ArgList$
  $KeyArgs$:=init_key_disjunct_fact(E)
  rec_fact_chain($Depth, B, ArgList, \{\}, KeyArgs$)

rec_fact_chain($Depth, B, ArgList, FactList, KeyArgs$)
  $NewArgList := \emptyset$
  for all $Fakt \in B$ **do**
    if (($Depth = 0$ **and** arg($Fakt$) $\subseteq ArgList$)
      or ($Depth \neq 0$ **and** arg($Fakt$) $\cap ArgList \neq \emptyset$))
      and $Fakt \notin FactList$
      and < restriction > = **true**
    then $NewArgList := NewArgList \cup$ arg($Fakt$)
      $FactList := FactList \cup \{Fakt\}$
  endfor
  $NewArgList := NewArgList \setminus ArgList$
  if $NewArgList = \emptyset$ **or** $Depth = 0$
  then return $FactList$
  else rec_fact_chain($Depth - 1, B, NewArgList, FactList, KeyArgs$)

Algorithmus 4.2: Konstruktion von Faktenketten

Bei den schwach determinierenden Faktenketten muß in < restriction > nach Definition 4.4 nur getestet werden, ob ein Faktum neue Terme der Tiefe $i+1$ einführt, so daß, umgewandelt in eine Termvariable, deren Belegung durch die Terme der maximalen Tiefe i dieses Faktums eindeutig festgelegt ist. Dieses testet die Funktion weak_determinate_fact.

Für beispieldisjunkte Faktenketten testet die Funktion key_disjunkt_fact, ob eine neues Faktum l einer Faktenkette die Forderung $arg(l_i) \cap ext(j, E \setminus \{e\}) = \emptyset$ aus Definition 4.6 erfüllt. Dabei ist $KeyArgs = ext(j, E \setminus \{e\})$ und wird genau dann in init_key_disjunkt_fact initialisiert, falls es eine Menge $T_i = ext(i, E)$ gibt, so daß $|T_i| = |E|$ gilt, wie in Definition 4.6 gefordert.

Die Funktion rec_fact_chain wird $O(min\{Depth, |B|\})$ oft rekursiv aufgerufen, wobei nur Werte mit $Depth < |B|$ sinnvoll sind. Jeder Aufruf enthält $|B|$ viele Durchläufe der **forall** Schleife. Diese sind nach oben durch $t(h)$ und die Kosten von < restriction > beschränkt.

Bei einer ij–nicht determinierenden Faktenkette ist $Depth = i$ zu setzen. Nicht determinierende Faktenketten können daher in $O(i * |B| * t(h))$ konstruiert werden.
Die nicht dargestellte Überprüfung der Sorten und funktionalen Abhängigkeiten ist durch ein Polynom in der Größe von $t(h)$ nach oben begrenzt, so daß der Aufwand für i–kompatible Faktenketten in derselben Größenord-

weak_determinate_fact($Fakt, ArgList, B$)
    bilde f, indem in $Fakt$ alle Terme aus arg($Fakt$)$\backslash ArgList$ durch
        Variablen ersetzt werden
    if $\exists f' \in B$ mit $\exists$ Substitution $\sigma : f\sigma = f' \wedge f' \neq Fakt$
    then return **false**
    else return **true**

key_disjunct_fact($Fakt, KeyArgs$)
    if arg$Fakt \cap KeyArgs = \emptyset$
    then return **true**
    else return **false**

init_key_disjunct_fact($E, Example$)
    bestimme die Mengen $T_1, \ldots, T_s$, der Terme die in den Beispielen E
        an Position $i, 1 \leq i \leq s$ vorkommen
    finde Menge T_i mit $|T_i| = |E|$
    return $KeyArgs := T_i \setminus$ nth_arg($Fakt, i$)

Algorithmus 4.3: Implementierungen von < restriction >

nung liegt.
Eine ij–schwach determinierende Faktenkette kann aus der ij–nicht determinierenden Faktenkette h desselben Beispiels in $O(t(h)^2)$ berechnet werden, somit liegt die Komplexität zur Berechnung schwach determinierender Faktenketten in $O(i * |B| * t(h)^2)$. Der Test weak_determinate_fact berechnet dagegen direkt nur die ij–schwach determinierende Faktenkette eines Beispiels. Er testet im schlimmsten Fall zwar $|B|$ viele Unifikationen für jedes verbundene Literal, da die Literale f' in diesem Test aber teilweise instanziiert sind, kann bei der PROLOG Implementierung die Indexierung ausgenutzt werden und dieser Aufwand reduziert sich daher erheblich.
Die Initialisierung und der Test für beispieldisjunkte Faktenketten liegen in $O(|E|)$. Für beispieldisjunkte Faktenketten ergibt sich damit eine Komplexität von $O(i * |B| * |E|)$. □

4.3.2 Konstruktion speziellster Generalisierungen

Die speziellsten Generalisierungen bezüglich der θ–Subsumtion werden nach dem Algorithmus 2.1 von Plotkin konstruiert. Die Konstruktion der MSGs unter Beachtung der Objekidentität unterscheidet sich von diesem Verfahren in der Konstruktion der Selektionen. Während Plotkin nur eine Liste S von Selektionen berechnet, gilt es für diese MSGs alle unterschiedlichen, maximalen Listen S_i von Selektionen zu berechnen, so daß für alle Selektionen $[p(t_1, \ldots, t_r) - p(u_1, \ldots, u_r)]$ einer Liste S_i gilt: Es gibt kein Paar t_j, u_j so daß Paare $t_j, u \ \wedge u \neq u_j$ oder $t, u_j \ \wedge t \neq t_j$ in einer anderen Selektion dieser Liste S_i auftreten. Dabei ist in der Implementierung zu verhindern, daß Permutationen der Listen S_i konstruiert werden. Auf alle diese Listen S_i wendet man dann Plotkins Generalisierung von Wörtern an. Ergebnis ist eine Menge von Regeln. Die speziellsten Regeln aus dieser Menge bilden die MSGs nach Haussler. Auf die Details der Implementierung soll auch hier verzichtet werden.

4.3.3 Ein effizienter Algorithmus zur θ–Subsumtion

In diesem Abschnitt wird der im System MAT verwendete effiziente Algorithmus zur θ–Subsumtion zwischen Generalisierungen und Faktenketten beschrieben. Dieser Algorithmus und die auf denselben Ideen beruhende Verbesserung des Reduktionsalgorithmus ist eine gemeinsame Arbeit mit Jörg–Uwe Kietz [Kietz und Lübbe, 1994].

Ursache der Ineffizienz von $\vdash_\theta$ ist der diesem Test inhärente Nichtdeterminismus bei der Wahl der Substitution θ. In Kapitel 2.2.3 wurde bereits die Sprache der determinierenden Literale [Muggleton und Feng, 1992] für Hypothesen (und damit D) vorgestellt, deren Grundsubstitutionen durch die Beispiele eindeutig festgelegt sind, also auch durch die Faktenketten (C). Tatsächlich ist diese Eigenschaft auf den Test $\vdash_\theta$ für determinierende Hornklauseln übertragbar, so daß $\vdash_\theta$ für derartige Hornklauseln in polynomieller Zeit berechenbar ist [Kietz und Lübbe, 1994]:

Definition 4.8 (deterministische θ–Subsumtion, $\vdash_{\theta DET}$) *Seien $D = D_0 \leftarrow D_{Body}$ und $C = C_0 \leftarrow C_{Body}$ zwei Hornklauseln. D* **θ–subsumiert** *C genau dann* **deterministisch** *mit der Substitution $\theta = \theta_0\theta_1 \ldots \theta_n$, notiert als $D \vdash_{\theta DET} C$, wenn $D_0\theta_0 = C_0$ gilt und eine Ordnung $D'_{Body} = D_1, \ldots, D_n$ von D_{Body} existiert, so daß für alle $i, 1 \leq i \leq n$, genau eine Substitution θ_i mit $\{D_1, \ldots, D_i\}\theta_0\theta_1 \ldots \theta_i \subseteq C_{Body}$ existiert.*

In [Kietz und Lübbe, 1994] wird bewiesen, daß $D \vdash_{\theta DET} C$ mit polynomieller Komplexität berechnet und die θ–Subsumtion zwischen determinierenden Hornklauseln auf diese Subsumtion zurückgeführt werden kann und damit in polynomieller Zeit berechenbar ist.

Der in [Kietz und Lübbe, 1994] beschriebene Algorithmus stellt auch für generelle Hornklauseln D eine Verbesserung dar, da er beim Testen von $D \vdash_{\theta DET} C$ auch den nicht determinierenden Teil $D_{NONDET} = D \setminus D_{DET}$ von D berechnet, falls $D_{DET} \vdash_{\theta DET} C$ gilt. Danach gilt $D \vdash_\theta C$ genau dann, wenn $D_{DET} \vdash_{\theta DET} C$ mit θ_{DET} und $D_{NONDET}\theta_{DET} \vdash_\theta C$ gelten. Die Komplexität, D in D_{DET} und D_{NONDET} zu teilen und $D_{DET} \vdash_{\theta DET} C$ zu testen, liegt in $O(|D_{DET}| * |D| * |C|)$.

Der verbleibende Test $D_{NONDET}\theta_{DET} \vdash_\theta C$ benötigt im worst case natürlich weiterhin exponentielle Rechenzeit. Die θ–Subsumtion zwischen beliebigen Hornklauseln läßt sich so in einen determinierenden und einen nicht determinierenden Anteil teilen.

Der verbliebene Subsumtionstest des nicht determinierenden, unter Umständen großen Teils D_{NONDET} einer Hornklausel $D = D_0 \leftarrow D_{DET}, D_{NONDET}$ kann auch weiter verbessert werden. Dieser Teil der Klausel läßt sich in solche Mengen aufteilen, die bezüglich jeder möglichen Substitution θ im Test $\vdash_\theta$ unabhängig voneinander sind. Als Resultat ergeben sich dann ebenso unabhängige Subsumtionstests zwischen disjunkten Teilmengen von D_{NONDET}.

Definition 4.9 (k–lokale Hornklausel)
Sei $D = D_0 \leftarrow D_{DET}, D_{NONDET}$ eine Hornklausel. Die Funktion vars berechne die Menge aller Variablen einer Formel. $LOC_i \subseteq D_{NONDET}$ heißt eine nicht–determinierende **Lokalität** *von D, wenn $(vars(LOC_i) \setminus vars(\{D_0, D_{DET}\})) \cap vars(D_{NONDET} \setminus LOC_i) = \emptyset$ gilt und keine echte, nicht leere Teilmenge $LOC_j \supset LOC_i$ existiert, die auch eine nicht–determinierende Lokalität von D ist. Eine nicht–determinierende Lokalität LOC_i heißt k–Lokalität für eine Konstante k, wenn $k \geq min(|(vars(LOC_i) \setminus vars(\{D_0, D_{DET}\})|, |LOC_i|)$ gilt. Eine Klausel heißt* **k–lokal***, wenn jede nicht–determinierende Lokalität k–lokal ist.*

Beschränkt man den nicht determinierenden Teil einer beliebigen Hornklausel derart, so ergibt sich für kleine Werte von k ein effizienter Algorithmus für k–lokale Hornklauseln. Der in [Kietz und Lübbe, 1994] vorgestellte Algorithmus verbessert den Test $\vdash_\theta$ zudem für beliebige Hornklauseln. da er diese in möglichst kleine disjunkte Teilmengen aufteilt, die unabhängig voneinander getestet werden können.

Einen θ–Subsumtions Algorithmus, der eine Aufteilung der Klauseln nach Lokalitäten vornimmt, wurde bereits von Gottlob und Leitsch beschrieben [Gottlob und Leitsch, 1985], jedoch nicht in Kombination mit dem Begriff der determinierenden Klauseln.

4.3.4 Ein verbesserter Algorithmus zur Reduktion von Hornklauseln

Die bereits angesprochene Verbesserung der Reduktion (siehe Kapitel 4.2.2) von Hornklauseln basiert auf denselben Ideen, wie der verbesserte Algorithmus für die θ–Subsumtion. Die Reduktion einer Klausel D benötigt mindestens $|D|$ viele Aufrufe eines NP Orakels für $\vdash_\theta$. Die Identifikation determinierender und lokaler Teilmengen der Klausel D führt zu teilweise unabhängigen, kleineren Reduktionen. Zuerst wird die maximale Teilmenge D_{DET} von D bestimmt, die D deterministisch θ–subsumiert. Der folgende Satz sagt aus, daß die so festgelegte Teilklausel D_{DET} keine redundanten Literale enthält. Somit ist jede Hornklausel, die sich selber deterministisch θ–subsumiert, reduziert. Darüber hinaus muß nach diesem Satz nur der nicht determinierende Teil der Hornklausel D reduziert werden.

Satz 4.4 *Sei $D = D_0 \leftarrow D_{DET}, D_{NONDET}$ eine Hornklausel, so daß $D_0 \leftarrow D_{DET}$ die maximale Teilmenge von D ist, für die $(D_0 \leftarrow D_{DET}) \vdash_{\theta DET} D$ gilt. Dann ist die Hornklausel*

$$D_0 \leftarrow D_{DET}, D^R_{NONDET}$$

die Reduktion von D, wobei D^R_{NONDET} die minimale Teilmenge von D_{NONDET} ist, für die gilt:

$$D \vdash_\theta (D_0 \leftarrow D_{DET}, D^R_{NONDET}).$$

Der Begriff der Lokalität kann zweimal zur Verbesserung der Reduktion ausgenutzt werden. Zum einen kann eine Lokalität LOC_i der Teilmenge D_{NONDET} einer Hornklausel D komplett entfernt werden, wenn $LOC_i \vdash_\theta (D \setminus LOC_i)$ gilt, d.h. die ganze Lokalität LOC_i ist redundant. Sind nur Teile einer Lokalität LOC_i redundant, so kann diese dennoch unabhängig von anderen Lokalitäten reduziert werden, indem die minimale Teilmenge $LOC_i^R \subseteq LOC_i$ berechnet wird, so daß $LOC_i \vdash_\theta (LOC_i^R \cup D_{DET} \cup \bigcup^j_{j \neq i} LOC_j)$ gilt. Unter Ausnutzung dieser Eigenschaften ergibt sich für die Reduktion von generellen Hornklauseln folgende Komplexität.

Satz 4.5 *Sei $D = D_0 \leftarrow D_{DET}, LOC_1, \ldots, LOC_n$ eine Hornklausel, wobei $D_0 \leftarrow D_{DET}$ die Hornklausel D deterministisch θ–subsumiert und alle LOC_i k–Lokalitäten von D_{NONDET} sind. Die Reduktion D^R von D kann mit $O(|D_{DET}| * |D|^2 + |LOC_1, \ldots, LOC_n|^2 + n * (k^{k+1} * |D|))$ Unifikationsversuchen berechnet werden.*

Auch dieser Algorithmus ruft, wie der von Gottlob und Leitsch [Gottlob und Leitsch, 1985], linear in $|D_{NONDET}|$ oft eine Prozedur für die *NP*–vollständige θ–Subsumtion auf. Allerdings sind die beteiligten Teilklauseln durch die Substituierung der determinierenden Variablen hier kleiner als in [Gottlob und Leitsch, 1985]. Dadurch sind die Kosten der Aufrufe von θ geringer. Natürlich hat auch dieser Algorithmus wie die θ–Subsumtion im schlechtesten Fall eine exponentielle Komplexität von $O(|D|^{|D|})$, falls D aus einer einzigen Lokalität besteht.

4.3.5 Effiziente Algorithmen zur Generalisierung, Subsumtion und Redundanz von Regelmodellen

Die Algorithmen zur Konstruktion der speziellsten Generalisierung von zwei Regeln können auch für Regelmodelle eingesetzt werden. Dazu bildet man jedes positive n–stellige Literalschema $P(t_1, \ldots, t_n)$ eines Regelmodells auf das $n + 1$–stellige Literal $pos(P, t_1, \ldots, t_n)$ ab. Negative Literalschemata $not(P(t_1, \ldots, t_n))$ werden auf $neg(P, t_1, \ldots, t_n)$ abgebildet. Als Resultat dieser Abbildung erhält man Regeln, auf die dann die in Abschnitt 4.3.2 dargestellten Algorithmen zur Berechnung der MSGs bzgl. $\vdash_{\theta_{inj}}$ angewandt werden können. Die resultierenden Regelmodelle sind zum Teil genereller als die speziellsten Generalisierungen bezüglich der Relation $\vdash_{RS}^{lit\_inj}$. Die Beachtung der Objektidentität auch für die Prädikatsvariablen verhindert, daß die konstruierten Regelmodelle spezieller als die zu generalisierenden Regelmodelle sind. Anschließend ist die beschriebene Abbildung der Literalschemata auf Literale wieder zu invertieren.

Der Test $R \vdash_{RS}^{lit\_inj} R'$ zwischen zwei Regelmodellen kann auf Grundlage von Satz 3.1 implementiert werden. Dazu gilt es zuerst die Reduktionen R_{red} und R'_{red} dieser Regelmodelle bezüglich der θ–Subsumtion zu konstruieren. Anschließend ist zu testen, ob eine injektive Abbildung Σ über der Menge der Literalschemata existiert, so daß $R_{red}\Sigma\ddot{\sigma} \subseteq R'_{red}$ Dazu werden die Regelmodelle zunächst gemäß obiger Abbildung zu Regeln umgeformt. Der Test $R_{red}\Sigma\ddot{\sigma} \subseteq R'_{red}$ kann dann mit dem effizienten Algorithmus für die θ–Subsumtion zwischen Regeln aus Abschnitt 4.3.3 ausgewertet werden, wobei man zusätzlich sicherstellt, daß in R_{red} durch Σ keine Literale unifiziert werden.

Da Regelmodelle nur mit $\Sigma = \emptyset$ bzgl. $\vdash_{RS}^{lit\_inj}$ reduziert werden können, kann zu diesem Zweck der Reduktionsalgorithmus aus Abschnitt 4.3.4 für Regeln zum Einsatz kommen. Dazu bildet man das Regelmodell wieder auf eine Regel ab, wobei die Prädikatsvariablen durch Konstanten ersetzt werden, die sonst nicht in dem Modell vorkommen. Nach der Reduktion dieser Regel bzgl. $\vdash_\theta$ macht man diese Abbildung wieder rückgängig.

Die Redundanz eines Regelmodelles R in einer Menge $\mathcal{R}$ von Regelmodellen schließlich wird mit Hilfe des in Satz 3.2 beschriebenen Verfahrens getestet: Gesucht wird ein Modell $R' \in \mathcal{R}$ mit der Reduktion R'_{red}, so daß

- $|R'_{red}| = |R_{red}|$, wobei R_{red} die Reduktion von R ist und
- $R'_{red} \vdash_{RS}^{lit\_inj} R_{red}$ mit $R'_{red}\Sigma\ddot{\theta} \subseteq R_{red}$, wobei Bild($\Sigma$) und Bild($\ddot{\theta}$) keine Konstanten enthalten und $\ddot{\theta}$ eine injektive Abbildung ist.

Für den letzten Test wird beim oben beschriebenen modifizierten Algorithmus für die θ–Subsumtion zusätzlich überprüft, ob in R'_{red} (Pradikats-)Variablen auf Konstanten abgebildet und verschiedene Variablen aus R'_{red} unifiziert werden .

4.3.6 Mats Parameter

Dieser Abschnitt stellt die Parameter des Werkzeuges Mat in einer Übersicht dar:

recursive_definition *(advanced)*
Mögliche Werte: oneof([yes,no])
Default: no
Beschreibung: Kontrolliert das Auftreten des Kopfprädikates im Körper der Faktenketten (und damit auch der Regelmodelle).

use_negated_literals *(advanced)*
Mögliche Werte: oneof([yes,no])
Default: yes
Beschreibung: Kontrolliert das Auftreten negierter Literalschemata im Körper der Regelmodelle.

max_linking_depth *(basic)*
Mögliche Werte: oneof([unlimited,0,1,2,3,4])
Default: 1
Beschreibung: Begrenzt die Tiefe der Verbindungsketten der Regelmodelle.

saturation_restriction *(basic)*
Mögliche Werte: oneof([indeterminate_facts,weak_determinate_facts])
Default: indeterminate_facts
Beschreibung: Kontrolliert die Einschränkung auf schwach determinierende Faktenketten.

user_sort_restriction *(basic)*
Mögliche Werte: oneof([yes,no])
Default: no
Beschreibung: Kontrolliert die Verwendung der Benutzersorten zur Beschränkung der Faktenketten.

use_head_out_args *(advanced)*
Mögliche Werte: oneof([yes,no])
Default: no
Beschreibung: Kontrolliert die Verwendung der Ausgabeargumente des Kopfliterals zur Beschränkung der Faktenketten.

arg_extention_disjunct *(basic)*
Mögliche Werte: oneof([no,auto,user_given])
Default: no
Beschreibung: Kontrolliert die Einschränkung auf beispieldisjunkte Faktenketten. Ist der Wert "auto", so wird entsprechend der Definition versucht einen einstelligen Schlüssel in den Beispielen zu finden. Ist der Wert "user_given" wird der Benutzer um Angabe einer Liste von Argumentspositionen gebeten, auf die dann diese Definition angewandt wird.

percentage_of_pos_examples *(basic)*
Mögliche Werte: unrestricted_or(integer_from_to(1,100))
Default: 30
Beschreibung: Der Prozentsatz der positiven Beispiele die MAT *zum Lernen benutzt.*

percentage_of_neg_examples *(basic)*
Mögliche Werte: unrestricted_or(integer_from_to(0,100))
Default: 30
Beschreibung: Der Prozentsatz der negativen Beispiele die MAT *zum Entfernen zu allgemeiner MSGs der Faktenketten benutzt.*

print_factchains *(advanced)*
Mögliche Werte: oneof([yes,no])
Default: yes
Beschreibung: Kontrolliert die Ausgabe der Faktenketten während des Lernlaufes.

factchain_lgg *(advanced)*
Mögliche Werte: oneof([one_oi_lgg,all_oi_lgg,weak_det_lgg,plotkin])
Default: all_oi_lgg
Beschreibung: Kontrolliert die Konstruktion der MSGs der Faktenketten. "one_oi_lgg" konstruiert genau eine, "all_oi_lgg" alle MSGs unter Beachtung der Objektidentität. "plotkin" konstruiert die MSG bzgl. $\vdash_\theta$.

remove_ground_literals *(advanced)*
Mögliche Werte: oneof([yes,no])
Default: no
Beschreibung: Kontrolliert das Entfernen von Grundliteralen zur Reduzierung der MSGs.

mpred_lgg *(advanced)*
Mögliche Werte: oneof([one_oi_lgg,all_oi_lgg,plotkin])
Default: one_oi_lgg
Beschreibung: Kontrolliert die Konstruktion der MSGs der Regelmodelle. "one_oi_lgg" konstruiert genau eine, "all_oi_lgg" alle MSGs unter Beachtung der Objektidentität.

max_number_of_pairs *(basic)*
Mögliche Werte: unrestricted_or(integer_ge(1))
Default: 5
Beschreibung: Suchbreite, d.h. dieser Wert gibt an, wie oft ein Regelmodell maximal in einer Iteration zur Konstruktion einer MSG benutzt wird. "unrestricted" bedeutet, daß alle möglichen Paare benutzt werden.

max_number_of_mpred_generalisation_layers *(basic)*
Mögliche Werte: unrestricted_or(integer_ge(1))
Default: 10
Beschreibung: Suchtiefe, d.h. die maximale Anzahl der Iterationen in Algorithmus 4.1. "unrestricted" bedeutet, daß der Algorithmus endet, wenn keine

neuen Regelmodelle mehr gefunden werden.

print_mpred_layer *(advanced)*
Mögliche Werte: oneof([yes,no])
Default: yes
Beschreibung: Kontrolliert die Ausgabe der jeweils neuen Regelmodelle in einer Iteration.

Der Parameter max_no_of_exceptions des Werkzeuges RDT wird zur Entfernung von zu generellen MSGs der Faktenketten benutzt. Der Parameter topology_restrictions dieses Werkzeuges entscheidet über die Beachtung der Prädikatstopologie bei der Konstruktion der Faktenketten.

4.4 Verwandte Ansätze

Das Werkzeug INCY des Systems MOBAL benutzt einen heuristischen, datengesteuerten Algorithmus zum Lernen von Regeln [Sommer, 1993]. Ausgehend von einem Beispiel spezialisiert INCY Hypothesen durch Hinzunahme von Fakten aus Mengen von Fakten, die mit einem der Objekte des Beispiels einen gemeinsamen Term teilen. Diese Klauseln werden zu Regelmodellen abstrahiert und analog zum Werkzeug RDT mögliche Instanzen auf Akzeptanz getestet. Es werden also sowohl Regeln wie auch Regelmodelle gelernt, die anderen Lernverfahren zur Verfügung gestellt werden. Im Gegensatz zu dem hier vorgestellten Verfahren, beschränkt sich dieser Ansatz nicht auf das Lernen von syntaktischen Vorgaben, sondern testet Instanzen des Suchraumes und benötigt daher ein konkretes Akzeptanzkriterium.

Feng und Muggleton definieren einen eingeschränkten logischen Formalismus höherer Ordnung, in welchem sie induktive Generalisierungen berechnen [Feng und Muggleton, 1992]. Ihre Sprache basiert auf dem logischen λ–Kalkül. Sie zeigen, daß Terme dieser Sprache eine eindeutige speziellste Generalisierung besitzen und geben einen Algorithmus zur Konstruktion dieser Generalisierungen an. Als Anwendung präsentieren sie die Generalisierung zweier transitiver Klauseln für die Begriffe vorgänger und weniger_als und erhalten die speziellste Generalisierung höherer Ordnung:

$$\forall \mathsf{XYZ.P(X,Y)} \leftarrow \mathsf{P(X,Z)} \ \ \& \ \mathsf{P(Z,Y)}.$$

Der Nachweis zum Einsatz dieses Verfahrens für komplexere Probleme mit nicht bekannten Lösungen fehlt.

4.5 Zusammenfassung

Nachdem im Kapitel 3 ein modellbasiertes Lernverfahren präsentiert und einige der theoretischen Grundlagen neu definiert wurden, stellte dieses Kapitel einen heuristischen Ansatz zum Lernen von Modellwissen in Form von

Regelschemata dar. Dadurch soll das fehlende Werkzeug zum automatischen Erwerb von Modellwissen bereitgestellt werden.

Der Ansatz basiert auf der Idee, daß die Verknüpfung der Fakten einer Sachbereichstheorie über gemeinsame Terme die syntaktische Struktur der zu entdeckenden Regeln beschreibt. Die Verknüpfung wird durch die Konstruktion einer Faktenkette eines Beispiels verwirklicht, wobei verschiedene Einschränkungen ein zu großes Ausmaß der Faktenketten verhindern. Der Einfluß der Einschränkungen auf das Lernergebnis läßt sich teilweise formal beschreiben. Die Faktenketten halten bereits Regelhaftigkeiten in den Daten fest, allerdings beschreiben sie nur genau ein Beispiel der Zielrelation.

Die speziellste Generalisierung dieser Faktenketten erlaubt das Steigen in der Allgemeinheitsordnung mit möglichst geringer Schrittweite. Anstatt alle Generalisierungen dieser Faktenketten im Raum der Regelmodelle zu konstruieren, werden speziellste Generalisierungen von einer steigenden Anzahl von Regelmodellen berechnet, bis keine neuen syntaktische Schablonen mehr gefunden werden. Dabei kommen auch bezüglich der allgemeiner–als–Relation nicht vollständige speziellste Generalisierungen zum Einsatz.

Das Verfahren setzt kein konkretes Kriterium für die Akzeptanz von Regeln oder Regelmodellen voraus und hat daher wenig Möglichkeiten den Suchraum gezielt einzugrenzen oder eine Menge von Regelmodellen als bereits vollständige Lösung zu identifizieren.

Kapitel 5

Experimente und Resultate

In diesem Kapitel wird die Eignung des heuristischen, datengesteuerten Ansatzes des Werkzeugs MAT zum Lernen von Regelmodellen experimentell überprüft. Dabei sollen Regelmodelle für Zielrelationen aus verschiedenen Sachbereichstheorien erworben werden. Diese Theorien sind unterschiedlich komplex und benötigen daher jeweils andere Einstellungen der Parameter von MAT.

Zu Beginn wird der Sachbereich TRAFFIC-LAW zur Darstellung von Verkehrsverstößen und deren gesetzlichen Folgen in Deutschland vorgestellt [Morik *et al.*, 1993, Kapitel 9.1]. Die in diesem Sachbereich modellierten Ereignisse sind hinreichend klein und lassen so ein positives Ergebnis erwarten.

Weitaus komplexer sind die Lernziele zweier Sachbereichstheorien aus dem Projekt BLEARN II [Klingspor und Morik, 1995]. Dieses Projekt hat die Navigation eines mobilen Roboters zum Inhalt.

Sicherheitsprobleme in der Telekommunikation werden in SPEED behandelt, einer Anwendung von MOBAL in Kooperation mit Alcatel Alshtom Recherche, Paris [Sommer *et al.*, 1994].

Bevor diese Sachbereiche detaillierter beschrieben und das Verfahren MAT zum Lernen von Regelmodellen für einzelne Zielrelationen in diesen Theorien eingesetzt wird, soll die Bewertung der Resultate diskutiert werden. Ein mögliches Kriterium ist das Zurückgreifen auf bereits durchgeführte Lernläufe mit dem modellbasierten Verfahren RDT. Anhaltspunkt ist hierbei, ob MAT die Regelmodelle lernt, die dem Werkzeug RDT bei erfolgreichen Versuchen vorgegeben wurden und wieviele überflüssige Regelmodelle dieses Verfahren dabei erzeugt.

Eine weitere Möglichkeit bietet die Überprüfung der gelernten Menge von Regelmodellen durch das Werkzeug RDT bei geeigneten Akzeptanzkriterien. Außerdem gilt es festzustellen, ob die Faktenketten der Beispiele genau die relevanten Fakten der Sachbereichstheorie enthalten.

Den Abschluß dieses Kapitels bildet eine zusammenfassende Bewertung der Experimente.

5.1 Traffic-Law

Der Sachbereich Traffic-Law ist mit dem System Mobal erhältlich und ist in [Morik *et al.*, 1993, Kapitel 9.1] ausführlich beschrieben. Da er zur Illustration geeignet ist und aufgrund seiner Größe positive Ergebnisse beim Lernen mit Mat ohne zu starke Beschränkung durch die Parameter erwarten läßt, soll er in dieser Arbeit als erster Test dienen.

Er beschreibt insgesamt 11 Fälle von Verstößen gegen die Verkehrsordnung. Die Prädikate beschreiben, wer verantwortlich für den Verstoß ist und welche rechtlichen Folgen dieser nach sich zieht. Insgesamt gibt es ca. 30 Prädikate, über 100 Fakten und eine Reihe von Regeln. Die Prädikate sind in einer Topologie angeordnet.

Zum Lernen von Regelhaftigkeiten eignet sich die Relation responsible. Diese zweistellige Relation beschreibt, welche Person für ein Ereignis verantwortlich ist. Ohne weitere Einschränkungen zu benutzen, konstruiert Mat fur das Beispiel responsible(ab,event6) die Faktenkette:

responsible(sw,event1) ←
involved_vehicle(event1,b_au_6773) &
owner(sw,b_au_6773) &
car_parked(event1,place1) &
car_towed(event1,b_au_6773) &
fine(event1,20) &
bus_lane(place1) &
sedan(b_au_6773).

Die Person sw ist verantwortlich für Ereignis event1, da sie der Besitzer des beteiligten Fahrzeugs b_au_6773 ist, welches an einer Bushaltestelle abgestellt wurde. Das Ordnungsgeld beträgt 20 DM. Die Prädikatsdeklarationen der im Ereignis event1 vorkommenden Prädikate sind:

responsible/2: <person>, <event>
involved_vehicle/2: <event>, <vehicle>
owner/2: <person>, <vehicle>
car_parked/2: <event>, <place>
car_towed/2: <event>, <vehicle>
fine/2: <event>, <amount>
bus_lane/1: <place>
sedan/1: <vehicle>

Die in [Morik *et al.*, 1993, S. 254] aus einer Standardmenge von Regelmodellen gelernte Regel

responsible(Z,X) ← involved_vehicle(X,Y) & owner(X,Z)

ist für alle Beispiele von responsible gültig, da nur leichte Verstöße, wie z.B. Parkvergehen, dargestellt sind, und nach deutschem Recht dafür unabhängig vom Fahrer stets der Besitzer verantwortlich ist.

Mat lernt in 19 Sekunden mit den in Tabelle 5.1 dargestellten Parametern 6 nicht redundante Regelmodelle. Keines davon ist ein Schema für obige Regel, aber Mat lernt das prädikatsvariablenfreie Regelschema

max_linking_depth	unlimited
use_head_out_args	no
percentage_of_pos_examples	unrestricted
factchain_lgg	all_oi_lgg
mpred_lgg	all_oi_lgg
max_number_of_pairs	unrestricted
max_number_of_mpred_generalisation_layers	unrestricted

Tabelle 5.1: Parametereinstellung von MAT für die Relation responsible

$$\text{responsible(X0, X1)} \leftarrow \text{involved\_vehicle(X1, X2)} \ \& \ \text{owner(X0, X2)} \\ \& \ \text{sedan(X2)},$$

eine echte Spezialisierung dieser Regel, die ebenso gültig ist, da sedan(X2) für alle Fahrzeuge X2 im Sachbereich wahr ist. Die anderen Regelmodelle enthalten zum Teil Prädikatsvariablen, sind aber entweder zu speziell oder zu allgemein. Wie erwartet wurde also in sehr kurzer Zeit eine kleine Menge von Regelmodellen gelernt, die ein geeignetes Regelmodell beziehungsweise eine Regel enthält, die alle Beispiele der Zielrelation beschreibt.

5.2 BLEARN II

Ziel des Projektes BLEARN II ist die Entwicklung flexibler und benutzerfreundlicher Robotersysteme [Klingspor und Morik, 1995]. Das Szenarium besteht aus einem Roboter, der sich in einer Umgebung bewegt und dabei mit Ultraschallsensoren Abstände zu anderen Objekten mißt. Die Forschungsaktivitäten konzentrieren sich dabei auf die Handhabung und die Navigation des Roboters. Anstatt dessen Bewegungen beispielsweise durch Angabe von Koordinaten zu steuern, soll der Roboter auf abstrakte Anweisungen wie "gehe durch die nächste Tür" reagieren. Dazu werden die Wahrnehmungen der Sensoren und die Handlungsanweisungen für den Roboter durch prädikatenlogische Begriffe repräsentiert. Handlung und dabei gemachte Wahrnehmungen sollen in einer gemeinsamen Repräsentation integriert werden.

In einer ersten Abstraktion werden dazu die von den Sensoren des Roboters während einer Fahrt gemessenen Abstände in Intervalle unterteilt und diese zu einfachen Basismerkmalen (engl. *basic features*) umgewandelt. Ein Beispiel für eine solche Sequenz von Merkmalen ist:

```
stable(t51,180,s17,20,23,0)
incr_peak(t51,180,s17,23,24,87)
stable(t51,180,s17,24,40,0)
no_movement(t51,180,s17,40,42,-999)
```

Die Argumente dieser Prädikate sind die Nummer der Roboterfahrt (engl.

trace), die Orientierung des Sensors, die Nummer des Sensors, Start– und Endzeitpunkt des Basismerkmales und der Gradient.

In diesem Szenarium der Roboternavigation gibt es eine Reihe von Lernzielen. Die Basismerkmale für einen Sensor sollen weiter zu komplexeren Sensormerkmalen abstrahiert werden, die ganze Muster solcher Basismerkmale beschreiben. Daneben gilt es aber auch, die Begriffe für Handlungsanweisungen über wahrnehmungsintegrierende Handlungsmerkmale zu definieren [Sklorz, 1995].

Dieser Sachbereich eignet sich zum Testen des Werkzeuges MAT, weil hier das syntaktische Muster der zu entdeckenden Regeln nicht immer bekannt ist. So kann ein Sensormerkmal durch eine Sequenz von Basismerkmalen beliebiger Länge charakterisiert sein. Diese Beobachtung gab bereits Anlaß zur Entwichlung des Lernverfahrens GRDT [Klingspor, 1994], bei dem Regelmodelle durch kontextfreie Grammatiken beschrieben werden und so eine beliebige Tiefe haben. Außerdem können einige der definierten Einschränkungen von Faktenketten sinnvoll zum Einsatz gebracht werden.

5.2.1 Lernen einfacher Sensormerkmale

Zum Lernen von einfachen Sensormerkmalen aus den Basismerkmalen wird ein Sachbereich mit 889 Fakten zu Basismerkmalen benutzt. Es existieren Beispiele für 4 Sensormerkmale. Zwei von MAT konstruierte Faktenketten der Sensormerkmale s_jump und s_concave sind:

```
s_jump(t51,s17,20,42,parallel) ←
    stable(t51,180,s17,20,23,0) &
    incr_peak(t51,180,s17,23,24,87) &
    stable(t51,180,s17,24,40,0) &
    no_movement(t51,180,s17,40,42,-999).

s_concave(t56,s4,37,86,diagonal) ←
    increasing(t56,45,s4,37,60,15) &
    something_happened(t56,45,s4,60,61,-21) &
    straight_to(t56,45,s4,61,86,-48).
```

Verkettungen mit Basismerkmalen aus anderen Roboterfahrten oder mit Termen anderer Sorten verhindert die entsprechende Prädikatsdeklaration:

s_concave/5: !<trace>, !<sensor>, !<time>, <time>, !<orientation>
stable/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
increasing/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
incr_peak/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
no_movement/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
straight_to/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>
something_happened/6: !<trace>, <s_alpha>, !<sensor>, !<time>, <time>, <grad>.

Das Auffinden der Ausgabeterme beendet die Verkettung mit dem Hintergrundwissen, sobald der Endzeitpunkt des Sensormerkmales in einem Basismerkmal auftritt. Die Faktenketten der für die Versuche gewählten Zielrelationen s_jump und s_concave wurden daher korrekt konstruiert.

Zielrelation	$\|E^+\|$	$\|B\|$	# Regelmodelle	Zeit
s_concave	27	889	17	18:22 min
s_jump	40	889	62	15:25 min

Tabelle 5.2: Lernen von Regelmodellen für Sensormerkmale

Die Ergebnisse der beiden Lernläufe mit dem Werkzeug MAT sind in Tabelle 5.2 festgehalten. Tabelle 5.3 gibt die wichtigen Teile der verwendeten Parametereinstellung wieder.

max_linking_depth	unlimited
use_head_out_args	yes
percentage_of_pos_examples	unrestricted
factchain_lgg	plotkin
mpred_lgg	all_oi_lgg
max_number_of_pairs	unrestricted
max_number_of_mpred_generalisation_layers	unrestricted

Tabelle 5.3: Parametereinstellung von MAT zum Lernen von Sensormerkmalen und operationalen Begriffen

Für beide Zielrelationen wird in kurzer Zeit eine nicht zu große Menge von Regelmodellen berechnet. Der Vergleich von mit in RDT verwendeten Regelmodellen entfällt hier, da aus den beschriebenen Gründen dieses Werkzeug in diesem Sachbereich nicht erfolgreich angewendet worden konnte [Klingspor, 1994]. Daher wurde getestet, ob RDT mit den gelernten Regelmodellen sinnvolle Regeln entdeckt. Das Akzeptanzkriterium bei diesen Tests war $\mathcal{A} = \{$inductive_leap_percentage = 20, min_no_of_pos_examples = 3, max_no_of_exceptions = 0, closed_world_assumption = no$\}$. Die Resultate sind in Tabelle 5.4 festgehalten.

Relation	# Regelmodelle	# Regeln	Abdeckung	Zeit
s_concave	17	2	26%	2:23 min
s_jump	62	7	50%	1:06 min

Tabelle 5.4: Resultate der BLEARN II Lernversuche mit RDT

Eine der gelernten Regel zur Relation s_jump ist:

s_jump(Tr,S,T1,T5,parallel) ←
  stable(Tr,Or,S,T1,T2,X9) &

incr_peak(Tr,Or,S,T2,T3,X7) &
stable(Tr,Or,S,T3,T4,0) &
no_movement(Tr,Or,S,T4,T5,-999).

Sie beschreibt eine korrekte Folge von Basismerkmalen für diese Relation und deckt 32,5% der positiven Beispiele ab. Insgesamt werden genau 50% der Beispiele von den 7 gefundenen Regeln abgedeckt, was als positives Resultat zu bewerten ist. Bei der Relation s_concave wird durch die 2 gelernten Regeln nur eine Abdeckung von 26% erreicht.

5.2.2 Lernen operationaler Begriffe

Der Sachbereich zum Lernen operationaler Begriffe enthält 29 Prädikate, bestehend aus 8 verschiedenen operationalen Begriffen, wahrnehmungsintegrierenden Handlungsmerkmalen und weiterem Hintergrundwissen. Insgesamt gibt es 345 Fakten. Für die ausgewählte Zielrelation move_through_door existieren nur 5 Beispiele. Folgende Faktenkette eines solchen Beispiels zeigt exemplarisch, wie wahrnehmungsintegrierende Handlungsmerkmale und weiteres Hintergrundwissen mit diesem Begriff verknüpft sind:

move_through_door(t204,60,92,90) ←
standing(t204,60,63,in_front_of_door,front,small_side,in_front_of_door) &
moving(t204,63,90,slow_move,front,in_front_of_door,back) &
parallel_moving(t204,63,90,slow_move,front,through_door,right_and_left) &
standing(t204,90,92,in_front_of_door,back,small_side,through_door) &
standing(t204,90,92,in_front_of_door,back,small_side,in_front_of_door) &
bg_opposite_direct(front,back) &
bg_opposite_direct(back,front)

Der Roboter fährt durch eine Tür, indem er erst mit seiner Vorderseite vor dieser Tür steht, dann durch diese hindurch fährt und zum Schluß mit seiner Rückseite vor dieser Tür steht.

Neben den Benutzersorten lassen sich für die einzelnen Begriffe bestimmte funktionale Abhängigkeiten definieren:

move_through_door/4: !<trace>, !<time>, <time>, <time>
standing/7: !<trace>, !<time>, <time>, <direct>, <direct>, <direct>,
<percecption>
moving/7: !<trace>, !<time>, <time>,<speed>, <direct>,
<percecption>, <direct>
parallel_moving/7: !<trace>, !<time>, <time>,<speed>, <direct>,
<percecption>, <direct>
bg_opposite_direct/2: !<direct>, !<direct>

Durch diese Prädikatsdeklarationen und die Abbruchbedingung **use_head_-out_args** werden nur relevante Fakten mit den Beispielen verkettet. MAT lernt mit der in Abbildung 5.3 angegebenen Parametereinstellung die drei folgenden voll instanziierten Regelmodelle:

move_through_door(Tr, Y, V4, V2)←
  standing(Tr, Y, Z, in_front_of_door, U, V, V1)
  moving(Tr, Z, V2, V3, U, in_front_of_door, back)
  standing(Tr, V2, V4, in_front_of_door, back, V, in_front_of_door)

move_through_door(Tr, Y, V4, V2)←
  standing(Tr, Y, Z, in_front_of_door, U, V, in_front_of_door)
  bg_opposite_direct(U, V1)
  moving(Tr, Z, V2, V3, front, in_front_of_door, V1)
  standing(Tr, V2, V4, in_front_of_door, V1, V, in_front_of_door)

move_through_door(Tr, Y, V4, V1)←
  standing(Tr, Y, Z, in_front_of_door, U, small_side, V)
  parallel_moving(Tr, Z, V1, V2, U, through_door, V3)
  standing(Tr, V1, V4, in_front_of_door, back, small_side, through_door)

Alle drei Regeln sind korrekte Beschreibungen des operationalen Begriffes move_through_door [Sklorz, 1995].

Lernen der Regelmodelle

$\|E^+\|$	$\|B\|$	# Regelmodelle	Zeit MAT
5	152	3	1:32 min

Testen der Regelmodelle

# Regeln	Abdeckung	Zeit RDT
3	100%	0:36 min

Tabelle 5.5: Lernen von Regelmodellen für die Relation move_through_door

Die Resultate der Tests mit dieser Zielrelation finden sich in der Tabelle 5.5.

5.3 SPEED

Der Umgang mit Sicherheitsproblemen in der Telekommunikation ist Inhalt des Projektes SPEED, einer Anwendung von MOBAL in Kooperation mit Alcatel Alshtom Recherche, Paris [Sommer *et al.*, 1994]. Ein Telekommunikationsnetzwerk ist ein verteiltes System, an dessen Schaltstellen unterschiedliche Operationen ausgeführt werden können. Ziel dieser Anwendung ist eine einheitliche Sicherheitspolitik zu garantieren, die nur authorisierten Personen bestimmte Operationen an einzelnen Schaltstellen erlaubt.

Eine Teilmenge der Prädikate aus SPEED ist:

may-operate/3: <user>, <component>, <operation>
time/1: <time-of-day>
optype/2: <operation>, <operation-type>
rented-by/2: <switch>, <company>
dept/1: <department>
manages/2: <department>, <switch>
manager/1: <user>

operator/1: <user>
works-in/2: <user>, <department>
sec-man/1: <user>

Die Relation may-operate beschreibt Benutzer, die an einer Komponente des Systems eine Operation ausführen dürfen. Der Sachbereich enthält 27 Prädikate, für die keine funktionalen Abhängigkeiten definiert sind und die alle kompatibel zu der Relation may-operate sind. Es existieren 139 positive Beispiele für diese Relation. Das Hintergrundwissen besteht aus 170 Fakten.

Mit der Parametereinstellung aus Tabelle 5.6 konstruiert MAT folgende Faktenkette des Beispiels may-operate(earlich,sabxb-01,op8):

may-operate(earlich,sabxb-01,op8) ←
sw(sabxb-01) &
owner(sabxb-01,wichtig-co) &
erc(sabxb-01,erb11) &
location(sabxb-01,bonn) &
manager(earlich) &
manages(std,sabxb-01) &
optype(op8,log-create) &
rented-by(sabxb-01,gross-kg) &
slog(sabxb-01,slogb1) &
stat(sabxb-01,op) &
subsystem(pabxb-17,sabxb-01) &
swtable(sabxb-01,table-b1) &
works-for(earlich,gross-kg) &
works-in(earlich,std) &
sw(pabxb-17) &
owner(pabxb-17,wichtig-co) &
co(gross-kg) &
co(wichtig-co) &
covers(wichtig-co,bonn) &
dept(std) &
has-dept(gross-kg,std) &
location(pabxb-17,bonn) &
stat(pabxb-17,op)

Selbst bei einer Tiefenbeschränkung von 1 sind die Faktenketten relativ groß. MAT lernt mit den Parametern aus Tabelle 5.6 184 Regelmodelle in 2:12 Stunden. Der Versuch, daraus mit RDT Regeln zu lernen, wurde nach 2 Tagen abgebrochen. Auch kann in so einer großen Menge von Regelmodellen nicht überprüft werden, ob die "richtigen" Regelmodelle darin enthalten sind. Sowohl die Faktenketten als auch die Regelmodelle enthalten zu viele Literale, deren Anzahl auch durch die Bildung speziellster Generalisierungen nicht wesentlich geringer wird.

Positive Ergebnisse lassen sich erst erreichen, wenn funktionale Abhängigkeiten für die Prädikate der Theorie definiert werden. Obige Faktenkette kann dadurch beispielsweise zu

max_linking_depth	1
use_head_out_args	no
arg_extention_disjunct	user_ given:[1,2,3]
percentage_of_pos_examples	30
factchain_lgg	plotkin
mpred_lgg	all_oi_lgg
max_number_of_pairs	5
max_number_of_mpred_generalisation_layers	2

Tabelle 5.6: Parametereinstellung I von MAT zum Lernen von Zugriffsrechten

max_linking_depth	unlimited
use_head_out_args	no
arg_extention_disjunct	no
percentage_of_pos_examples	unrestricted
factchain_lgg	plotkin
mpred_lgg	one_oi_lgg
max_number_of_pairs	3
max_number_of_mpred_generalisation_layers	10

Tabelle 5.7: Parametereinstellung II von MAT zum Lernen von Zugriffsrechten

```
may-operate(earlich,sabxb-01,op8) ←
    sw(sabxb-01) &
    manager(earlich) &
    optype(op8,log-create) &
    works-in(earlich,std) &
    dept(std) &
    manages(std,sabxb-01) &
    service_provider(std)
```

reduziert werden. Dabei wird aber sehr weitgehendes Wissen über die Zusammenhänge der Prädikate bezüglich der Relation may-operate benötigt. Die definierten Abhängigkeiten erlauben nur wenigen Prädikaten die Einführung neuer Terme. Mit der Parametereinstellung aus Tabelle 5.7 lernt MAT aus diesen reduzierten Faktenketten 42 Regelmodelle in 7:13 min. Daraus lernt RDT mit $\mathcal{A}$ = {inductive_leap_percentage = any, min_no_of_pos_examples= 3, max_no_of_exceptions= 0, closed_ world_assumption= yes} 6 Regeln in 1:29 min, die 92.8% der Beispiele abdecken.

5.4 Zusammenfassung

Die durchgeführten Experimente machen deutlich, daß die Konstruktion der Faktenketten starken Einfluß auf das Lernergebnis hat. In dem Sachbereich der Roboternavigation war eine Reduzierung auf die relevanten Anteile der Faktenketten ohne Kenntnisse der zu entdeckenden Regeln möglich. Die Ergebnisse sind als erfolgreich zu bewerten, zumal die in dem Projekt BLearn II bisher durchgeführten Lernversuche mit vorgegebenem Modellwissen keine besseren Resultate brachten.

Im Sachbereich Speed wurden ohne Verwendung spezieller Einschränkungen keine verwertbaren Resultate erzielt. Erst die Definition von Abhängigkeiten unter Ausnutzung der zu lernenden Regeln konnte die Faktenketten auf wirklich relevante Teile reduzieren. Dadurch wurden dann aber Regelmodelle erworben, die zu Regeln mit einer hohen Abdeckung der Beispiele führten.

Insgesamt kann beobachtet werden, daß ohne ausreichende Beschränkung der Faktenketten und der Suchbreite von Mat, dieses Verfahren dazu neigt, eine zu große Menge von Regelmodellen zu lernen, deren Elemente zudem oft viele Literale enthalten, also zu speziell sind. Auch die Auswahl einer kleinen Beispielmenge für den Lernlauf trägt nicht wesentlich zu einer guten Lösung bei.

Kapitel 6

Schlußfolgerung und Ausblick

Ziel dieser Arbeit war die Entwicklung eines Verfahrens zum automatischen Erwerb von Regelmodellen. Dieses Verfahren soll die beschriebene Lücke bei der Kooperation der Lernverfahren im System MOBAL schließen.

Regelmodelle kommen in der Werkbank MOBAL zur Reduzierung und Strukturierung des Suchraumes des Lernverfahrens RDT zum Einsatz. In dieser Arbeit wurde eine neue Allgemeinheitsordnung über Regelmodellen als Erweiterung der θ–Subsumtion definiert. Sie dient zum Abschneiden von Teilen des Suchraumes während eines Lernlaufes und stellt eine Verbesserung der bisher benutzten Relation dar, welche in Kapitel 3 dieser Arbeit als zu restriktiv charakterisiert wurde. Auch wurde erstmals ein Redundanzbegriff für Regelmodelle definiert, der mehr überflüssige Regelmodelle für die Vorgabe des Suchraumes erkennt als der bisherige Test. Die Realisierungen dieser Definitionen in der Werkbank MOBAL verbessern somit das Werkzeug RDT zum Entdecken von Regelhaftigkeiten.

Zum Lernen von Regelmodellen wurde eine heuristischer Ansatz vorgestellt und implementiert, der auf der Berechnung speziellster Generalisierungen von Regelmodellen beruht. Um Hintergrundwissen zu berücksichtigen werden relevante Teile dieser Formelmenge mit den Beispielen zu Faktenketten verknüpft. Dabei werden zum Teil neue Ideen zur Beschränkung dieser Saturierung der Beispiele benutzt.

Der Ansatz Regelmodelle durch speziellste Generalisierungen von Faktenketten zu lernen, kann nur eingeschränkt als erfolgreich bewertet werden. Wie die durchgeführten Experimente zeigen, kann das Werkzeug MAT zwar derart über die Parameter gesteuert werden, daß der Algorithmus terminiert, die erworbene Menge von Regelmodellen kann aber nicht immer als sinnvolle Eingabe für das System RDT dienen. Insbesondere werden durch diesen Ansatz oft zu viele und zu spezielle Regelmodelle gelernt.

Dagegen gibt es auch einige positive Resultate, die vor allem darauf beruhen, daß die Faktenketten in diesen Fällen nur die wirklich relevanten

Fakten der Beispiele enthalten und so die speziellste Generalisierung dieser Faktenketten bereits ausreichend allgemein ist.

Zusammenfassend bleibt festzustellen, daß der Versuch Regelmodelle zu lernen, ohne eine Bewertungsfunktion für jedes einzelne Regelmodell zu haben, schwierig ist. In dieser Beobachtung steckt auch mögliches Potential für weitere Arbeiten. Zieht man einen Test der Hypothesen oder einen noch zu definierenden Test für Regelmodelle hinzu, so stehen die üblichen Techniken zur Suchraumeinschränkung zur Verfügung. Allerdings stellt sich dann die Frage, ob das Lernen von Modellwisen überhaupt sinnvoll ist und ein Lernverfahren nicht direkt Regeln entdecken sollte.

Die einzelnen Algorithmen des Werkzeuges Mat sind zum Teil sehr effizient. Insbesondere für die verbesserten Algorithmen zur θ–Subsumtion und Reduktion von Hornklauseln konnte dies nachgewiesen werden [Kietz und Lübbe, 1994]. Sie kommen daher bereits in anderen ILP Lernverfahren zum Einsatz.

Auch die Konstruktion der Faktenketten mit den in dieser Arbeit definierten Einschränkungen der Verkettung kann in anderen Lernverfahren zur Saturierung der Beispiele oder anderen Zwecken eingesetzt werden. Eine Übertragung der Ideen der Faktenketten auf Datenbanken ist im Bereich des *Knowledge Discovery* denkbar.

Danksagung

Ein wichtiger Dank sei Katharina Morik für die Betreuung meiner Diplomarbeit ausgesprochen. Sie gab zusammen mit Jörg-Uwe Kietz die Anregung zu dieser Arbeit und stand für fruchtbare Diskussionen bereit. Mein besonderer Dank gilt Jörg-Uwe Kietz für die kritischen Kommentare und interessanten Diskussionen während der Entwicklung dieser Arbeit. Seine Anregungen halfen in einigen schwierigen Situationen weiter. Bei Joachim Hertzberg bedanke ich mich für die Begutachtung dieser Arbeit. Mein weiterer Dank gilt den Mitgliedern der Gruppe Maschinelles Lernen der GMD für Hinweise bei der Integration des Verfahrens in MOBAL sowie allen Mitarbeitern und Studenten des Lehrstuhls VIII.

Anhang A

Grundlagen der Komplexitätstheorie

Für eine große Anzahl wichtiger und praktisch relevanter Probleme sind bisher weder effiziente Algorithmen gefunden worden, noch konnte der Beweis geführt werden, daß es für diese Probleme keine effizienten Algorithmen gibt. Dabei versucht man, ein Problem bezüglich eines geeigneten Reduktionsbegriffes auf ein anderes Problem zu reduzieren. Gelingt dies, so ist das erste Problem nicht schwerer als das zweite und jede untere Schranke des ersten Problems führt zu einer unteren Schranke gleicher Größenordnung für das zweite Problem. Eine weitere Eigenschaft ist, daß jeder Algorithmus für das zweite Problem zu einem gleich effizienten Algorithmus für das erste Problem führt. Lassen sich zwei Probleme gegenseitig aufeinander reduzieren, so sind sie gleich schwer.

Von besonderer Bedeutung ist dabei die Klasse der gleich komplexen, *NP*–vollständigen Probleme. Für diese Klasse von Problemen wurde nachgewiesen, daß es entweder für jedes oder aber für kein *NP*–vollständiges Problem einen deterministischen Algorithmus mit polynomieller Laufzeit gibt (also keinen effizienten Algorithmus). Allgemein wird von der zweiten Annahme ausgegangen, da die Entdeckung eines polynomiellen Verfahrens für ein einziges *NP*–vollständiges Problem zugleich den Nachweis erbringen würde, daß alle diese Probleme effiziente Lösungsverfahren besitzen.

Ein Problem läßt sich formalisieren, indem man die Ein– und Ausgaben des Problems über einem Alphabet Σ (z.B. $\Sigma = \{0, 1\}$) codiert. Ein Problem ist dann eine binäre Relation R auf $\Sigma^* \times \Sigma^*$, wobei ein Paar (x, y) genau dann ein Element der Relation R ist, wenn y eine zulässige Ausgabe oder Lösung zur Eingabe x ist. Entscheidungsprobleme lassen sich durch eine Funktion $f : \Sigma^* \to \{0, 1\}$ darstellen und können mit einer Sprache $L = f^{-1}(1)$ identifiziert werden. Die Sprache zu einem Entscheidungsproblem ist also die Menge der Eingaben, die auf eine positive Ausgabe abgebildet werden.

Damit läßt sich der bereits angesprochene Reduktionsbegriff für Entscheidungsprobleme formal wie folgt definieren [Garey und Johnson, 1979, S. 34]:

Definition A.1 (polynomiell-reduzierbar, $\trianglelefteq_{pol}$) *Eine Sprache $L_1 \subseteq \Sigma_1^*$ läßt sich* **polynomiell** *auf eine Sprache $L_2 \subseteq \Sigma_2^*$* **reduzieren**, *notiert als $L_1 \trianglelefteq_{pol} L_2$, wenn es eine polynomiell berechenbare Transformation $f : \Sigma_1^* \rightarrow \Sigma_2^*$ gibt, so daß gilt:*

$$\forall x \in \Sigma_1^* : x \in L_1 \Leftrightarrow f(x) \in L_2.$$

Für eine Klasse C von Sprachen heißt ein Problem beziehungsweise eine Sprache vollständig, wenn sie zu den schwierigsten Problemen aus C gehört [Garey und Johnson, 1979, S. 37, S. 109].

Definition A.2 (C–vollständig, C–schwierig)
*Eine Sprache $L \subseteq \Sigma^*$ heißt C–***vollständig**, *falls $L \in C$ ist und für alle $L' \in C$ gilt $L' \trianglelefteq_{pol} L$.*
*Eine Sprache L heißt C–***schwierig**, *wenn für alle $L' \in C$ gilt: $L' \trianglelefteq_{pol} L$.*

Die Klasse *NP* enthält alle Entscheidungsprobleme, die von einer nicht deterministischen Turingmaschine in polynomieller Zeit entschieden werden können. Eine Sprache L ist demnach *NP*–vollständig, wenn L in der Komplexitätsklasse *NP* ist und jedes andere Problem aus *NP* bezüglich $\trianglelefteq_{pol}$ nicht schwieriger als L ist. *NP*–schwierige Problem müssen selber nicht in *NP* liegen.

Nachdem Cook beim Erfüllbarkeitsproblem (SAT) der erste Beweis für die *NP*–vollständigkeit eines Problems gelang, vereinfacht folgender Zusammenhang den Nachweis der *NP*–vollständigkeit anderer Probleme [Garey und Johnson, 1979, S. 38f].

Satz A.1 *$L_2 \in NP, L_1$ NP–vollständig, $L_1 \trianglelefteq_{pol} L_2 \Rightarrow L_2$ NP–vollständig.*

Gehört also ein Problem L_2 in die Klasse *NP* und ist mindestens so schwer wie ein Problem L_1 aus der Klasse der schwierigsten Probleme in *NP*, so gehört auch L_2 zu den schwierigsten Problemen in *NP*.

Der Beweis der *NP*–Vollständigkeit eines Problems $L \subseteq \Sigma^*$ folgt nach diesen Ergebnissen nachstehendem Muster:

1. Es wird bewiesen, daß $L \in NP$ ist.
2. Ein geeignetes *NP*–vollständiges Problem $L' \subseteq (\Sigma')^*$ wird ausgewählt.
3. Nun gilt es, eine deterministische in polynomieller Zeit berechenbare Transformation $f : (\Sigma')^* \rightarrow \Sigma^*$ anzugeben.
4. Abschließend ist noch $x \in L' \Leftrightarrow f(x) \in L$ zu beweisen.

Gelingt nun für ein Lernproblem der Nachweis der NP–vollständigkeit, so wird sich unter der Annahme $NP \neq P$ kein effizientes Lernverfahren für dieses Problem finden lassen.

Obige Theorie läßt sich auf Suchprobleme verallgemeinern. Zu diesem Zweck werden Suchprobleme mit ihren Stringrelationen R identifiziert. Die Verallgemeinerung der polynomiellen Reduktion auf Suchprobleme, besteht nun darin, bei einer Reduktion von Suchproblemen $R_1 \trianglelefteq R_2$ einen fiktiven Algorithmus für das Problem R_2 als Unterprozedur in einem Algorithmus für R_1 einzusetzen und diese Unterprozedur nur polynomiell oft aufzurufen. Dieses Vorgehen läßt sich durch Orakel–Turingmaschinen formalisieren [Garey und Johnson, 1979, S. 111f].

Definition A.3 (Orakel–Turingmaschine) *Eine Orakel–Turingmaschine mit einem Orakel $g : \Sigma^* \to \Sigma^*$ ist eine deterministische Turingmaschine mit einem zusätzlichen Orakelband und zwei zusätzlichen Zuständen q_f (Orakelfrage) und q_a (Orakelantwort). Außer im Zustand q_f arbeitet die Orakel–Turingmaschine wie üblich. Im Zustand q_f und Inhalt x des Orakelbandes wird in einem Schritt der Inhalt des Orakelbandes gelöscht, die Antwort $g(x)$ auf das Orakelband geschrieben und die Orakel–Turingmaschine geht in den Zustand q_a.*

Eine Funktion f realisiert eine Relation $R \subset \Sigma^* \times \Sigma^*$, wenn für alle $x \in \Sigma^+$ gilt: Gibt es kein $y \in \Sigma^+$ mit $(x, y) \in R$, so ist $f(x) = \epsilon$. Ansonsten ist $f(x) = y$ mit $(x, y) \in R$. Das Orakel berechnet also eine Funktion g, die eine bestimmte Relation R realisiert. Berechnet eine Orakel–Turingmaschine mit einem Orakel g in polynomieller Zeit eine Funktion f und läßt sich das Orakel g durch einen polynomiellen Algorithmus ersetzen, so ist auch f polynomiell berechenbar. Damit ergibt sich folgender Reduktionsbegriff für Suchprobleme [Garey und Johnson, 1979, S. 113]:

Definition A.4 (polynomiell–Turing–reduzierbar, $\trianglelefteq^T_{pol}$) *Seien R und R' zwei Stringrelationen über Σ. Die Relation R heißt genau dann polynomiell–Turing–reduzierbar auf R', notiert als $R \trianglelefteq^T_{pol} R'$, wenn eine Orakel–Turingmaschine M existiert, deren Orakel die Relation R' realisiert und die selber polynomiell zeitbeschränkt eine Realisierung von R berechnet.*

Mit der Menge $\{ja, nein\}$ als zulässige Ausgaben lassen sich alle Sprachen oder Entscheidungsprobleme auch als Suchprobleme auffassen. Diese Konstruktion wird bei der Definition des Begriffes C–schwierig für Suchprobleme ausgenutzt [Garey und Johnson, 1979, S. 113].

Definition A.5 (C-schwierig) *Eine Relation (Suchproblem) R heißt genau dann C–schwierig, wenn es eine C–vollständige Sprache L mit $L \trianglelefteq^T_{pol} R$ gibt.*

Die Funktionsweise einer Orakel–Turingmaschine macht deutlich, daß die C–schwierigen Probleme mindestens so schwierig sind wie die C–vollständigen Probleme.

Erwähnt werden sollen hier noch die Komplexitätsklassen RP, $co\text{–}C$ und $PSPACE$ ($NPSPACE$).

RP ist die Klasse aller Probleme (Sprachen) für die es eine polynomiell zeitbeschränkte probalistische Turingmaschine gibt, die unzulässige Eingaben nicht erkennt und Wörter der Sprache mit einer Wahrscheinlichkeit größer als 1/2 erkennt. Es gilt: $P \subseteq RP \subseteq NP$. Für eine Klasse C von Sprachen ist $co\text{–}C$ die Menge der Sprachen, deren Komplement in der Komplexitätsklasse C liegt. $PSPACE$ ($NPSPACE$) ist die Klasse der Probleme, die von einer (nicht–) deterministischen Turingmaschine mit polynomiellem Platzbedarf erkannt werden können. Es wird vermutet, daß $NP \neq co\text{–}NP$ ist, und es gilt $P \subseteq \begin{matrix} co\text{–}NP \\ NP \end{matrix} \subseteq PSPACE = NPSPACE$.

Die in Kapitel 2.2.3 beschriebenen Einschränkungen des induktiven Lernenproblems besitzen zum Teil polynomielle Algorithmen oder aber sie sind NP–vollständig oder noch schwerer (z.B. $PSPACE$–schwierig).

Literaturverzeichnis

[Bell und Weber, 1993] Siegfried Bell und Steffo Weber. On the close logical relationship between FOIL and the frameworks of Helft and Plotkin. In *ILP'93 Workshop*, Bled, 1993. Als erweiterte Fassung auch veröffentlicht als LS-8 Report 4 in den Forschungsberichten des Lehrstuhls VIII, Fachbereich Informatik der Universität Dortmund.

[Bergadano und Gunetti, 1994] F. Bergadano und D. Gunetti. Learning Clauses by Tracing Derivations. In Stefan Wrobel (Hrsg.), *Fourth Int. Workshop on Inductive Logic Programming*, S. 11 – 29, 1994.

[Blair und Subrahmanian, 1989] H.A. Blair und V.S. Subrahmanian. Paraconsistent Logic Programming. *Theoretical Computer Science*, 68:135–154, 1989.

[Bossu und Siegel, 1985] G. Bossu und P. Siegel. Saturation, Nonmonotonic Reasoning. *Artificial Intelligence*, 25, 1985.

[Bratko und King, 1993] Ivan Bratko und Ross King. Applications of Inductive Logic Programming. *SIGART Bulletin*, 5(1):43 – 49, 1993.

[Buntine, 1988] Wray Buntine. Generalized Subsumption and Its Applications to Induction and Redundancy. *Artificial Intelligence*, 36:149 – 176, 1988.

[Cohen, 1993] William W. Cohen. Learnability of Restricted Logic Programs. In *ILP'93 Workshop*, Bled, 1993.

[Cohen, 1994] William W. Cohen. Grammatically Biased Learning: Learning Logic Programs Using an Explicit Antecedent Description Language. *AI Journal*, 68(2):243–302, 1994.

[De Raedt und Bruynooghe, 1993] L. De Raedt und M. Bruynooghe. A theory of clausal discovery. In Stephen Muggleton (Hrsg.), *The Third International Workshop on Inductive Logic Programming*, April 1993.

[De Raedt, 1992] L. De Raedt. Interactive concept–learning and constructive induction by analogy. *Machine Learning*, 8:107 – 150, 1992.

[Dolsak und Muggleton, 1992] B. Dolsak und S. Muggleton. The Application of Inductive Logic Programming to Finite Element Mesh Design.

In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, S. 453–472. Academic Press, 1992.

[Dzeroski *et al.*, 1992] Saso Dzeroski, Stephen Muggleton und Stuart Russell. PAC-Learnability of Determinate Logic Programs. *Proceedings of 5th Annual Conference on Computational Learning Theory*, S. 128 – 135, 1992.

[Emde *et al.*, 1983] W. Emde, C. Habel und C.-R. Rollinger. The discovery of the equator or concept driven learning. In *IJCAI-83*, S. 455–458. Morgan Kaufman, 1983.

[Feng und Muggleton, 1992] C. Feng und S. Muggleton. Towards inductive generalisation in higher order logic. In *Ninth International Workshop on Machine Learning*, S. 154 – 162. Morgan Kaufman, 1992.

[Garey und Johnson, 1979] M. Garey und D. Johnson. *Computers and intractability: A guide to the theory of NP–completeness.* W.H.Freeman, San Francisco, 1979.

[Gold, 1967] E Mark Gold. Language Identification in the Limit. *Information and Control*, 14:447–474, 1967.

[Gottlob und Fermüller, 1993] Georg Gottlob und Christian G. Fermüller. Removing redundancy from a clause. *Artificial Intelligence*, 61:263–289, 1993.

[Gottlob und Leitsch, 1985] G. Gottlob und A. Leitsch. On the Efficiency of Subsumption Algorithms. *Journal of the Association for Computing Machinery*, 32(2):280–295, 1985.

[Haussler, 1989] David Haussler. Learning Conjunctive Concepts in Strutural Domains. *Machine Learning*, 4:7–40, 1989.

[Helft, 1987] Nicolas Helft. Inductive generalisation: A logical framework. In *Proceedings of the 2nd European Working Session on Learning*, 1987.

[Helft, 1989] Nicolas Helft. Induction as nonmonotonic inference. In *Proceedings of the 1st International Conference on Knowledge Representation and Reasoning*, 1989.

[Hofbauer und Kutsche, 1989] Dieter Hofbauer und Ralf-Detlef Kutsche. *Grundlagen des maschinellen Beweisens.* Vieweg, 1989.

[Jung, 1993] Bernhard Jung. On Inverting Generality Relations. In Stephen Muggleton (Hrsg.), *Procs. of the 3rd International Workshop on Inductive Logic Programming*, Nummer IJS-DP-6707 in J Stefan Institute Technical Report, S. 87 – 101, 1993.

[Kietz und Dzeroski, 1993] Jörg-Uwe Kietz und Saso Dzeroski. Inductive Logic Programming and Learnability. *SIGART Bulletin*, 5(1):22 – 32, 1993.

[Kietz und Lübbe, 1994] Jörg-Uwe Kietz und Marcus Lübbe. An Efficient Subsumption Algorithm for Inductive Logic Programming. In *11th Int. Conference on Machine Learning*, 1994. Auch veröffentlicht in Proceedings of the 4th International Workshop on Inductive Logic Programming, Bad-Honnef, GMD, 1994.

[Kietz und Wrobel, 1991] Jörg-Uwe Kietz und Stefan Wrobel. Controlling the Complexity of Learning in Logic through Syntactic and Task-Oriented Models. In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 16, S. 335 – 360. Academic Press, London, 1991. Auch als Arbeitspapiere der GMD erhältlich No. 503, 1991.

[Kietz, 1993a] Jörg-Uwe Kietz. A Comparative Study of Structural Most Specific Generalizations Used in Machine Learning. In Stephen Muggleton (Hrsg.), *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, Nummer IJS-DP-6707 in J. Stefan Institute Technical Reports, S. 149 – 164, 1993. Auch als Arbeitspapiere der GMD erhältlich No. 667, 1992.

[Kietz, 1993b] Jörg-Uwe Kietz. Some Lower Bounds for the Computational Complexity of Inductive Logic Programming. In Pavel Brazdil (Hrsg.), *Machine Learning - Proceedings of ECML-93*, Lecture Notes in Artificial Intelligence, S. 115 –123, Berlin, Heidelberg, New York, 1993. Springer. Auch als Arbeitspapiere der GMD erhältlich No. 718, 1992.

[Klingspor und Morik, 1995] Volker Klingspor und Katharina Morik. Towards Concept Formation Grounded on Perception and Action of a Mobile Robot. In *Proc. of the 4th Intern. Conference on Intelligent Autonomous Systems*, 1995. to appear.

[Klingspor, 1994] Volker Klingspor. GRDT: Enhancing Model–Based Learning for its Application in Robot Navigation. In Stefan Wrobel (Hrsg.), *Fourth Int. Workshop on Inductive Logic Programming*, S.107 – 121, 1994.

[Lloyd, 1987] J.W. Lloyd. *Foundations of Logic Programming.* Springer, Berlin, New York, 2nd Auflage, 1987.

[Maher, 1986] M.J. Maher. Equivalences of Logic Programs. In Goos und Hartmanis (Hrsg.), *3rd International Conference on Logic Programming*, 1986.

[Mitchell, 1982] Tom M. Mitchell. Generalization as Search. *Artificial Intelligence*, 18(2):203 – 226, 1982.

[Morik *et al.*, 1993] K. Morik, S. Wrobel, J.-U. Kietz und W. Emde. *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications.* Academic Press, London, 1993.

[Morik, 1993] Katharina Morik. Balanced Cooperative Modeling. *Machine Learning*, 11:217 – 235, 1993.

[Muggleton und Buntine, 1992] Stephan Muggleton und Wray Buntine. Machine Invention of First-Order Predicates by Inverting Resolution. In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 12, S. 261–280. Academic Press, London, 1992.

[Muggleton und Feng, 1992] Stephan Muggleton und Cao Feng. Efficient Induction of Logic Programs. In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 13, S. 281–298. Academic Press, London, 1992.

[Muggleton und Raedt, 1993] Stephen Muggleton und Luc De Raedt. Inductive Logic Programming: Theory and Methods. CW 178, Department of Computing Science, K.U. Leuven, May 1993.

[Muggleton, 1992] Stephan Muggleton. Inductive Logic Programming. In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 1, S. 3–28. Academic Press, London, 1992.

[Niblett, 1988] Tim Niblett. A Study of Generalisation in Logic Programs. In *Proceedings of the 3th European Working Session on Learning*, 1988.

[Nienhuys-Cheng *et al.*, 1993] S.-H. Nienhuys-Cheng, P.R.J. van der Laag und L.W.N. van der Torre. Constructing refinement operators by decomposing logical implication. In *Third Congres of the Italian Association for Artificial Intelligence*, S. 178 – 189. Springer Verlag, 1993.

[Page und Frisch, 1992] David Page und Alan Frisch. Generalization and Learnability: A Case Study of Constrained Atoms. In Stephen Muggleton (Hrsg.), *Inductive Logic Programming*, Band 1, Kapitel 2, S. 29 – 62. Academic Press, London, San Diego, 1992.

[Pazzani und Kibler, 1992] M.J. Pazzani und D. Kibler. The Utility of Knowledge in Inductive Learning. *Machine Learning*, 9:57–94, 1992.

[Pitt und Valiant, 1988] Leonard Pitt und Leslie G. Valiant. Computational Limitations on Learning from Examples. *Journal of the ACM*, 35:965–984, 1988.

[Plotkin, 1970] Gordon D. Plotkin. A Note on Inductive Generalization. In B. Meltzer und D. Michie (Hrsg.), *Machine Intelligence*, Kapitel 8, S. 153–163. American Elsevier, 1970.

[Plotkin, 1971a] Gordon D. Plotkin. A Further Note on Inductive Generalization. In B. Meltzer und D. Michie (Hrsg.), *Machine Intelligence*, Kapitel 8, S. 101–124. American Elsevier, 1971.

[Plotkin, 1971b] Gordon D. Plotkin. *Automatic Methods of Inductive Inference*. Dissertation, University of Edinburgh, 1971.

[Quinlan, 1983] J. Ross Quinlan. Learning Efficient Classification Procedures and Their Application to Chess End Games. In R.S. Michalski, J.G. Carbonell und T.M. Mitchell (Hrsg.), *Machine Learning - An Artificial Intelligence Approach*, S. 463 – 482. Tioga, Palo Alto, CA, 1983.

[Quinlan, 1990] J.R. Quinlan. Learning Logical Definitions from Relations. *Machine Learning*, 5(3):239 – 266, 1990.

[Robinson, 1965] J. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Rouveirol, 1991] Celine Rouveirol. Semantic Model for Induction of First Order Theories. In *Proceedings 12th IJCAI*, S. 685–691. IJCAI, Morgan Kaufmann, 1991.

[Shapiro, 1981] E.Y. Shapiro. Inductive Inference of Theories from Facts. Research Report 192, Yale University, 1981.

[Sklorz, 1995] Stefan Sklorz. Repräsentation operationaler Begriffe zum Lernen aus Roboter–Sensor Daten. Diplomarbeit, Universität Dortmund, 1995. to appear.

[Sommer *et al.*, 1993] Edgar Sommer, Werner Emde, Jörg-Uwe Kietz, Katharina Morik und Stefan Wrobel. MOBAL 2.2 User Guide. 777, Arbeitspapiere der GMD, Sptember 1993.

[Sommer *et al.*, 1994] Edgar Sommer, Katharina Morik, Jean-Michel Andre und Marc Uszynski. What On-line Learning Can Do For Knowledge Acquisition - A Case Study. *Knowledge Aquisition*, 1994. Auch als Arbeitspapiere der GMD erhältlich No. 757.

[Sommer, 1993] Edgar Sommer. Cooperation of data–driven and model–based methods for relational learning. In R.S. Michalski und G. Tecuci (Hrsg.), *Second International Workshop on Multistrategy Learning*, 1993.

[Tausend, 1992] Birgit Tausend. Lernen von Hornklauseln mit Programmierschemata. In K. Reiss, M.Reiss und H. Spandl (Hrsg.), *Maschinelles Lernen – Modellierung von Lernen mit Maschinen*, S. 125–142. Springer-Verlag, 1992.

[Thieme, 1989] Sabine Thieme. The aquisition of model knowledge for a model–driven machine learning approach. In K. Morik (Hrsg.), *Knowledge representation and Organization in Machine Learning*, S. 177 – 191. Springer-Verlag, 1989.

[Valiant, 1984] Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Warren, 1982] D.H.D. Warren. Higher-order extensions to PROLOG: are they needed? *Machine Intelligence*, 10:441–454, 1982.

[Wirth und O'Rorke, 1991] R. Wirth und P. O'Rorke. Constraints on Predicate Invention. In *Eigth International Workshop on Machine Learning*. Morgan Kaufmann, 1991.

[Wrobel, 1987] Stefan Wrobel. Higher-Order Concepts in a Tractable Knowledge Representation. In K. Morik (Hrsg.), *GWAI-87 11th German Workshop on Artificial Intelligence*, S. 129–138. Springer Verlag, October 1987.