Anwendung des Lernverfahrens RDT auf eine relationale Datenbank

Guido Lindner

betreut von
Prof. Dr. Katharina Morik
und
Dr. Joachim Hertzberg

August 1994

Diplomarbeit am Fachbereich Informatik an der
Universität Dortmund

Zusammenfassung

Die Verbindung maschinellen Lernenverfahren mit relationalen Datenbanken ist für verschiedene Bereiche wie *knowledge discovery in databases*, deduktiven Datenbanken und maschinellen Lernen selbst von steigendem Interesse. Diese Arbeit beschreibt die Anwendung des logikorientierten Lernverfahrens RDT auf relationale Datenbanken. Nach einer Einführung in den Aufbau relationaler Datenbanken und der Beschreibung des Verfahrens RDT wird die Frage der Repräsentation von Datenbank für das logikbasierte Lernen diskutiert. Weiterhin wird gezeigt, wie bestimmte Datenbankeigenschaften für die Einschränkung des Hypothesenraums ausgenutzt werden können. Abschließend wird RDT, angewand auf relationale Datenbanken (RDT/DB), in experimentellen Tests über dem im maschinellen Lernen bekannten Sachbereich KRK und einer Anwendung aus dem Bereich der Roboternavigation getestet.

Danksagung

An erster Stelle danke ich Katharina Morik, daß sie mich im Laufe meines Studiums an die Thematik dieser Arbeit herangeführt hat und mein Interesse für das Gebiet des maschinellen Lernens geweckt hat.Weiterhin gilt mein besonderer Dank Joachim Hertzberg und Siegfried Bell, die immer ein offenes Ohr für mich hatten. Insbesondere möchte ich noch Volker Klingspor sehr danken, mit dem ich zu jeder Zeit interessante Diskussionen führen konnte.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Zielsetzung der Arbeit

In dem ersten Abschnitt dieses Kapitels wird die Motivation zu der folgenden Arbeit gegeben. Der Abschnitt 1.2 beschreibt dann das Szenario, in das diese Arbeit einzuordnen ist. Im folgenden Abschnitt wird dann die Aufgabenstellung der Arbeit konkretisiert. Der letzten Abschnitt dieses Kapitels gibt einen Überblick über den weiteren Aufbau der vorliegenden Arbeit.

1.1 Induktives Lernen und relationale Datenbanken

1.1.1 Maschinelles Lernen als *knowledge discovery in databases*

Im Rahmen des *knowledge discovery in databases* (KDD) erhält das maschinelle Lernen (ML) eine immer größere Bedeutung. KDD ist das Entdecken von potentiell nützlichen Informationen aus Daten. Aus der Definition für KDD von Frawley, Piatetsky–Shapiro und Matheus wird die Ähnlichkeit der beiden Gebiete deutlich [Frawley *et al.*, 1991].

Definition 1 (*knowledge discovery in databases)*
Gegeben seien eine Sprache $\mathcal{L}$, eine Menge von Fakten $\mathcal{F} \subseteq \mathcal{L}$, und ein Kriterium $\mathcal{C}$. KDD ist das Problem, eine Aussage $\mathcal{S} \in \mathcal{L}$ zu finden, so daß $\mathcal{S}$ eine Faktenmenge $\mathcal{F}_\mathcal{S} \subseteq \mathcal{F}$ beschreibt, $\mathcal{S} \neq \mathcal{F}_\mathcal{S}$, und $\mathcal{S}$ dem Kriterium $\mathcal{C}$ genügt.

Auch im maschinellem Lernen geht man von einer Sprache $\mathcal{L}_H$ zur Beschreibung der Hypothese und einer Menge von Beispielen (Fakten) aus. Das Ergebnis eines maschinellen Lernvorganges ist eine Hypothese, die den Kriterien des Benutzers genügt. Diese Hypothese entspricht der Aussage $\mathcal{S}$ aus der Definition des KDD's. In diesem Sinne bezeichnen Matheus, Chan und Piatetsky–Shapiro KDD als eine Teilmenge des maschinellen Lernens [Matheus *et al.*, 1993].

Definition 2 (Lernen aus Beispielen)
Lernen aus Beispielen ist die Suche nach einer Beschreibung eines Begriffs $\mathcal{C}$, zu dem man eine endliche Anzahl an Beispielen gegeben hat.

Gegeben sei eine Menge von Beispielen in einer Sprache $\mathcal{L}_B$, eine Sprache $\mathcal{L}_H$ zur Beschreibung des zu lernenden Begriffs (Hypothesensprache) und ein Kriterium zur Akzeptanz einer Hypothese. Eine Aussage $H \in \mathcal{L}_H$, die dem Akzeptanzkriterium genügt, ist eine Beschreibung des Begriffs $\mathcal{C}$.

Lernen als Suche zu definieren, wurde erstmals von Mitchell 1982 vorgestellt [Mitchell, 1982]. In dieser Form des Lernens schließt man vom „Einzelfall“[1] auf eine allgemeine Beschreibung. Diese Art des logischen Schließens bezeichnet man als induktiven Schluß. Aus diesem Grund spricht man hier vom induktiven Lernen.

Aufgrund der Ähnlichkeit der zwei Forschungsbereiche entstand die Idee, Verfahren des induktiven Lernens im Rahmen des KDD einzusetzen. Bisher wurden schon einige induktive Lernverfahren im Rahmen des KDD eingesetzt. Holsheimer und Siebes [Holsheimer und Siebes, 1993] beschreiben den Einsatz verschiedener induktiver Verfahren im KDD, z.B. von ID3 [Quinlan, 1986], AQ15 [Michalski *et al.*, 1986] und CN2 [Clark und Niblett, 1989].

Die Bedeutung von KDD in unserem Zeitalter kann man leicht an der Tatsache erkennen, daß sich das Datenaufkommen in der Welt alle 20 Monate verdoppelt [Frawley *et al.*, 1991]. Dieses Potential an Informationen kann von Menschen alleine nicht mehr gesichtet und bewertet werden, so daß eine maschinelle Unterstützung dringend erforderlich ist. So existieren zum Beispiel bei der amerikanischen Raumfahrtbehörde NASA Datenbank voll mit Meßwerten von Raumsonden, von denen bis heute nicht klar ist, wann sie einmal ausgewertet werden können.

1.1.2 Induktives Lernen in deduktiven Datenbanken

Eine weitere Motivation, ein Verfahren des induktiven Lernens auf relationale Datenbanken anzuwenden, ist die Möglichkeit, diese Verfahren als induktive Komponente in deduktiven Datenbanken einzusetzen. Deduktive Datenbanken bestehen aus expliziten Daten und impliziten Daten in Form von Regeln. Eine solche induktive Komponente könnte beim Aufbau und der Pflege einer deduktiven Datenbank durch das Entdecken von bisher nicht bekannten Abhängigkeiten hilfreich sein. So beschreiben Džeroski und Lavrač den Einsatz von LINUS [Lavrač und Džeroski, 1992] zum Entdecken von virtuellen Relationen in deduktiven Datenbanken [Džeroski und Lavrač, 1993].

Bei den bisher in den Abschnitten 1.1.1 und 1.1.2 genannten Verfahren handelt es sich ausschließlich um Attributwerte–orientierte–Verfahren. Hier ist es nun von Interesse, ein Verfahren des logikbasierten Lernens wie RDT, das als Hypothesensprache eingeschränkte Prädikatenlogik 1. Stufe hat, in

[1] In Anführungszeichen, da es meist doch eine Menge von Beispielen handelt.

Zusammenhang mit relationalen Datenbanken einzusetzen.

1.1.3 Kopplung des Modellierungssystems MOBAL

Ein weiterer Motivationspunkt ist, Erfahrungen mit einem anderen Speicherkonzept für das Modellierungssystem MOBAL [Morik *et al.*, 1993] zu sammeln. In einem ersten Schritt sollte *eine* der vielen Systemkomponenten von MOBAL Anwendung auf relationale Datenbanken finden. Aufgrund der in den Abschnitten 1.1.1 und 1.1.2 schon beschriebenen Einsatzmöglichkeiten von Lernverfahren in relationalen Datenbanken wurde die Lernkomponente RDT von MOBAL ausgewählt.

Das Modellierungssystem MOBAL ist nach dem *sloppy modeling* Prinzip [Morik, 1989] aufgebaut. Aufgabe des Systems ist, den manuellen Prozeß der Wissensakquisition, der von unvollkommenem und änderbarem Wissen gekennzeichnet ist, durch Kontrolle der Wissensbasis auf Inkonsistenz zu unterstützen und so dem Benutzer eine Prüfung der Wissensbasis zu ermöglichen. Weiterhin werden in MOBAL durch die verschiedenen eingebundenen Lernverfahren dem Anwender nicht bekannte Zusammenhänge aufgezeigt.

1.2 Szenario

Das Szenario setzt sich aus einem Anwender, einer relationalen Datenbank und dem System MOBAL zusammen. Ziel ist, Begriffsdefinitionen für Relationen oder einzelne Attribute von Relationen aus der Datenbank zu lernen. In der Abbildung 1.1 ist das Szenario graphisch dargestellt.

RDT angewandt auf relationale Datenbanken nutzt drei verschiedene Informationsquellen:

1. den Benutzer,
2. die relationale Datenbank und
3. das Sachbereichswissen aus der MOBAL–Wissensbasis

Der Benutzer übt in dem angestrebten System eine kontrollierende Funktion aus. Er bestimmt die Parameter für die einzelnen Lernläufe und entscheidet, ob gelernte Zusammenhänge in das Sachbereichswissen übernommen werden sollen.

Die relationale Datenbank dient in diesem System als Faktenbasis zum Lernen. Weitere Informationen stehen dem Lernverfahren durch das Sachbereichswissen aus der MOBAL–Wissensbasis zur Verfügung.

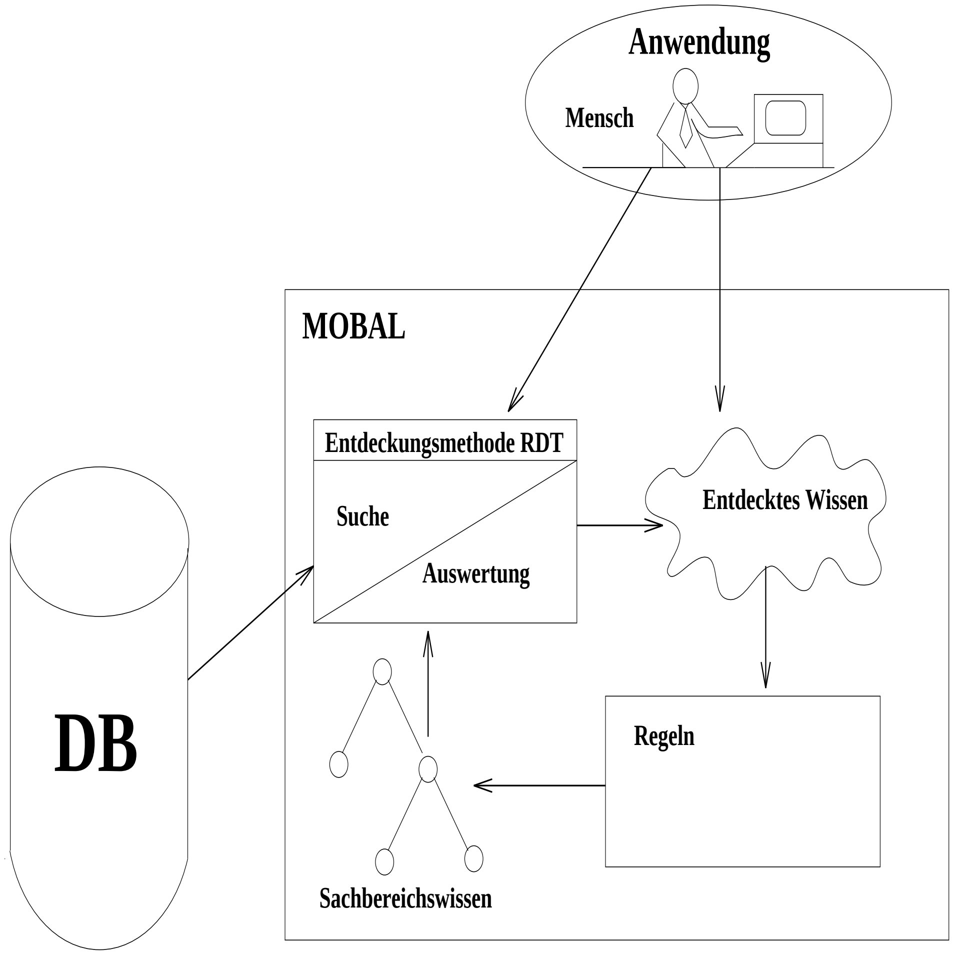

Abbildung 1.1: Einsatz–Szenario

1.3 Aufgabenbeschreibung

1.3.1 Lernen mit RDT über Datenbanken

In dieser Arbeit soll das Lernverfahren RDT (*rule discovery tool*) [Kietz und Wrobel, 1992] auf relationale Datenbanken angewendet und in das bestehende Wissensakquisitionssystem MOBAL als externes Tool integriert werden (RDT/DB). Ziel ist es, den Inhalt des Datenbanksystems als Beispielmenge für das logikbasierte Lernverfahren zugänglich zu machen.

Hierbei ist für das Sachbereichswissen zu prüfen, welche Arten von Informationen, die das System MOBAL zur Verfügung stellt, für das Lernen über relationale Datenbanken genutzt werden können.

1.3.2 MOBAL und Datenbanken

Weiterhin soll die Frage, ob die Anbindung einer Komponente von Mobal (RDT) an eine Datenbank sinnvoll ist, beantwortet werden, oder ob der Aufwand durch die Kommunikation über ein Netz und der Verwaltung der Daten in einem Datenbank–Managementsystem zu groß ist. Durch eine solche Anbindung soll versucht werden, Erkenntnisse für eine spätere Nutzung einer relationalen Datenbank als Wissensbasis zu erzielen.

1.4 Übersicht

In den folgenden Kapitel wird eine Einführung in die Thematik der relationalen Datenbanken gegeben und zwei Beispieldatenbanken vorgestellt, an denen in den folgenden Kapiteln verschiedene Sachverhalte veranschaulicht werden. Im Kapitel 3 wird dann das Lernverfahren RDT und die Verbindung zur induktiven logischen Programmierung beschrieben. Nach den theoretischen Grundlagen werden in Kapitel 4 verschiedene Repräsentationsmöglichkeiten vorgestellt, relationale Datenbanken in eine für RDT geeignete Prädikatendarstellung zu überführen. Das Kapitel 5, RDT/DB, enthalt die Diskussion, welche Teile der MOBAL–Wissensbasis zum Lernen mit RDT/DB ausgenutzt werden können und welche datenbankspezifischen Eigenschaften von RDT/DB zur Hypothesenraumeinschränkung ausgenutzt werden. Zum Abschluß des Kapitels 5 wird noch eine Idee vorgestellt, die das Laufzeitverhalten von RDT/DB wesentlich verbessert. Im anschließenden Kapitel 6 werden technische Details von RDT/DB vorgestellt, sowie Hinweise zur Nutzung von RDT/DB gegeben. In Kapitel 7 werden die Tests, die mit RDT/DB durchgeführt wurden, vorgestellt und analysiert. Hierzu werden die in Kapitel 2 vorgestellten Sachbereiche benutzt. Im abschliessenden Kapitel 8, Zusammenfassung und Ausblick, werden die Ziele dieser Arbeit den Lösungen gegenübergestellt und ein Ausblick zu dem bearbeiteten Themengebiet gegeben.

Kapitel 2

Relationale Datenbanken

Dieses Kapitel gibt eine Einführung in die Thematik der relationalen Datenbanken. Der erste Abschnitt beschreibt die Architektur eines solchen Datenbanksystems und der Abschnitt 2.2 das logische Modell der relationalen Datenbanken. Im Abschnitt 2.3 wird eine Charakterisierung der Datenbanksprache SQL gegeben, die auch zur Realisierung der vorliegenden Arbeit verwendet wurde. Abschließend wird der Leser mit zwei Beispieldatenbanken vertraut gemacht, an denen im weiteren Verlauf bestimmte Sachverhalte veranschaulicht werden und die der späteren Evaluierung dienen werden.

2.1 Architektur eines Datenbanksystems

Ein Datenbanksystem besteht aus zwei Komponenten, einer Datenbank und einem Datenbank–Managementsystem (DBMS).

Definition 3 (Datenbank)
Eine Ansammlung von Daten, die allen Anwendern eines Bereiches als gemeinsame Basis aktueller Informationen dient und entsprechend ihrer natürlichen Zusammenhänge strukturiert ist, bezeichnet man als Datenbank (DB) [Schlageter und Stucky, 1983, S.4].

Durch die Integration der Daten, die bisher meist nur einzelnen Anwendergruppen zugänglich waren, ist es notwendig, anwenderspezifische Sichten auf Teile der Datenbank zu ermöglichen. Jede Anwendergruppe soll nur Zugriff auf die Daten haben, die sie benötigt oder nutzen darf.

Hierzu wurde 1975 von der *ANSI/X3/SPARC Study Group on Database Management Systems* das 3–Ebenen–Strukturschema als Architektur für DBMS in relationalen Datenbanken vorgeschlagen, das auch in ORACLE 7 Anwendung findet (siehe Abbildung 2.1).

Die Ebenen der Architektur sind :

Externe Ebene: In der externen Ebene werden die einzelnen Sichtweisen der Anwendergruppen beschrieben.

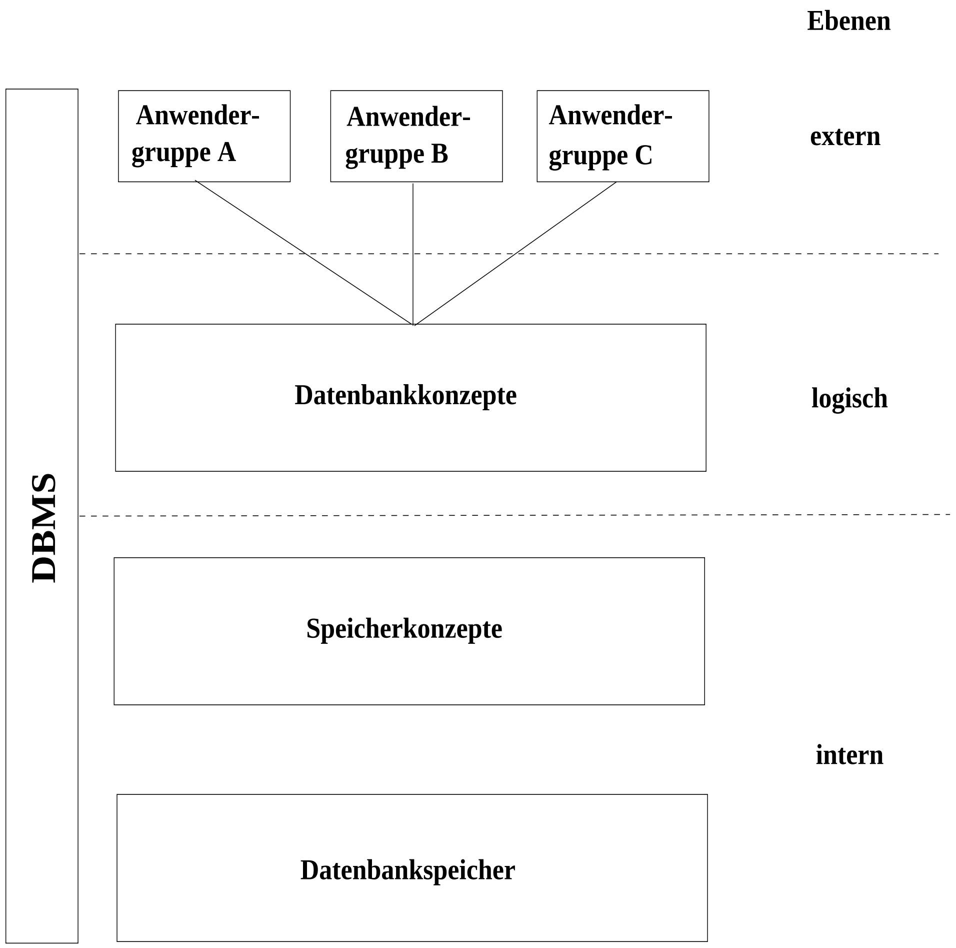

Abbildung 2.1: ANSI/SPARC 3–Ebenen Strukturschema

Logische Ebene: Die logische oder konzeptionelle Ebene beschreibt das Datenmodell, in dem die Objekte einheitlich und eindeutig beschrieben sind.

Interne Ebene: Auf der internen Ebene ist die physikalische Speicherstruktur aller Objekte konzipiert.

Auf die externe und interne Ebene wird hier nicht näher eingangen. Von wesentlicherer Bedeutung ist die logische Ebene, hier stehen die Informationen, die später für die Anwendung von RDT auf die Datenbank benötigt werden.

Die logische Ebene in ORACLE 7 wird durch das *data dictionary* beschrieben. Das *data dictionary* ist wie alle Datenbankinformationen in dem relationalen Modell dargestellt. Auf die im *data dictionary* enthaltenen Informationen hat der Benutzer im allgemeinen nur Leserechte. Diese Tabellen heißen in ORACLE 7 auch Systemtabellen. Sie bestehen aus Basistabellen und dem Benutzer verfügbaren Views[1] [Bobrowski, 1993].

Die Basistabellen basieren auf einer Gruppe von Tabellen, in denen Informationen über die zugeordnete Datenbank gespeichert sind. Diese Tabellen können nur von ORACLE 7 gelesen und geschrieben werden. Beispiele für Tabellen in relationalen Datenbanken sind in Abschnitt 2.4 dargestellt.

Die dem Benutzer verfügbaren Views, auch System–Views genannt, dienen der Zusammenfassung und übersichtlichen Darstellung der Informationen aus den Basistabellen. Die System–Views wandeln die Informationen in den Basistabellen in nutzbare Informationen.

Das *data dictionary* stellt beispielsweise folgende Informationen zur Verfügung:

- die Namen der Benutzer,
- Zugriffsberechtigungen, die jedem Benutzer zugeteilt wurden und
- Informationen über Integritätsbedingungen.

2.2 Das Relationenmodell

Das Relationenmodell wurde 1970 erstmals von Edgar F. Codd vorgestellt. Es ist heute Grundlage der meisten Datenbanksysteme, die kommerziell erhältlich sind. Hierzu zählt auch das Datenbanksystem ORACLE 7, das für diese Arbeit eingesetzt wurde.

Das mathematische Konzept, das dem relationalem Datenbankmodell zugrunde liegt, ist die Mengentheorie über Relationen, welche eine endliche Teilmenge des kartesischen Produkts über Wertebereiche ist [Ullmann, 1988, S. 43–65].

Jede Relation beschreibt eine Beziehung zwischen verschiedenen Mengen, die durch einen eindeutigen Namen gekennzeichnet ist. Diese Mengen sind durch Wertebereiche definiert. So ist zum Beispiel die Menge der ganzen Zahlen ein Wertebereich oder der Menge der Wochentage, die ein Wertebereich bestehend aus sieben einzelnen, festgelegten Werten ist. Werte dürfen nur atomar sein. Die in dem Datenbanksystem ORACLE 7 erlaubten Wertebereiche sind in Tabelle 2.1 aufgelistet.

Definition 4 (Relation)
Das kartesische Produkt der Wertemengen $D_1, \ldots, D_k$ ist die Menge von k–Tupeln $(v_1, \ldots, v_k)$, für die gilt, $\forall v_i : v_i \in D_i, 1 \leq i \leq k$.Eine Teilmenge dieses Produkts ist eine Relation. Ein Element einer Relation heißt Tupel.

[1] Views sind virtuelle Ausschnitte aus einer oder mehren Tabellen, die wiederum auch Views sein können.

Datentyp	Beschreibung	Bemerkung
CHAR(L)	Zeichenkette der festen Länge L	
VARCHAR2(L)	Zeichenkette mit max. Länge L	
NUMBER(P,S)	Numerische Daten variabler Länge. Max. Stelligkeit P und S Nachkommastellen.	S ist Optional
DATE	Datumsangaben fester Länge	
LONG	Zeichenketten variabler Länge	LONG–Daten sind Textdaten, die bei Portierungen konvertiert werden müssen.
RAW(L)	Binäre Daten mit max. Länge L	Die Daten werden nicht von ORACLE 7 interpretiert.
LONG RAW	Binäre Daten variabler Länge	Dient zum speichern von Grafiken,Ton, ...
ROWID	Binäre Daten, die Zeilenadressen darstellen	Interner Datentyp
MLSLABEL	Binäre Daten variabler Länge, die Labels des Betriebssystems darstellen.	Interner Datentyp

Tabelle 2.1: ORACLE 7 Datentypen

Eine anschauliche Darstellung ist die Sichtweise, Relationen als Tabellen zu betrachten, bei der jede Zeile ein Tupel und jede Spalte eine entsprechende Komponente der Relation ist. Die Spalten der Tabelle haben Namen, die eindeutig bzgl. der Relation sind, und heißen Attribute. Diese Terminologie wird von nun an verwendet, wenn von Relationen in einer Datenbank gesprochen wird. An dieser Stelle sei erwähnt, daß die Ausdrücke Relation und Tabelle im eigentlichen Sinne nicht synonym sind. Der Begriff der Relation kommt aus der Mathematik, und im Relationenmodell ist es eine spezielle Menge.

Da Relationen als Mengen definiert sind, kann in einer Relation auch kein Tupel mehrfach auftreten. Jedes Tupel der Relation ist eindeutig zu unterscheiden. Ein Tupel kann durch die gesamte Wertekombination der Attribute oder durch eine geeignete Teilmenge beschrieben werden, diese Beziehungen bezeichnet man als funktionale Abhängigkeiten (vgl. Definition 5). Eine solche Kombination von Attributen nennt man Relationenschlüssel oder Schlüsselkandidat (vgl. [Date, 1990, S. 276], [Finkenzeller *et al.*, 1989]).

Definition 5 (Funktional Abhängig)
Sei $R(A_1, \ldots, A_n)$ *eine Relation und es gelte* $X, Y \subseteq \{A_1, \ldots, A_n\}$.
Y *ist funktional abhängig von* X $(X \to Y)$, *wenn jedem* X *ein* Y *eindeutig zugeordnet werden kann.*

Von diesen Schlüsselkandidaten gibt es wiederum ausgezeichnete Elemente, die Primärschlüssel.

Definition 6 (Primärschlüssel)
Sei R *eine Relation mit den Attributen* $(A_1, \ldots, A_k)$ *und* $(A_j, \ldots, A_l)$ *eine minimale Attributkombination.* $(A_j, \ldots, A_l)$ *ist Primärschlüssel von* R, *wenn gilt:* $(A_j, \ldots, A_l) \to (A_1, \ldots, A_k)$.

Definition 7 (Fremdschlüssel)
Sei $(A_i, \ldots, A_s)$ *eine Attributkombination einer Relation R2, die gleichzeitig Primärschlüssel einer Relation R1 ist.* $(A_i, \ldots, A_s)$ *heißt dann Fremdschlüssel der Relation R2.*

Weiterhin gehören zu einem relationalen Modell noch eine Reihe von Integritätsbedingungen:

Entitäts–Integrität: Jede Relation besitzt einen Primärschlüssel.

Referentielle Integrität:
Jeder Fremdschlüssel verweist auf einen Primärschlüssel einer anderen Relation.

Benutzerdefinierte–Integrität: Der Benutzer kann einzelne Wertebereiche definieren.

Diese theoretisch vorgeschriebenen Integritätsvorschriften werden bisher von fast keinem Datenbank–Managementsystem (DBMS) in vollem Umfang realisiert [Finkenzeller *et al.*, 1989, S.46].

Auch die aktuelle Version des Datenbanksystems ORACLE 7 erfüllt diese Integritätsvorschriften nicht vollständig. Beispielsweise überprüft das DBMS von ORACLE 7 nicht die Definition eines Primärschlüssels bei der Erstellung einer Tabelle, d.h. es können Tabellen ohne Primärschlüssel erstellt werden.

Aus diesem Grunde hat Codd 1983 die Anforderungen an ein minimales relationales Datenbanksystem aufgestellt, welches die folgenden Eigenschaften erfüllt muß [Lockemann und Schmidt, 1987]:

1. Alle Informationen sind durch Werte repräsentiert, die in Form von Tabellen dargestellt werden können.

2. Der Benutzer kennt keine Verweisstrukturen zwischen den einzelnen Tabellen.

3. Es sind wenigstens die Operatoren zur Selektion, Projektion und Verbindung von Tabelleneinträgen definiert.

Ein vollständiges relationales Datenbanksystem muß nach Codd noch die schon beschriebenen Integritätsbedingungen automatisch kontrollieren und weitere Operatoren zur Datenänderung bereitstellen.

2.3 Die Datenbanksprache SQL

SQL ist die von ORACLE 7 verwendete Datenbanksprache. Dieser Abschnitt soll und kann keine Einführung in die Datenbanksprache SQL geben. Hier soll dem Leser, der nicht mit SQL vertraut ist, eine Charakterisierung der Datenbanksprache gegeben werden.

SQL (Standard Query Language) wurde 1986 erstmal von dem ANSI[2] als ein Standard für relationale Datenbanksprachen definiert und 1987 von der ISO[3] übernommen [Date, 1990, S. 28].

Dieser Standard wurde 1989 in der ISO 9075:1989 erweitert, trotzdem fehlten noch wichtige Eigenschaften, wie sie von Codd für ein relationales System gefordert wurden. Eine wesentlich erweiterte Fassung der ISO 9075 wurde 1992 festgelegt [Weber, 1993]. Das in ORACLE 7 implementierte SQL erfüllt diesen Standard.

Allgemein unterscheidet man bei relationalen Datenbanksprachen zwischen:

Prädikatenkalkülsprachen beschreiben eine Ergebnisrelation durch Eigenschaften (Prädikate), die die Elemente der Ergebnisrelation erfüllen muß.

Algebraische Sprachen beschreiben eine Ergebnisrelation als Resultat von Mengenoperationen auf den Relationen der Datenbank.

SQL ist eine tupelorientierte Prädikatenkalkülsprache, die aber auch algebraische Elemente enthält [Lockemann und Schmidt, 1987, S. 605–606]. SQL ist somit nichtprozedural und mengenorientiert.

SQL läßt sich in drei Sprachteile aufteilen:

1. Eine Sprache für die Definition von Objekten (**D***ata* **D***efinition* **L***anguage* (DDL))

2. Eine Sprache zur Manipulation der Objekte (**D***ata* **M***anipulation* **L***anguage* (DML))

3. Eine Sprache zur Kontrolle der Privilegien (**D***ata* **C***ontrol* **L***anguage* (DCL))

---

[2] American National Standard Institute, Dokument X3.135-1986

[3] International Organization for Standardization, Dokument ISO/TC97/SC21/WG3 N117

2.4 Beispiele relationaler Datenbanken

An dieser Stelle werden zwei Beispiele relationaler Datenbanken vorstellen. Diese zwei Beispiele sollen im weiteren Verlauf der Arbeit zur Veranschaulichung verschiedener Sachverhalte dienen und später auch die Szenarien für die Tests mit RDT/DB sein. Als erstes wird in Abschnitt 2.4.1 der aus dem maschinellem Lernen bekannte KRK–Sachbereich vorgestellt. Dieser Sachbereich wurde aufgrund seiner Überschaubarkeit und relativen Einfachheit ausgewählt. Weiterhin ist dieser Sachbereich ein oft verwendetes Testszenario im maschinellen Lernen, so daß man Vergleiche mit anderen Verfahren anstellen kann. So wurde der KRK–Sachbereich zu Vergleichtests der Lernverfahren LINUS und FOIL [Lavrač und Džeroski, 1992] und RDT und FOIL [Lindner und Robers, 1994] verwendet.

Aus der Sicht des *knowledge discovery in databases* ist diese Anwendung allerdings nicht akzeptabel, denn hier wird besonderer Wert auf die „reale“Anwendung gelegt. Um diesem Punkt gerecht zu werden wurde ein zweites Beispiel aus dem Bereich der Roboternavigation (siehe Abschnitt 2.4.2) ausgewählt.

2.4.1 KRK–Sachbereich

Die KRK–Schachendspielsituation wurde als Beispiel für das maschinelle Lernen zuerst von Quinlan beschrieben [Quinlan, 1983]. KRK steht für king–rook versus king, d.h. es spielen der weiße König und der weiße Turm gegen den schwarzen König. Das Lernproblem besteht darin, die illegalen Stellungen der drei Figuren auf dem Schachbrett von den legalen abzugrenzen, d.h. der Begriff `illegal` soll gelernt werden. Es existieren insgesamt 64^3 (=262 144) mögliche Stellungen, von denen (unter der Annahme, daß Weiß am Zug ist) ca. 33% als `illegal` zu klassifizieren sind.

Man kann drei Arten illegaler Stellungen unterscheiden:

1. Ein Feld ist mit mehr als einer Figur belegt.

2. Die beiden Könige sind unmittelbar benachbart.

3. Der schwarze König steht im Schach, d.h. er steht entweder auf derselben Spalte oder derselben Zeile wie der weiße Turm, wobei der weiße König nicht zwischen den beiden steht.

KRK in einer relationalen Datenbank

Der KRK–Sachbereich ist in der Datenbank durch die Relation illegal (siehe Tabelle 2.2) mit ca. 3500 Einträgen repräsentiert. Das Schachbrett ist hierfür in ein numerisches Koordinatensystem eingeteilt. Das Attribut `ID` ist ein künstlicher Primärschlüssel, der zur eindeutigen Identifizierung der einzelnen Beispiele eingefügt wurde.

white king x	white king y	white rook x	white rook y	black king x	black king y	ID
4	5	2	2	2	4	452224
...	...	...	...	...	...	...

Tabelle 2.2: illegal mit ID

x	y
1	1
1	2
...	...

Tabelle 2.3: adjacent

Weiterhin ist in der Datenbank als Hintergrundwissen die Relation adjacent (siehe Tabelle 2.3) enthalten. Die Relation adjacent beschreibt, daß zwei Felder benachbart sind, und ist in der Datenbank vollinstanziiert.

2.4.2 Sachbereich der Roboternavigation

Das Beispiel zur Roboternavigation ist aus dem aktuellen Forschungsprojekt B-Learn II[4] übernommen. In dem Szenario geht es um die Bewegung eines Roboters in einem einfachen Raum (siehe Bild 2.2), in dem er sich durch die Messungen seiner 24 Distanzultraschallsensoren orientieren soll. Hierzu werden die Messungen auf verschiedenen Ebenen zu Definitionen von Begriffen abstrahiert. Eine genaue Beschreibung der verschiedenen Abstraktionsebenen geben Katharina Morik und Anke Rieger in [Morik und Rieger, 1993].

Von den 24 Sensoren des Roboters (siehe Abbildung 2.3) sind jeweils drei auf jeder Seite und jeder Ecke beangebracht.

Jede Messung liefert für jeden Sensor die Distanz zu dem ihm am nächst gelegensten Objekt. Für den Gesamtzeitraum werden die Messungen der einzelnen Sensoren in Intervalle unterteilt, in denen sich die Meßwerte linear ändern. Diese Intervalle werden *basic features* genannt. Das Faktum `decreasing(t24,Or,s5,14,15,-22)` beschreibt z.B., daß im Pfad mit der Kennzeichnung `t24` der Sensor `s5` mit der Orientierung `Or` von dem Zeitpunkt `14` bis zum Zeitpunkt `15` die Distanz zu dem nächsten Objekt linear mit der Steigung `-22` kleiner geworden ist. Für den Trace 24 und den in Fahrtrichtung auf der linken Seite befindlichen Sensor 5 erhält man folgende Sequenz von *basic features* (siehe [Klingspor, 1994]).

- `no_measurement(t24,Or,s5,1,14,_)`
- `decreasing    (t24,Or,s5,14,15,-22)`

[4] Das Projekt B-Learn II (P7274) wird von der EG und dem Forschungsministerium von NRW finanziert.

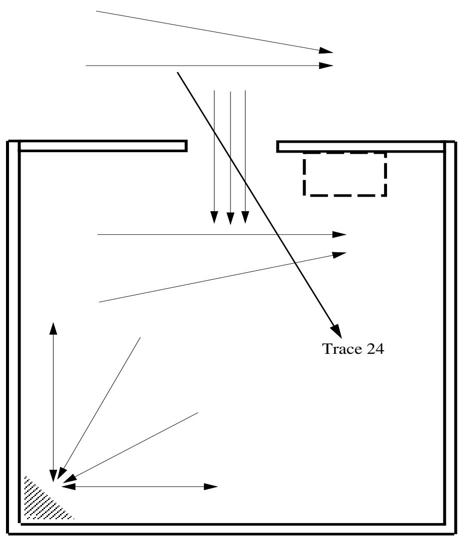

Abbildung 2.2: Roboterbewegung in einem Raum

trace	s_orientation	sensor	start_time	end_time	grad
t24	33	s5	14	15	-22
t24	33	s5	22	26	-30
t9	321	s20	3	5	-33
t9	336	s21	1	4	-36
...	...	...	...	...	...

Tabelle 2.4: *basic feature* decreasing

- `incr_peak     (t24,Or,s5,15,16,47)`
- `decreasing    (t24,Or,s5,16,18,-30)`
- `no_measurement(t24,Or,s5,18,22,_)`
- `decreasing    (t24,Or,s5,22,26,-30)`

Die *basic features* haben als Argumente die Tracenummer[5], die Orientierung des Sensors, die Sensorbezeichnung, Start- und Endzeit der Messung und den Gradienten (siehe auch Tabelle 2.4). Jedes *basic feature* ist als Relation in der Datenbank repräsentiert, insgesamt gibt es 9 verschiedene *basic features.* Neben den *basic features* gibt es noch *sensor features*, die eine von einem Sensor gemessene Kantenkonstellation des Raums beschreiben.

Späteres Lernziel wird es sein, Abfolgen von *basic features* zu lernen, die ein *sensor feature* beschreiben. D.h. ein *sensor feature* beschreibt das Verhalten

[5] Schlüsselattribute sind unterstrichen.

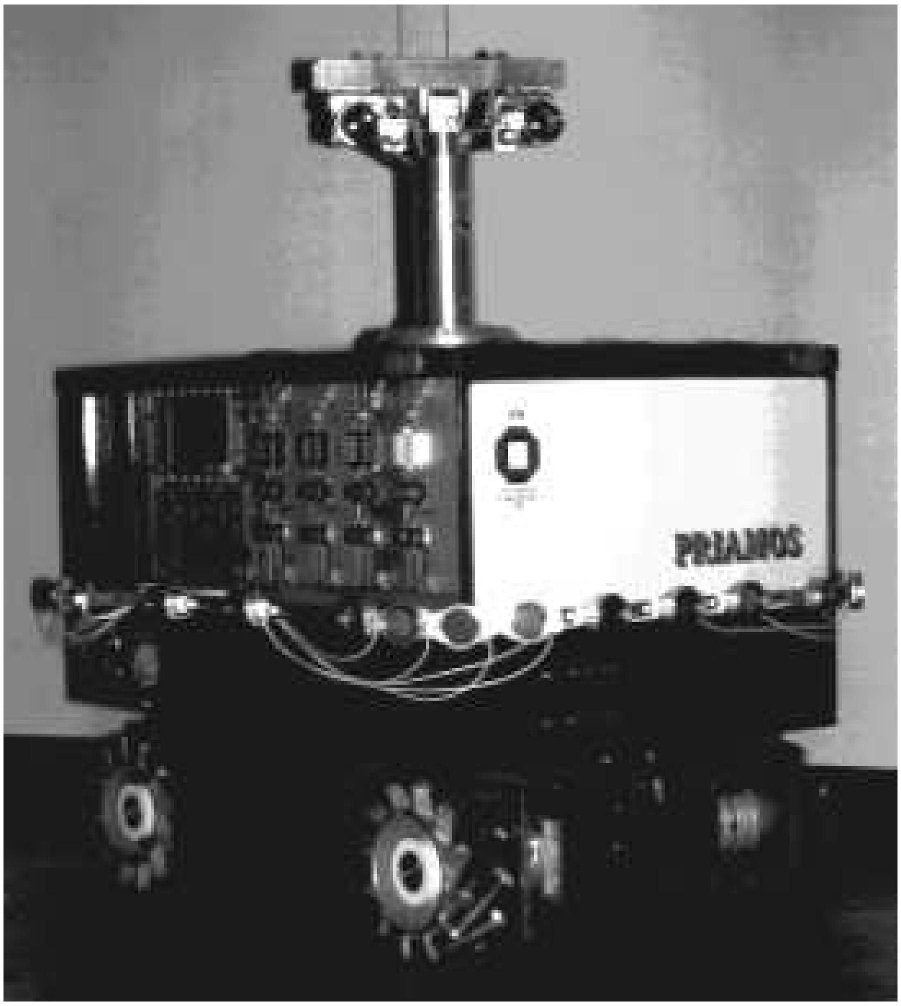

Abbildung 2.3: Der Roboter PRIAMOS.

eines Sensors vom Zeitpunkt t1 bis tn. Insgesamt gibt es folgende vier *sensor features*.

1. s_jump
2. s_line
3. s_convex
4. s_concave

In den Abbildungen 2.4 – 2.7 sind Beispiele für die einzelnen Kantenkonstellationen, die die verschiedenen *sensor features* beschreiben, dargestellt. Die Tabelle 2.5 zeigt wie die *sensor features*, hier am Beispiel s_jump, in der Datenbank repräsentiert sind. Die Argumente der *sensor features* sind die

trace	sensor	start_time	end_time	m_orientation
t24	s23	1	21	diagonal
t24	s3	11	17	diagonal
t9	s16	11	26	parallel
t9	s20	4	26	parallel
...	...	...	...	...

Tabelle 2.5: *sensor feature* s_jump

Tracenummer, die Sensorbezeichnung, die Start- und Endzeit der Abfolge und die Bewegungsrichtung in Bezug zu der Kantenkonstellation.

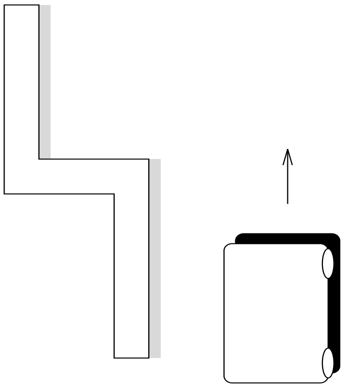

Abbildung 2.4: *sensor feature:* s_jump

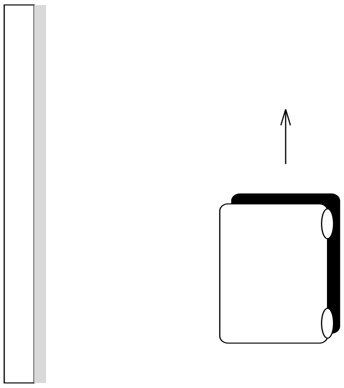

Abbildung 2.5: *sensor feature:* s_line

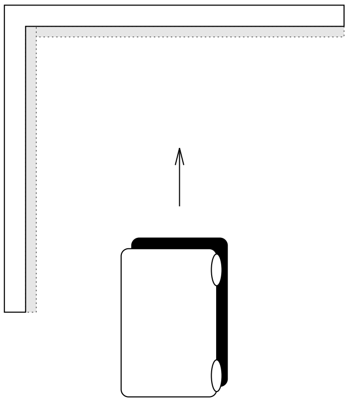

Abbildung 2.6: *sensor feature:* s_convex

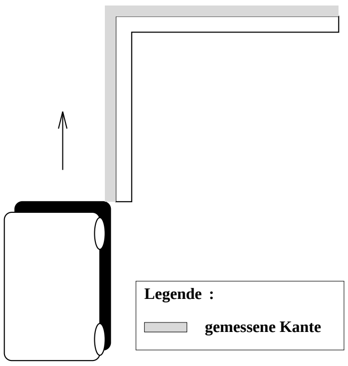

Abbildung 2.7: *sensor feature:* s_concave

Kapitel 3

Das Lernverfahren RDT

Bevor auf die Besonderheiten beim Lernen mit RDT über relationale Datenbanken eingangen wird, soll in diesem Abschnitt das Verfahren RDT aus der Sicht des maschinellen Lernens charakterisiert und beschrieben werden. Im Abschnitt 3.1 wird eine Einführung in die induktive logische Programmierung gegeben, damit in dem folgenden Abschnitt gezeigt werden kann, daß RDT ein Verfahren der induktiven logischen Programmierung ist. Im Abschnitt 3.3, Hypothesenraumeinschränkungen, werden dann die Ideen von RDT beschrieben. Abschließend wird der Algorithmus von RDT beschrieben und RDT nach den Punkten aus Abschnitt 3.1.3 beurteilt.

3.1 Induktive logische Programmierung

Stephen Muggleton beschreibt das Gebiet induktive logische Programmierung[1] als die Schnittmenge zwischen logischer Programmierung und maschinellem Lernen [Muggleton, 1992, S. 3-21]. Die Zielsetzung des induktiven logischen Programmierens kommt aus dem induktiven maschinellen Lernen. Informell kann ILP wie folgt beschrieben werden: Ziel ist es, mittels Induktion eine Theorie in einer eingeschränkten Prädikatenlogik 1.Stufe, ausgehend von Beispielen und Hintergrundwissen, zu lernen.

Die zwei zentralen Punkte im ILP sind die Verwendung von Hintergrundwissen und der Tatsache, daß das Lernergebnis in einer eingeschränkten Prädikatenlogik 1. Stufe vorliegt [Lavrač und Džeroski, 1994, S.13].

Formal wird ILP folgendermaßen definiert:

Definition 8 (ILP)

Gegeben seien eine Sprache $\mathcal{L}_{\mathcal{E}}$, eine Menge positiver und negativer Beispiele, $\mathcal{E}^+ \cup \mathcal{E}^- = \mathcal{E}$, eine Hypothesensprache $\mathcal{L}_{\mathcal{H}}$ in einer eingeschränkten Prädikatenlogik 1.Stufe und Hintergrundwissen $\mathcal{B}$, $\mathcal{B} \not\models \mathcal{E}^+$.

[1] In dieser Arbeit steht induktive logische Programmierung immer für den Forschungsbereich und die Abkürzung ILP für die technische Betrachtung des Bereichs.

ILP ist das Problem, eine Hypothese $\mathcal{H} \subseteq \mathcal{L}_{\mathcal{H}}$zu finden für die gilt:

1. *Konsistenz :* $\mathcal{B} \wedge \mathcal{E} \wedge \mathcal{H} \not\models \bot$
2. *Vollständigkeit:* $\mathcal{B} \wedge \mathcal{H} \models \mathcal{E}^{+}$
3. *Korrektheit :* $\mathcal{B} \wedge \mathcal{H} \not\models \mathcal{E}^{-}$

3.1.1 Hintergrundwissen

Lernen mit Hintergrundwissen ähnelt der natürlichen Lernsituation. Mit bereits bekanntem Wissen und aktuellen Beispielen für einen Sachverhalt versucht man, eine Theorie zu diesem Sachverhalt zu finden. Für die einzelnen Lernverfahren ist es deshalb von zentraler Bedeutung, in welcher Form und in welchem Umfang Hintergrundwissen zur Verfügung steht. Durch die Hypothesensprache $\mathcal{L}_{\mathcal{H}}$, das Hintergrundwissen und die Beispiele wird der Hypothesenraum beschrieben. Vom Hintergrundwissen ist es abhängig, wie leicht ein Begriff zu lernen ist. Durch zusätzliches Hintergrundwissen kann eine kürzere und verständlichere Beschreibung für den zu lernenden Begriff gefunden werden. Zu umfangreiches Hintergrundwissen kann allerdings auch hinderlich für den Lernvorgang sein, da zusätzliche Prädikate berücksichtigt werden müssen. Dadurch wird ein größerer Hypothesenraum beschrieben. In vielen ILP–Systemen wird die Repräsentation des Hintergrundwissens auf Grundfakten beschränkt [Lavrač und Džeroski, 1994, S.13].

3.1.2 Hypothesensprache $\mathcal{L}_{\mathcal{H}}$

Aus dem Bereich der logischen Programmierung kommt der Repräsentationsformalismus für das Lernergebnis in der induktiven logischen Programmierung. Eine eingeschränkte Prädikatenlogik als Hypothesensprache zu verwenden, hat einige Vorteile. Neben der direkten Interpretierbarkeit des Lernergebnisses durch ein System, ist die Ausdrucksstärke im Vergleich zu Standardverfahren des maschinellen Lernens, wie den TDIDT[2]–Verfahren, ein Argument für die induktive logische Programmierung. ILP – Verfahren sind in der Lage, Relationen auszudrücken. Mit dem Repräsentationsformalismus konnten auch die Techniken aus dem Bereich der logischen Programmierung übernommen werden [Muggleton und De Raedt, 1993].

3.1.3 ILP–Charakteristiken

Die einzelnen Verfahren des ILP werden üblicherweise nach den folgenden Punkten charakterisiert (siehe [Muggleton und De Raedt, 1993; Lavrač und Džeroski, 1994]) :

[2] *top–down induction of decision tree*

Inkrementell / empirisch
Hier wird nach der Vorgehensweise in der Behandlung der Beispiele unterschieden. Inkrementelle Verfahren bekommen von dem Benutzer Schritt für Schritt die Beispiele vorgeben, während bei empirischen Verfahren die Beispielmenge beim Start des Verfahrens vorgegeben ist und sich nicht mehr ändert. Ein klassisches inkrementelles Verfahren ist MIS von Shapiro [Shapiro, 1981], während FOIL [Quinlan, 1990] ein Beispiel für ein empirisches Verfahren ist.

Single / Multiprädikatslernen
Singleprädikat Lerner können in einem Lernlauf nur eine einzige Begriffsdefinition lernen, während beim Multiprädikatslernen die Menge der Zielbegriffe nicht beschränkt ist.

Deklarativer Bias: Syntaktischer / semantischer Bias
Der deklarative Bias definiert den Aufbau des Hypothesenraums, der von dem Lerner betrachtet wird. Beim deklarativen Bias unterscheidet man zwischen syntaktischem und semantischem Bias. Ein syntaktischer Bias beschränkt den Hypothesenraum in der Form der erlaubten Hypothesen. Als Gegenstück zum syntaktischen Bias bestimmt ein semantischer Bias den Aufbau der Hypothesen aufgrund der gegebenen Prädikate aus den Beispielen und dem Hintergrundwissen [Muggleton und De Raedt, 1993].

3.2 Lernaufgabe in RDT

RDT [Kietz und Wrobel, 1992] ist Bestandteil des Wissensakquisitionssystems MOBAL [Morik *et al.*, 1993] und wurde an der Gesellschaft für Mathematik und Datenverarbeitung (GMD) entwickelt.

Die Lernaufgabe von RDT wird wie folgt beschrieben [Morik *et al.*, 1993, S. 169 -191]:

Gegeben: Hintergrundwissen $\mathcal{B}$ und eine Menge positiver und negativer Beispiele, $\mathcal{E}^+ \cup \mathcal{E}^- = \mathcal{E}$, für einen zu lernenden Begriff C in einer funktionsfreien Klausellogik.

Ziel: Finde eine Hypothese $\mathcal{H}$ in funktionsfreier Klausellogik für die gilt:

1. die Hypothese $\mathcal{H}$, das Hintergrundwissen $\mathcal{B}$ und die Menge der Beispiele $\mathcal{E}$ sind konsistent.
2. Aus dem Hintergrundwissen $\mathcal{B}$ und der Hypothese $\mathcal{H}$ folgen die positiven Beispiele, aber keine negativen.
3. $\mathcal{H}$ ist die generellste Hypothese.

Die Repräsentationssprache $\mathcal{L}$ für RDT in MOBAL ist eine funktionsfreie Klausellogik[3], d.h. die Beispielsprache $\mathcal{L}_{\mathcal{E}}$, die Hypothesensprache $\mathcal{L}_{\mathcal{H}}$ und die Sprache für das Hintergrundwissen haben denselben Repräsentationformalismus. Für den von RDT verwendeten Repräsentationsformalismus ist die logische Inferenz entscheidbar. Beim eigentlichen Lernvorgang berücksichtigt RDT allerdings nur Beispiele und Hintergrundwissen in Form von Grundfakten. Durch die Inferenzmaschine von MOBAL werden alle möglichen Grundfakten mit den gegebenen Regeln aus den bekannten Fakten abgeleitet, innerhalb der vorgegebenen Tiefenbeschränkung der Inferenzmaschine.

RDT erfüllt also die Anforderungen aus der Definition 8 und ist folglich ein ILP–Verfahren.

Trotz der Einschränkung der Hypothesensprache auf funktionsfreie Klausellogik, wird immer noch ein sehr großer Hypothesenraum durch $\mathcal{L}_{\mathcal{H}}$ beschrieben, so daß weitere Einschränkungen gemacht werden müssen.

3.3 Hypothesenraumeinschränkung

Mit RDT wird der modellbasierte Ansatz im maschinellen Lernen verfolgt, d.h. es wird von RDT kein fester Hypothesenraum definiert. Stattdessen gibt der Benutzer zusätzliches Wissen über die Form der möglichen Hypothesen vor. Hier berücksichtigt RDT drei Arten von Modellwissen.

1. Regelmodelle
2. Prädikatentopologien
3. Sorten

Diese drei Arten von Modellwissen können von dem Benutzer explizit vorgegeben oder durch bestimmte Tools von MOBAL auf Basis des bekannten Sachbereichswissens berechnet werden.

3.3.1 Regelmodelle

Bei RDT wird der Hypothesenraum durch die Vorgabe von Schemata in Form von Regeln 2. Stufe eingeschränkt. Diese definieren die syntaktische Form der möglichen Regeln. Ein Regelmodell[4] ist also eine Regel, die anstatt der Sachbereichsprädikate Prädikatvariablen enthält. Diese Regelmodelle können durch den Benutzer vorgegeben werden oder durch die Systemkomponente MAT (*model aquisition tool*) berechnet werden.

[3] Der genaue Repräsentationsfromalismus ist in [Morik *et al.*, 1993, S. 27 - 69] beschrieben

[4] In MOBAL werden Regelmodelle Metaprädikat genannt.

Beispiel: Um die Regel

tochter(A,B) & tochter(B,C) $\rightarrow$ grossmutter(A,C)

mit RDT lernen zu können, muß das Regelmodell

```
mx(P1,Q): P1(A,B) & P1(B,C) --> Q(A,C)
```

von dem Benutzer als weitere Information vorgegeben werden. Dabei bedeutet `mx(P1,Q)` das Foldende: Das Regelmodell heißt mx und beschreibt eine Beziehung zwischen den Prädikatvariablen `P1` und `Q`.

In der Abbildung 3.1 sind weitere Regelmodelle dargestellt.

```
m1(P1,Q):P1(X)-->Q(X).
m2(P1,P2,Q):P1(X) & P2(X)-->Q(X).
m3(P1,P2,P3,Q):P1(X) & P2(X) & P3(X)-->Q(X).
m4(P1,P2,P3,P4,Q):P1(X) & P2(X) & P3(X) & P4(X)-->Q(X).
m5(P1,Q):P1(X,Y)-->Q(X).
m6(P1,P2,Q):P1(X) & P2(X,Y)-->Q(X).
m7(P1,P2,P3,Q):P1(X) & P2(X) & P3(X,Y)-->Q(X).
m8(P1,P2,P3,Q):P1(X) & P2(X,Y) & P3(X,Z)-->Q(X).
m9(P1,P2,P3,P4,Q):P1(X) & P2(X) & P3(X) & P4(X,Y)-->Q(X).
m10(P1,P2,P3,P4,Q):P1(X) & P2(X) & P3(X,Y) & P4(X,Z)-->Q(X).
```

Abbildung 3.1: Regelmodelle

Im folgenden stehen kleine griechische Buchstaben für die Substitution von Termvariablen und große griechische Buchstaben für die Substitution von Prädikatvariablen. Eine Instanziierung Σ ist eine endliche Menge von Paaren P/p. Hierbei ist P eine Prädikatvariable und p ein Prädikatsymbol. Die Stelligkeit von beiden ist gleich. Die Menge aller p heißt der Bereich von Σ ($range(\Sigma)$), während $R\Sigma$ die Instanziierung des Regelmodells R mit Σ beschreibt. Ein Regelmodell R ist grundinstanziiert oder auch vollinstanziiert ($ground(R\Sigma)$), wenn alle Prädikatvariablen mit Prädikatsymbolen belegt sind. Weiterhin seien $\mathcal{P}$ die Menge aller Sachbereichsprädikate und $\mathcal{R}$ die Menge aller vorgegebenen Regelmodelle.

Der durch die Regelmodelle $\mathcal{R}$ beschriebene Hypothesenraum von RDT läßt sich dann als Menge beschreiben:

$$\mathcal{H} = \{R\Sigma \mid R \in \mathcal{R} \ \wedge \ range(\Sigma) \subseteq \mathcal{P} \ \wedge \ ground(R\Sigma)\}$$

Die Regelmodelle werden nach einer erweiterten Form der θ–Subsumtion [Plotkin, 1970] ihrer Allgemeinheit nach partiell geordnet. Hierbei setzen Jörg Uwe Kietz und Stefan Wrobel nicht die ursprüngliche Bedeutung der θ–Subsumtion ein, da Regelmodelle weder wahr noch falsch sein können; nur vollinstanziierte Regelmodelle können wahr oder falsch sein [Kietz und Wrobel, 1992]. Deshalb ist die Generalisierungsbeziehung $\geq_{RS}$ zwischen Regelmodellen folgendermaßen definiert [Morik *et al.*, 1993, S.182]:

Definition 9 (Generalisierungsbeziehung $\geq_{RS}$)
Seien R und R' Regelmodelle, σ eine Substitution bezüglich Termvariablen und Σ eine Substitution bezüglich Prädikatvariablen, so daß Σ verschiedene Prädikatvariablen nicht gleichsetzt. R ist genereller als R', $R \geq_{RS} R'$, genau dann wenn $R\sigma\Sigma \subseteq R'$.

In der Abbildung 3.2 ist die Ordnung für die in Abbildung 3.1 gezeigten Regelmodelle dargestellt. Die Pfeile zeigen auf die nächst generelleren Regelmodelle.

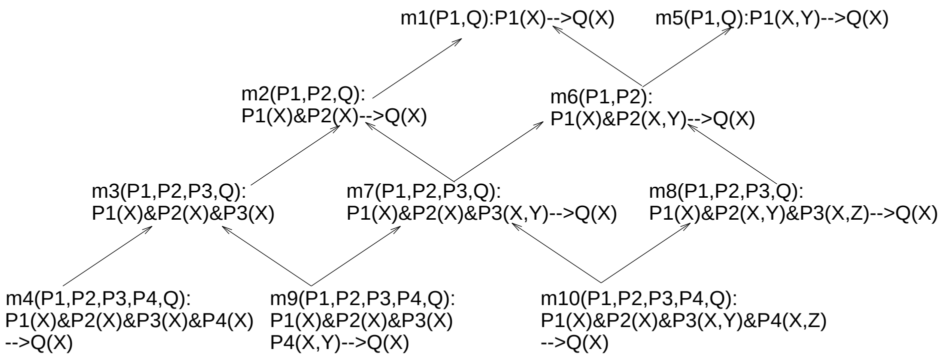

Abbildung 3.2: Generalisierungsbeziehung zwischen Regelmodellen[5].

Die Regelmodelle werden von RDT schrittweise instanziiert und jeweils getestet. RDT beginnt beim Lernen mit dem generellsten Regelmodell und geht dann zu den spezielleren (Top–Down–Strategie). Hierdurch ergibt sich eine weitere Einschränkung des Hypothesenraums. Spezialisierungen von akzeptierten oder gemäß dem Akzeptanzkriterium als zu speziell bewerteten Hypothesen müssen nicht mehr getestet werden. Die Reihenfolge, in der die Prädikatvariablen der Regelmodelle instanziiert werden, erfolgt nach der *relation chain* der Attributvariablen der einzelnen Prädikatvariablen.

Die *relation chain* beschreibt die Verknüpfung der Attributvariablen in einem Regelmodell. Es werden nur Prädikatvariablen instanziiert, die in einer solchen Verknüpfungsbeziehung mit den bereits belegten Prädikatvariablen stehen. An einem Beipiel soll der Sachverhalt veranschaulicht werden:

[5] Quelle: [Kietz und Wrobel, 1992]

Beispiel: Gegen sei folgendes Regelmodell:

```
m_example(P1,P2,C): P1(Y) & P2(X,Y) --> C(X)
```

Es macht keinen Sinn nach einer Belegung für `P1` zu suchen, ohne das vorher `P2` instanziiert wurde. `P1` steht in keiner direkten Beziehung zu `C`. Jedes Prädikat mit der Stelligkeit von `P1` könnte instanziiert werden ohne eine Auswirkung auf das Akzeptanzkriterium (siehe Abschnitt 3.4).

Definition 10 (*relation chain)*
Sei $\mathcal{P}(X_1, \ldots, X_m)$ eine Menge von Prädikaten, die von den Attributen $X_1, \ldots, X_m$ abhängen. Weiterhin sei dann $\mathcal{H}$ eine Hypothese der Form $\mathcal{H} : \mathcal{P}(X_1, \ldots, X_m) \rightarrow q(Y_1, \ldots, Y_n)$, $\forall\ Y_i : Y_i \in (X_1, \ldots, X_n), 1 \leq i \leq n$. Dann ist die Verknüpfung der Variablen X_i mit der Konklusion q folgendermaßen über die relation chain*$(rc(X_i))$ definiert:*

$$rc(X_i) = \begin{cases} \emptyset & | \quad X_i \in \{Y_1, \ldots, Y_n\} \\ [P_t, rc(X_j)] & | \quad sonst \end{cases}$$

Hierbei ist $P_t(X_l, \ldots, X_k) \in \mathcal{P}(X_1, \ldots, X_m)$,und es gilt: $X_i, X_j \in \{X_l, \ldots, X_k\}, X_i \neq X_j$ und $rc(X_j)$.

In dem obrigen Beispiel ist die *relation chain* von X, $rc(X) = \emptyset$, und von Y, $rc(Y) = [P2]$.

Eine Variable kann nach Definition 10 mehrere *relation chains* besitzen. Aus diesem Grund ist über die *relation chains* noch eine Ordnung $\leq_P$ definiert (vgl. [Morik *et al.*, 1993, S. 184], nach der die Prädikate instanziiert werden.

Definition 11 (Ordnung $\leq_P$)
Seien $length(L)$ eine Funktion, die die Länge einer Liste berechnet, und $\alpha(X) = min(\{length(rc(X))\})$ der Abstand der Variablen X zur Konklusion. Dann läßt sich eine Ordnung über den zu instanziierenden Prämissen PR über den Abstand der Variablen der Prämisse zur Konklusion definieren:

$$PR \leq_P PR',\ gdw. \quad \begin{array}{l} min(\{\alpha(X) \mid X \text{ ist Attribut von } PR\}) \leq \\ min(\{\alpha(X) \mid X \text{ ist Attribut von } PR'\}) \end{array}$$

Bisher wurde gezeigt, wie der Hypothesenraum für RDT definiert ist und vollständig durchsucht wird. Weiterhin wurde gezeigt, in welcher Reihenfolge die Prädikatvariablen der Regelmodelle instanziiert werden. Jetzt bleibt die Frage: Mit welchen Prädikaten können die Prädikatvariablenbelegt werden? Neben der Stelligkeit der Prädikate werden von RDT dazu die in Abschnitt 3.3 erwähnte Prädikatentopologie und die Sorten verwendet.

3.3.2 Prädikatentopologie

Durch die in Abschnitt 3.3.1 beschriebenen Regelmodelle wird der Hypothesenraum syntaktisch eingeschränkt. Eine semantische Einschränkung in RDT ergibt sich aus der Prädikatentopologie. Prädikate können strukturiert in Form eines Graphen repräsentiert werden, dessen Knoten Prädikate zugeordnet sind. Ein solcher Graph ist eine Abstraktion eines Regelgraphen, indem durch die Vereinigung ähnlicher Prädikate zu Prädikatsklassen lokale Abhängigkeiten vernachlässigt wurden. Die Knoten eines solchen Graphen sind somit Prädikatsklassen, und die Verbindungen zwischen den einzelnen Knoten stehen für inferenzielle Abhängigkeiten. Ein solcher Graph heißt in MOBAL Topologie oder Prädikatentopologie. Es werden in RDT nur Prädikate instanziiert, wenn die resultierende Hypothese kompatibel zur Prädikatentopologie ist.

Definition 12 (Topologiekompatibilität)
Eine Hypothese ist mit einer Topologie kompatibel, wenn die Prädikate der Prämisse in einem Vorgängerknoten oder im Knoten des Prädikats der Konklusion sind.

Aus dieser Definition folgt, daß die Suche nach Instanzen für die Prädikatvariablen auf wenige Topologieknoten beschränkt werden kann. Eine solche Topologie kann der Benutzer vorgegeben oder durch die Systemkomponente PST ([Klingspor, 1991], [Morik *et al.*, 1993, S. 149-168]) in MOBAL berechnen lassen.

3.3.3 Sortenbeziehungen

Eine weitere semantische Einschränkung des Hypothesenraums ergibt sich aus der Nutzung von Sorteneigenschaften. In dem System MOBAL gibt es zwei Arten von Sorten:

1. Benutzerdefinierte Sorten (Benutzersorten)
2. Argumentsorten[6]

Es ist leicht ersichtlich, daß nur sortenverträgliche Hypothesen zu gewünschten Lernergebnissen führen können. An dem schon bekannten Beispiel soll dies veranschaulicht werden.

Gegeben sei folgendes Regelmodell:

```
m_example(P1,P2,C):P1(Y) & P2(X,Y) --> C(X)
```

[6] Werden durch die MOBAL–Komponente STT berechnet [Kietz, 1988].

Nach der *relation chain* wird erst `P2` instanziiert, da nur die Variable `X` durch den Zielbegriff gebunden ist. Für `P2` können nur Prädikate ausgewählt werden, die die Sorte von `X` als Sorte ihres ersten Arguments haben.

In der aktuellen Implementierung von RDT werden allerdings weder die Benutzersorten noch die Argumentsorten von STT (Sort Taxonomie Tool) genutzt. RDT berechnet und verwaltet die Sorten selbst.

3.4 Akzeptanzkriterium

Aus dem *sloppy modeling* Ansatz heraus ist es leicht verständlich, warum in RDT ein schwächeres Akzeptanzkriterium als die in Abschnitt 3.1 beschriebenen Bedingungen vom Benutzer wählbar ist. Der Benutzer kann so auch mit unvollständigen und nicht korrekten Beispielen lernen lassen, indem er z.B. Einschränkungen in der Vollständigkeit und Korrektheit des Lernergebnisses zuläßt. Es ist sogar eher unwahrscheinlich, daß die Beispielmenge und das Hintergrundwissen hundertprozentig richtig sind. Auf diese Weise erhält der Anwender eine größere Anzahl von Regeln als Lernergebnis. Aus dieser Menge muß er dann entscheiden, welche Regeln in das Sachbereichswissen übernommen werden sollen.

Das Akzeptanzkriterium setzt sich aus den folgenden Faktoren zusammen [Kietz und Wrobel, 1992]:

Sei $P(X_1, \ldots, X_m)$ eine Menge von Prädikaten, die von den Attributen $X_1, \ldots, X_m$ abhängen. Diese Prädikate sind durch Konjunktion verknüpft. Weiterhin sei H eine Hypothese der Form $H : P(X_1, \ldots, X_m) \rightarrow q(Y_1, \ldots, Y_n)$, $m \geq n$ und $\forall\, Y_i : Y_i \in (X_1, \ldots, X_m), 1 \leq i \leq n$.

Dann sind mögliche Faktoren des Akzeptanzkriteriums:

- Anzahl der positiven Beispiele, die durch die Hypothese abgedeckt sind: $\mid pos(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge q(c_1, \ldots, c_n) \in \mathcal{E}^+\} \mid$
- Anzahl der negativen Beispiele, die durch die Hypothese abgedeckt sind: $\mid neg(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge q(c_1, \ldots, c_n) \in \mathcal{E}^-\} \mid$
- Anzahl der Instanzen, für die die Schlußfolgerung nicht bekannt ist: $\mid pred(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge q(c_1, \ldots, c_n) \notin \mathcal{E}\} \mid$
- Anzahl aller möglichen Instanzen der Hypothese:
$$\begin{aligned} \mid total(H) \mid &:= \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m)\} \mid \\ &= \mid pos(H) \cup neg(H) \cup pred(H) \mid \end{aligned}$$
- Anzahl der Beispiele, für die die Schlußfolgerung gilt: $\mid concl(q) \mid := \mid \{c_1, \ldots, c_n) \mid q(c_1, \ldots, c_n)\} \mid$
- Anzahl der Beispiele, für die die Schlußfolgerung nicht von der Hypothese abgedeckt wird: $\mid uncover(H) \mid := \mid concl(H) \backslash pos(H) \mid$

Aus diesen Faktoren kann der Benutzer ein Akzeptanzkriterium aufbauen. Das voreingestellte Akzeptanzkriterium von RDT ist zum Beispiel:

$|\texttt{pos(H)}| \geq \texttt{3}$ & $(|\texttt{pos(H)}| \geq \texttt{0.5} * |\texttt{total(H)}|$ & $(|\texttt{neg(H)}| \leq \texttt{0.1} * |\texttt{total(H)}|))$.

Aus dem Akzeptanzkriterium läßt sich dann nach Satz 1 und der θ–Subsumtion ein Pruning–Kriterium ableiten. Intuitive ist klar, wie das Pruning–Kriterium abgeleitet wird. Durch das abgeleitete Pruning–Kriterium werden zu spezielle Regelmodelle bestimmt und dann nicht weiter instanziiert und getestet. Ein abgeleitetes Pruning–Kriterium für das voreingestellte Akzeptanzkriterium ist z.B. $pos(H) < 3$. Eine Hypothese, die weniger als drei positive Beispiele abdeckt, ist zu speziell. Eine weitere Spezialisierung der Hypothese könnte nie mehr als drei positive Beispiele abdecken. Im folgenden wird die formale Begründung gegeben.

Definition 13 ($\geq_\theta$)
Seien C_1 und C_2 zwei Klauseln. C_1 θ–subsumiert C_2, gdw. es eine Substitution θ gibt, so daß $C_1\theta \subseteq C_2$ gilt. Dann ist C_1 genereller als C_2 bzgl. der θ–Subsumtion, $C_1 \geq_\theta C_2$, oder C_2 spezieller als C_1 bzgl. der θ–Subsumtion, $C_2 \leq_\theta C_1$ [Muggleton und De Raedt, 1993; Plotkin, 1970; Plotkin, 1971].

Satz 1 *Seien H und H' zwei Hypothesen und $H \geq_\theta H'$, dann gilt:*

1. $pos(H) \supseteq pos(H')$
2. $neg(H) \supseteq neg(H')$
3. $total(H) \supseteq total(H')$
4. $uncovered(H) \subseteq uncovered(H')$

Beweis zu 1:
Seien $P(X_1, \ldots, X_m)$ eine Menge von Prädikaten, die von den Attributen $X_1, \ldots, X_m$ abhängen und durch Konjunktionen verknüpft sind. Weiterhin sei H eine Hypothese der Form $H : P(X_1, \ldots, X_m) \rightarrow q(X_1, \ldots, X_n)$ und $pos(H) = \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge q(c_1, \ldots, c_n)\}$

Wenn $H \geq_\theta H'$ gilt folgt aus der Definition 13:

I. $H\theta \subseteq H'$

Aus I folgt, da die Prämisse P eine Konjunktion von Prädikaten ist, daß H gleich H' ist oder aber H' wenigstens ein Literal weniger besitzt als H. Daraus folgt $pos(H) \supseteq pos(H')$.

Der Beweis zu 2,3 und 4 ist analog zu 1.

3.5 Algorithmus

Nachdem in den Abschnitten 3.2 – 3.4 die wichtigsten Ideen von RDT beschrieben wurden, soll in diesem Abschnitt die Vorgehensweise noch einmal an der Pseudo–Code Formulierung in Abbildung 3.3 bis 3.4 anschaulich skizziert werden [Morik *et al.*, 1993, S. 188 – 189].

```
rdt(Q)

    Initialisiere RS und LEAVES

    for all Regelmodelle R für die gilt, daß ihre
        Konklusion C mit Q unifizierbar ist:
        Füge RΣ, Σ = {C\Q}, in RS ein.

    endfor

    while RS≠ {}
        TOO-GENERAL = ∅
        Nimm das generellste Regelmodel R bzgl. ≥_RS aus RS
        Instanziiere-und-teste(R,TOO-GENERAL)
        for all X ∈ RS, X ≤_θ R:
            Entferne X aus RS,
            for all Y ∈ TOO-GENERAL:
                for all verschiedenen Σ : Yσ ≤_RS XΣ
                    Füge XΣ in RS ein.
                endfor.
            endfor.
        endfor.

    endwhile.
```

Abbildung 3.3: Hauptschleife von RDT

RS und LEAVES enthalten den aktuellen Suchstatus. Beim Start von RDT enthält RS alle Regelmodelle, deren Konklusion mit Q unifizierbar ist. Diese Regelmodelle werden dann nach der Ordnung $\leq_{RS}$ (siehe Abschnitt 3.3.1) mittels Breitensuche instanziiert und getestet. In LEAVES werden alle akzeptierten oder als zu speziell klassifizierten Hypothesen[7] verwaltet. Die Instanziierung der Prädikatvariablen P in den Regelmodellen erfolgt nach der Ordnung $\leq_P$ (siehe Abschnitt 3.3.1). Prädikate, die mit der so bestimmten Prädikatvariable P unifiziert werden können, müssen nun noch topologiekompatibel zur Konklusion und sortenkompatibel mit den bereits instanziierten Prädikaten der Hypothese sein.

[7] Der Begriff Hypothese wird hier auch für teilinstanziierte Regelmodelle verwendet.

```
instanziiere-und-teste(HYPO,TOO-GENERAL):-

    while RS≠ {}
        Nimm eine Hypothese H aus HYPO.
        if Prämissen in H nicht belegt
        then (
            Sei PREM die kleinste dieser Prämissen bzgl. ≤_P und
            P das Prädikat.
            for all p für die gilt:
                1. Stelligkeitskompatibel mit PREM,
                2. Topologiekompatibel zur Konklusion,
                3. Sortenkompatibel mit H.
                    Setze H auf HΣ, Σ = {P/p},
                    test(H).
            endfor. )
        else (
            if eine Konstante T zu lernen in H
            then (
                Wähle T mit dem kleinsten α(T)
                for all Werte t die T annehmen kann
                    Setze H auf Hσ, σ = {T/t},
                    test(H,TOO-GENERAL).
                endfor. )
            fi )
        fi.

    endwhile.

test(H,TOO-GENERAL):-

    if ∀H' ∈ LEAVES: H' ≰_θ H
    then (
        Teste H bzgl. des Pruning–Kriteriums
        if H zu speziell
        then Füge H in LEAVES ein.
        elseif H vollinstanziiert
            then (if H erfüllt das Akzeptanzkriterium
                    then (
                        Füge H in LEAVES ein
                        Füge H in die MOBAL-Wissensbasis ein.)
                    else Füge H in TOO-GENERAL ein.
                    fi. )
            else Füge H in HYPO ein.
            fi. )
       fi. )

    fi.
```

Abbildung 3.4: Instanziieren und testen von Hypothesen in RDT

Das Regelmodell wird dann mit den Prädikaten, die die Bedingungen erfüllen, instanziiert und getestet, d. h. es werden die Faktoren des Akzeptanzkriteriums für die Hypothese berechnet. Ist die Hypothese nach dem Pruning–Kriterium zu speziell, wird sie in die Struktur LEAVES eingefügt. Andernfalls werden die Hypothesen, die noch nicht vollinstanziiert sind, wieder in HYPO eingetragen und später weiterinstanziiert. Vollinstanziierte Hypothesen, die das Akzeptanzkriterium erfüllen, werden in LEAVES und in die MOBAL–Wissensbasis eingetragen. Erfüllen die vollinstanziierten Hypothesen das Akzeptanzkriterium nicht, so sind diese Hypothesen zu generell, da sie auch nicht durch das Pruning–Kriterium rausgefallen sind. Diese Hypothesen werden in TOO-GENERAL eingetragen. Für die zu generellen Hypothesen wird überprüft, ob in RS speziellere Regelmodelle bzgl. $\leq_\theta$ existieren. Solche spezielleren Hypothesen werden dann mit den zu generellen Hypothesen instanziiert.

3.6 RDT: Ein ILP–Verfahren

Nachdem die Ideen und die Vorgehensweise bekannt des Verfahren RDT sind, wird in diesem Abschnitt RDT nach den Kriterien aus Abschnitt 3.1.3 beurteilt.

RDT erfüllt folgende Charakteristiken:

- Empirisch: RDT erhält die komplette Beispielmenge beim Start des Verfahrens. Der Benutzer kann den Lernlauf nach dem Start nicht mehr beeinflussen.
- Singleprädikatslernen: Es kann immer nur ein Zielbegriff pro Lernlauf gelernt werden.
- Syntaktischer Bias: Durch die vorgegebenen Regelmodelle wird der Hypothesenraum für RDT beschrieben. Die Regelmodelle beschreiben die syntaktische Form des Hypothesenraums. RDT besitzt zudem einen semantischen Bias mit der Prädikatentopologie und den Sorten.

Kapitel 4

Repräsentation der Datenbank für RDT/DB

Um über eine relationale Datenbank mit dem Lernverfahren RDT lernen zu können, muß in einem ersten Schritt die tabellarische Darstellung in geeigneter Form durch Prädikate repräsentiert werden. Diese Transformation muß eine injektive Abbildung sein, damit in den Lernläufen wieder auf die Datenbank zugriffen werden kann. Die Prädikate werden in die MOBAL–Wissensbasis eingetragen.

Problemstellung:
Gegeben sei eine Menge von Relationen $\mathcal{R}$ und R $\in \mathcal{R}$ über die Attribute $a_1, ..., a_n$, die in dieser Form nicht von RDT zum Lernen berücksichtigt werden können.
Ziel:
Finde eine geeignete Prädikatendarstellung für R.

Eine solche Transformation soll automatisch aus den Informationen, die im *data dictionary* der Datenbank zur Verfügung stehen, erfolgen. Die automatische Transformation der Datenbankrelation in Prädikate für RDT kann keine optimale Repräsentation für jeden Sachbereich garantieren. Ziel ist es daher, eine für den allgemeinen Fall günstige Repräsentation der Datenbank, zum Lernen mit RDT/DB zu finden.

Im folgenden Abschnitt werden drei verschiedene Repräsentationsmöglichkeiten vorgestellt, die dann im Abschnitt 4.2 diskutiert und beurteilt werden. Im Abschnitt 4.3 wird zum Abschluß dieses Kapitels beschrieben, wie das *data dictionary* zur Umrepräsentierung der Relationen der Datenbank in eine Prädikatdarstellung genutzt wird.

4.1 Mögliche Repräsentationen

Bei vielen bisherigen Anwendungen von induktiven Lernverfahren wurden die einzelnen Tabellen (Relationen) der Datenbank durch logische *joins* zu einer *universal relation* [Ullmann, 1989, S. 1026-1069] zusammengefaßt. So konnte man schnell und einfach existierende attribut–orientierte Lernverfahren einsetzen.

Eine solche Vorgehensweise ist für ein Verfahren wie RDT nicht geeignet, da man die einzelnen Relationen der Datenbanken verliert. Es ist aber eine Zielsetzung dieser Arbeit, Beschreibungen für einzelne Relationen der Datenbank zu lernen.

Zur Umsetzung der Relationen in Prädikate sind zahlreiche Möglichkeiten denkbar. Im folgenden werden drei Möglichkeiten in Betracht gezogen und diskutiert.

Repräsentation I:

Der Tabellenname wird als Prädikatname und die Attribute der Relation als Prädikatattribute übernommen.

Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow R(a_1, ..., a_n)$.

Für die Beispieldatenbanken aus dem Abschnitt 2.4 erhält man folgende Prädikate:

Notation: Prädikatnamen werden groß und in Fettdruck geschrieben. Die Zahl nach dem Schrägstrich, z.B. **ILLEGAL/7**, gibt nach MOBAL–Konvention die Stelligkeit des Prädikats an. Das Attribute $< Spalte >$ beschreibt die Werte der Spalte, während Konstanten in der Form $Konstante$ gesetzt sind.

Beipiel KRK: Repräsentation I

- Relation illegal:

  ILLEGAL/7:
  $< white\_king\_x >$,
  $< white\_king\_y >$,
  $< white\_rook\_x >$,
  $< white\_rook\_y >$,
  $< black\_king\_x >$,
  $< black\_king\_y >$,
  $< ID >$

- Relation adjacent:

  ADJACENT/2:
  $< x >, < y >$

Beispiel B-Learn: Repräsentation I

- Relation decreasing:

  DECREASING/6:
  $< trace >$,
  $< s\_orientation >$,
  $< sensor >$,
  $< start\_time >$,
  $< end\_time >$,
  $< grad >$

- ...

- Relation s_jump:

  S_JUMP/5:
  $< trace >$,
  $< sensor >$,
  $< start\_time >$,
  $< end\_time >$,
  $< m\_orientation >$

- ...

Repräsentation II:

Eine andere Möglichkeit ist die Zerlegung der Relation in bis zu n Prädikate, indem jedes Nichtschlüsselattribut auf ein Prädikat und die Relation auf ein Prädikat mit den Primärschlüsseln als Attribute abgebildet wird.

Seien A_x der Spaltenname, $a_j, ..., a_l$ Primärschlüsselattribute der Relation R und x läuft im Intervall $[1, ..., n]$ ohne $j \dots l$.
Abbildungsvorschrift:
R über $a_1, ..., a_n \rightarrow (\forall x \in [1, \dots, n] \backslash \{j, \dots, l\} : A_x(R, a_j, \dots, a_l, a_x))$ und $R(a_j, ..., a_l)$,

---

Beipiel KRK: Repräsentation II:

- Relation illegal:

  WHITE_KING_X/3:
  illegal,
  $< ID >$,
  $< white\_king\_x >$

  WHITE_KING_Y/3:
  illegal,
  $< ID >$,
  $< white\_king\_y >$

  ...

  ILLEGAL/1: $< ID >$

- Relation adjacent:

  ADJACENT/2: $< x >, < y >$

Beispiel B-Learn: Repräsentation II:

- Relation decreasing:

  DECREASING/4:
  $< trace >$,
  $< sensor >$,
  $< start\_time >$,
  $< end\_time >$

  S_ORIENTATION/6:
  decreasing,
  $< trace >$,
  $< sensor >$,
  $< start\_time >$,
  $< end\_time >$,
  $< s\_orientation >$

  GRAD/6:
  decreasing,
  $< trace >$,
  $< sensor >$,
  $< start\_time >$,
  $< end\_time >$,
  $< grad >$

- ...

- Relation s_jump:

  S_JUMP/3:
  $< trace >$,
  $< sensor >$,
  $< start\_time >$

  END_TIME/4:
  s_jump,
  $< trace >$,
  $< sensor >$,
  $< start\_time >$,
  $< end\_time >$

  M_ORIENTATION/4:
  s_jump,
  $< trace >$,
  $< sensor >$,
  $< start\_time >$,
  $< m\_orientation >$

- ...

Repräsentation III:

Die dritte Form ist ähnlich der Repräsentation II. Auch hier erhält man maximal n Prädikate, nur mit dem Unterschied, daß die Relationsbezeichnung (Tabellenname) mit in den Prädikatnamen gezogen wird.

Seien A_x der Spaltenname, $a_j, ..., a_l$ Primärschlüsselattribute der Relation R und x läuft im Intervall $[1, ..., n]$ ohne $j \ldots l$.
Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow (\forall x \in [1, \ldots, n] \backslash \{j, \ldots, l\} : R\_A_x(a_j, \ldots, a_l, a_x))$ und $R(a_j, ..., a_l)$,

---

Beipiel KRK: Repräsentation III:

- Relation illegal:

  ILLEGAL_WHITE_KING_X/3:
  $< ID >$,
  $< white\_king\_x >$

  ILLEGAL_WHITE_KING_Y/3:
  $< ID >$,
  $< white\_king\_y >$

  ...

  ILLEGAL/1 : $< ID >$

- Relation adjacent:

  ADJACENT/2:
  $< x >, < y >$

Beispiel B-Learn: Repräsentation III:

- Relation decreasing:

  DECREASING_GRAD/5:
  $< trace >, < sensor >, < start\_time >$,
  $< end\_time >, < grad >$

  DECREASING_S_ORIENTATION/5:
  $< trace >, < sensor >, < start\_time >$,
  $< end\_time >, < s\_orientation >$

  DECREASING/4:
  $< trace >, < sensor >$,
  $< start\_time >, < end\_time >$

- ...

- Relation s_jump:

  S_JUMP_END_TIME/4:
  $< trace >, < sensor >, < start\_time >$,
  $< end\_time >$

  S_JUMP_M_ORIENTATION/4:
  $< trace >, < sensor >, < start\_time >$,
  $< m\_orientation >$

  S_JUMP/3:
  $< trace >$,
  $< sensor >$,
  $< start\_time >$

- ...

4.2 Diskussion der Repräsentationen

4.2.1 Repräsentation I

Der Vorteil der Repräsentation I liegt in der sehr einfachen Überführungsvorschrift. Dem stehen allerdings drei gravierende Nachteile gegenüber:

1. Das Lernziel wird auf die Relationen beschränkt.
2. Es stehen nur die Relationen zur Beschreibung des Zielbegriffs zur Verfügung.
3. Es gibt freie Variablen in der Konklusion.

Lernziel nur Relationen. In der Repräsentation I können einzelne Attribute der Relation nicht mehr als Lernziel für RDT ausgewählt werden. RDT könnte mit dieser Repräsentation nur über die ganzen Relation der Datenbank lernen. Es dürfte aber gerade von Interesse sein, auch über einzelne Attribute zu lernen.

Beschreibungssprache. In dem logikorientierten Lernverfahren RDT wird die Beschreibung für einen Begriff gesucht. Als Beschreibungssprache dienen dabei die dem System bekannten Prädikate. In der Repräsentation I stehen nur die einzelnen Relationen als Prädikate zur Verfügung. Die Beschreibung eines Zielbegriffs durch einzelne Attribute bestimmter Relationen ist folglich nicht möglich.

Freie Variablen. An einem Beispiel soll die Problematik deutlich gemacht werden:

Gegeben ist ein Regelmodell der Form

```
m1(P,C):P(X1,X3) & P(Y1,Y3) -->  C(X1,Y1,X2,Y2,X3,Y3,E)
```

und die Relationen aus der KRK Beispieldatenbank. Sei nun $m1$ vollinstanziiert:

$$\texttt{ADJACENT(X1,X3) \& ADJACENT(Y1,Y3)} \rightarrow$$
$$\texttt{ILLEGAL(X1,Y1,X2,Y2,X3,Y3,E)}$$

Die Variablen `X2` und `Y2` der Konklusion sind nicht in der Prämisse gebunden. Die Mächtigkeit der Wertebereiche von `X2`, `Y2` und `E` bestimmen die maximale Anzahl der möglichen Belegungen für `ILLEGAL`.

Für das Beispiel sieht das folgendermaßen aus: `X2` und `Y2` können beliebige Werte zwischen 0 und 7 annehmen. Daraus ergeben sich $8*8 = 64$ Instanziierungsmöglichkeiten. Die Relation `ADJACENT` gibt an welche Brettpositionen benachbart sind, insgesamt gibt es 22 benachbarte Brettpositionen. Daraus ergeben sich für `X1`, `Y1`, `X3` und `Y3` $22 * 22 = 484$ Belegungen im Prädikat `ILLEGAL`. Die Mächtigkeit M von `E` ist gleich der Anzahl der Datensätze in der Tabelle illegal, da `E` Primärschlüssel der Relation ist. Insgesamt gibt es somit $64 * 484 * M = 30976 * M$ Instanzen der Konklusion.

Der Benutzer müßte in einem solchen Fall die Größenordnung des Wertebereichs der freien Variablen kennen und im Akzeptanzkriterium (*total* und *pred*) berücksichtigen.

4.2.2 Repräsentation II

Mit der Repräsentation II werden die Nachteile der Repräsentation I im wesentlichen vermieden, da die Primärschlüsseleigenschaft berücksichtigt wird. Allerdings können unter bestimmten Umständen auch die Nachteile der Repräsentation I auftreten. Dies tritt aber nur ein, wenn in der Datenbank keine oder alle Attribute als Primärschlüssel ausgezeichnet wurden.

Doppelte Attribute Der Nachteil der Repräsentation II ist die Möglichkeit, daß verschiedene Relationen auf dasselbe Prädikat abgebildet werden können. In der BLearn Datenbank aus Abschnitt 2.4.2 gibt es verschiedene Relationen von *sensor features*, die aber die gleichen Attribute besitzen. Nach der Abbildungsvorschrift für die Repräsentation II erhält man zum Beispiel für das Attribut m_orientation der vier Relationen nur ein Prädikat. Das hat zur Folge, daß Informationen, die in der Datenbank schon explizit dargestellt waren, wieder verschleiert werden.

Beispiel:
Attribut m_orientation der *sensor features* in der Repräsentation II

M_ORIENTATION/4:
$< tabellenname >$,
$< trace >$,
$< sensor >$,
$< start\_time >$,
$< m\_orientation >$

In der Datenbank gibt es das Attribut m_orientation in neun Relationen.

Primärschlüsseldarstellung. Die Primärschlüssel der einzelnen Relationen werden nicht als einzelne Prädikate repräsentiert. Dies beruht auf zwei Überlegungen.

1. Die Primärschlüssel sind schon durch die Darstellung der Relation als Prädikat repräsentiert (vgl. Abbildungsvorschrift).
2. Nach der Definition 6 für Primärschlüssel in relationalen Datenbanken ist dieser eine minimale Attributkombination. Diese Attribute besitzen schon eine sehr starke semantische Verknüpfung, so daß sie nicht mehr einzeln repräsentiert werden müssen.

Ein weiterer Grund ist die Anzahl der entstehenden Prädikate. Die Menge der Prädikate ist, wie in Abschnitt 3.1 beschrieben, entscheidend für die Ausdrucksstärke der Hypothesensprache. Allerdings wirkt sich auch eine Überrepräsentation nachteilig auf die Lerndauer des Verfahrens aus, da mehr Prädikate berücksichtigt werden müssen.

4.2.3 Repräsentation III

Die Repräsentation III ist eine Erweiterung der Repräsentation II, in der der Tabellenname statt als Attribut des Prädikats in den Prädikatnamen übernommen wird. In der Repräsention II werden somit gleiche Attributnamen aus unterschiedlichen Relationen auch in unterschiedlichen Prädikaten repräsentiert.

Repräsentation	freie Variablen	max # Prädikate pro Relation	Zusammenfassung von Attributen
I	ja	1	nein
II	nein	$n+1$	ja
III	nein	$n+1$	nein

Tabelle 4.1: Vergleich der Repräsentationen I, II und III

Die Repräsentation III stellt sicher, daß im Gegensatz zur Repräsentation I keine freien Variablen in der Konklusion einer Regel auftreten können und mehrfach auftretende Attribute nicht zu einem Prädikat zusammengefaßt werden. Jeweils einen dieser Punkte erfüllen die anderen Repräsentationen nicht. In Tabelle 4.1 sind die angesprochenen Eigenschaften der Repräsentationen gegenübergestellt.

Allerdings reicht die Repräsentation III alleine nicht aus, um z.B auch Verkettungen über Attribute, die keine Schlüsselattribute sind, zu lernen. An einem Beispiel aus dem Sachbereich der Roboternavigation aus Abschnitt 2.4.2 soll die Problematik veranschaulicht werden.

Gegeben sind folgende zwei Regelmodelle:

```
m1(P1,P2,C):P1(S,T,U,V) & P2(S,T,V,X) --> C(S,T,U)
m2(P1,P2,P3,C):P1(S,T,U,V) & P2(S,T,V,X) & P3(S,T,X,Z)
```

```
--> C(S,T,U)
```

Mit diesen Regelmodellen sollen zum Beispiel aus der Datenbank B-Learn folgende Verkettungsregeln gelernt werden:

```
INCR_PEAK(S,T,U,V) & INCREASING(S,T,V,X)
→ S_JUMP(S,T,U)

DECR_PEAK(S,T,U,V) & INCREASING(S,T,V,X) & STABLE(S,T,X,Z)
→ S_JUMP(S,T,U)
```

RDT/DB würde m2 nicht als mögliche Hypothese betrachten, wenn m1 akzeptiert wurde, da m1 bzgl. der Ordnung $\leq_{RS}$ genereller ist als m2. Damit dieser Fall nicht auftritt, muß das Ende der Verkettung im Konklusionsbegriff gebunden sein. Eine Lösung ist, die Repräsentationen I und III zu kombinieren. Das Problem der freien Variablen in der Konklusion, wie in Abschnitt 4.2.1 beschrieben, ist auch in einer Kombination der Repräsentationen I und III nicht vollständig gelöst, aber auf die Primärschlüssel beschränkt.

4.2.4 Gewählte Repräsentation

Wie zu Beginn dieses Kapitels beschrieben wird, eine Abbildung der Datenbankrelationen durch Prädikate gesucht, die dem Verfahren RDT entgegenkommt und sich automatisch für jede vorgegebene Datenbank erstellen läßt. Das Ergebnis aus der Diskussion der in Betracht gezogenen Repräsentationen in Abschnitt 4.2 ist eine Kombination der Repräsentationen I und III.

Insgesamt ergibt sich folgende Abbildungsvorschrift:

Sei A_x ist der Spaltenname, $a_j, ..., a_l$ sind Primärschlüsselattribute der Relation R, und x läuft im Intervall $[1, ..., n]$ ohne $j \ldots l$.

R über $a_1, ..., a_n \rightarrow (\forall x \in [1, \ldots, n] \backslash \{j, \ldots, l\} : R\_a_x(a_j, \ldots, a_l, a_x))$, $R(a_j, ..., a_l)$ und $Rel\_R(a_1, \ldots, a_n)$.

Definition 14 (Datenbankprädikate)
Prädikate, die das Ergebnis der Abbildungsvorschrift sind heißen Datenbankprädikate.

Präfix	Beschreibung
USER	Beschränkt den View auf den Benutzer, z.B. nur seine selbst erstellten Objekte
ALL	Views mit dem Präfix ALL beziehen sich auf die Gesamtsicht des Benutzers auf die Datenbank. Diese Views liefern Informationen über Objekte, auf die er Zugriffsrechte hat.
DBA	Dieses Präfix erlaubt die Sicht auf die ganze Datenbank.

Tabelle 4.2: Präfixe von System–Views

4.3 Nutzung des *data dictionary*

Nach der Frage, wie die Relationen der Datenbank repräsentiert werden sollen, stellt sich die Frage, welche Relationen sollen zum Lernen genutzt werden.

In Abschnitt 2.1 wurde schon beschrieben, welche Art von Informationen in dem *data dictionary* abgelegt sind.

Welche Informationen werden benötigt?

- Die Relationen, die dem Benutzer zugänglich sind und
- die Primärschlüssel dieser Relationen .

Die benötigten Informationen sind in ORACLE 7 in folgenden Gruppen von System–Views enthalten:

- tab_columns
- ind_colomns
- constraints

Eine solche Gruppe von System–Views besteht aus drei Views mit ähnlichen Informationen, die sich jedoch durch ihre Präfixe unterscheiden. Die Bedeutung der einzelnen Präfixe ist in der Tabelle 4.2 dargestellt.

Zum Lernen sollen alle Informationen die dem Benutzer in der Datenbank zugänglich sind genutzt werden. Deshalb werden die Informationen mit dem Präfix ALL aus der Datenbank gelesen. Mit diesem Präfix hat der Benutzer auch Zugang zu Systemtabellen, die aber keine Informationen über den Sachbereich, der in der Datenbank dargestellt ist, enthalten und deshalb nicht als Prädikate in die MOBAL–Wissensbasis eingetragen werden.

Kapitel 5

RDT/DB

In diesem Kapitel werden die speziellen Änderungen für RDT/DB beschrieben, die durchgeführt wurden, um RDT auf relationale Datenbanken anzuwenden. Als erstes wird in einem Exkurs die SQL–Anweisung SELECT COUNT eingeführt, die für die Umsetzung des Akzeptanzkriteriums in SQL–Anweisungen in Abschnitt 5.1 von zentraler Bedeutung ist. Im darauffolgenden Abschnitt 5.2 werden Möglichkeiten, die MOBAL–Wissensbasis in RDT/DB zu nutzen, diskutiert. Weiterhin werden in Abschnitt 5.3 die Hypothesenraumeinschränkungen von RDT/DB beschrieben. Im letzten Abschnitt dieses Kapitels wird dann noch eine spezielle Funktion in RDT/DB erläutert.

Exkurs: SELECT COUNT**–Anweisung in SQL**[1]

Die Auswahl von Spalten und Zeilen erfolgt in SQL durch den SELECT–Befehl. Grundlage des Exkurs ist SQL–ORACLE. Der Befehl SELECT besteht aus sechs Komponenten, um die Spalten und Zeilen, die ausgewählt werden sollen, zu spezifizieren.

1. SELECT : Angabe der Spalten.
2. FROM : Angabe der Tabellen.
3. WHERE : Bedingen, die die Ergebnistupel erfüllen müssen.
4. GROUP BY: Gruppenbildung bei gleichen Werten in der angegebenen Spalte.
5. HAVING : Selektion spezifischer Gruppen.
6. ORDER BY : Sortierfolge der Tupel der Ergebnistabelle.

Bis auf die ersten zwei Komponenten sind alle optional. In diesem Exkurs wird im weiteren nur auf die ersten drei Komponenten des SELECT–Befehls

[1]Dieser Abschnitt ist für Leser gedacht, die nicht mit SQL vertraut sind, damit sie die folgende Anweisungen verstehen können.

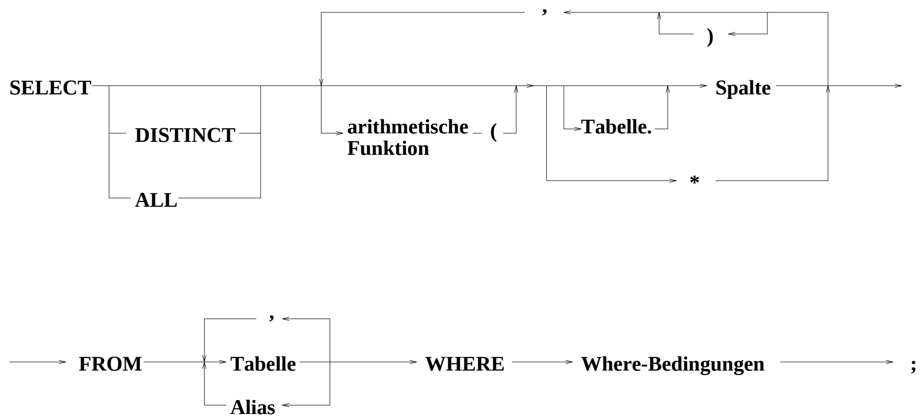

Abbildung 5.1: Vereinfachte Syntax der SELECT–Anweisung

eingegangen, da nur diese zum weiteren Verständnis der Arbeit wichtig sind. Die Syntax dieser Komponenten ist in Abbildung 5.1 dargestellt. In der SELECT–Komponente können zusätzlich arithmetische Gruppenfunktionen auf ausgewählten Spalten angewendet werden. Hierzu gehört auch die Funktion COUNT. COUNT bestimmt die Anzahl der ausgewählten Elemente und COUNT(*) die Anzahl der Elemente, die die WHERE–Bedingung erfüllen.
Ende des Exkurses.

5.1 Umsetzung des Akzeptanzkriteriums in SQL–Anfragen

Um das Verfahren RDT auf relationale Datenbanken anwenden zu können, müssen geeignete SQL–Anfragen bestimmt werden, die die einzelnen Faktoren des Akzeptanzkriteriums berechnen.

Seien $P(X_1, \ldots, X_m)$ eine Menge von Prädikaten, die von den Attributen $X_1, \ldots, X_m$ abhängen. Diese Prädikate sind durch Konjunktionen verknüpft. Weiterhin sei H eine Hypothese der Form $H : P(X_1, \ldots, X_m) \rightarrow q(Y_1, \ldots, Y_n),\ \forall Y_i\ :\ Y_i\ \in\ (X_1, \ldots, X_m), 1\ \leq\ i\ \leq\ n$. Die Prädikate der Hypothese H sind von der Form $R\_a_x(PS, a_x)$, bzw. $R(PS)$ oder $Rel\_R(a_1, \ldots, a_n)$, wobei R die Relationenbezeichnung (Tabellenname) in der relationalen Datenbank, a_x ein Attribut der Relation R und PS der Primärschlüssel der Relation R ist. Weiterhin sei $ps(Q)$ eine Funktion, die den Primärschlüssel des Prädikats Q bestimmt, $tabelle(Q)$ eine Funktion, die den zugehörigen Tabellennamen zum Prädikat Q bestimmt und $v(P)$ eine Funktion, die die Prämissen P in SQL–Bedingungen überführt.

Die Bestandteile des Akzeptanzkriteriums können durch folgende SQL–Anfragen berechnet werden:

- $\mid pos(H) \mid \Rightarrow$
  SELECT COUNT(*) FROM $tabelle(q), tabelle(P)$
  WHERE $v(P(c_1, \ldots, c_m))$;

- Negative Beispiele stellen in relationalen Datenbanken ein besonderes Problem dar, da in relationalen Datenbanken keine negierten Relationen existieren. Durch die Einbindung von RDT/DB in MOBAL hat der Benutzer die Möglichkeit, negative Fakten des Zielbegriffs in die MOBAL–Wissensbasis einzugeben und diese beim Lernen zu berücksichtigen (siehe dazu auch Abschnitt 5.2.4).

  $\mid neg(H) \mid \Rightarrow$
  SELECT COUNT(*) FROM $tabelle(P), tabelle(q)$
  WHERE $v(P(c_1, \ldots, c_m) \ \wedge not(q))$,
  $not(q)$ ist die Menge der negativen Instanziierungen des Zielbegriffs q aus der MOBAL–Wissensbasis.

- $\mid pred(H) \mid = \mid total(H) \mid - \mid pos(H) \mid - \mid neg(H) \mid$

- $\mid total(H) \mid \Rightarrow$
  SELECT COUNT(*) FROM $tabelle(P)$
  WHERE $(v(P(c_1, \ldots, c_m))$

- $\mid concl(H) \mid \Rightarrow$
  SELECT COUNT($ps(q)$) FROM $tabelle(q)$;

- $\mid uncover(H) \mid = \mid concl(H) \mid - \mid pos(H) \mid$

Von diesen Faktoren müssen nur *pos*, *neg* und *total* für jede partielle Hypothese mittels Datenanfragen berechnet werden. Während *concl* nur einmal zu Beginn des Lernlaufes berechnet werden muß, da der Faktor nicht von $P(X_1, \ldots, X_m)$ abhängt. Der Faktor *total* wird in RDT/DB etwas anders behandelt als in RDT. Dieses ergibt sich aus der Anwendung von RDT auf relationale Datenbanken. So werden unter *total* alle bekannten Instanzen in der Datenbank gezählt und *pred* aus der Beziehung $\mid total(H) \mid = \mid pos(H) \cup neg(H) \cup pred(H) \mid$ berechnet, so daß $\mid pred(H) \mid = \mid total(H) \backslash pos(H) \backslash neg(H) \mid$ ist. Somit drückt *pred* das Vorhersagemaß bzgl. der Datenbankinformationen aus. In der aktuellen Implementierung von RDT erfolgt die Berechnung von *total* und *pred* genau umgekehrt, d.h. es wird *pred* berechnet, und *total* aus den anderen Faktoren berechnet. Allerdings erfolgt die Berechnung von *pred* durch die Multiplikation von *pos* mit der Anzahl der Elemente aus der größten bekannten Sorte in MOBAL. Aus diesem Grund wurde die oben beschriebene Berechnung der Faktoren des Akzeptanzkriteriums gewählt.

Ein weiteres Argument für die Berechnung von *pred* und *total* in der Form für RDT/DB, kann aus der Repräsentation der Datenbank in Prädikate und der Primärschlüsseleigenschaft angeführt werden. Da die Primärschlüsselattribute des Zielbegriffs mit sehr hoher Wahrscheinlichkeit schnell in der Prämisse gebunden sind, kann die Konklusion keine weiteren Belegungen besitzen. Aus diesen Gründen wurde die oben beschriebene Berechnung der Faktoren des Akzeptanzkriteriums gewählt.

5.2 Nutzung der MOBAL–Wissensbasis beim Lernen

In diesem Abschnitt wird die Verwendung spezieller Wissenskomponenten aus der MOBAL–Wissensbasis in RDT/DB diskutiert. In den folgenden Abschnitt 5.2.1 und 5.2.2 wird die Ausnutzung der Regelmodelle und der Prädikatentopologie aus der MOBAL–Wissensbasis für RDT/DB beschrieben. In Abschnitt 5.2.3 wird dann die Verwendung der MOBAL–Wissensbasis als Hintergrundwissen für RDT/DB diskutiert. In den letzten zwei Abschnitt wird dann noch die Möglichkeit beschrieben, negative Beispiele aus der MOBAL–Faktenbasis und die *built–in*–Prädikate von MOBAL zum Lernen über relationale Datenbanken zu nutzen.

5.2.1 Regelmodelle

Die Regelmodelle werden in RDT/DB zur Hypothesenraumeinschränkung in derselben Weise genutzt wie in RDT (siehe Abschnitt 3.3.1). Da RDT/DB in das Wissensakquisitionssystem MOBAL integriert ist, kann auf die in MOBAL vorhandenen Regelmodelle zugegriffen werden.

5.2.2 Prädikatentopolgie

Auch die Prädikatentopologie kann in derselben Weise genutzt werden, wie in RDT (siehe Abschnitt 3.3.2). Da die Datenbankprädikate, wie in Kapitel 4 beschrieben, als Prädikate in die MOBAL-Wissensbasis eingefügt werden, kann der Benutzer auch für sie eine Topologie erstellen oder durch das System berechnen lassen.

5.2.3 MOBAL–Prädikate als Hintergrundwissen

In diesem Abschnitt soll der Frage nachgegangen werden, inwieweit die Faktenbasis und die Prädikate von MOBAL beim Lernen mit RDT/DB über relationale Datenbanken als Hintergrundwissen berücksichtigt, bzw. genutzt werden kann. Hierzu wird folgende Annahme gemacht: Die Prädikatsnamen aus der MOBAL–Wissensbasis und die der Datenbankprädikate sind verschieden.

Das Problem bei einer Kombination der beiden Wissensbasen (MOBAL–Wissensbasis und Datenbank) ist die Auswertung des Akzeptanzkriteriums.

Um mit MOBAL-Prädikaten als Hintergrundwissen und RDT/DB lernen zu können, muß nach folgendem Schema vorgegangen werden:

1. Instanziierung der Prädikatsvariablen des Regelmodells nach der Ordnung $\leq_P$.

2. Belegung der Prädikatsvariablen mit einem geeigneten Datenbankprädikat oder MOBAL–Prädikat.

3. Fallunterscheidung: Sind nur Datenbankprädikate instanziiert ?

   Ja: Die Auswertung der Faktoren des Akzeptanzkriterium kann durch die Datenbank erfolgen (siehe Abschnitt 5.1). Es wird dazu maximal eine Anfrage pro zu berechnendem Faktor benötigt.

   Nein: Es wurde mindestens ein MOBAL–Prädikat in das Regelmodell instanziiert. In diesem Fall müssen die Faktoren des Akzeptanzkriteriums durch eine Summe von Datenbankanfragen berechnet werden. Für alle Belegungen der MOBAL–Prädikate werden die einzelnen Faktoren des Akzeptanzkriteriums berechnet. Die Summe über die einzelnen Faktoren sind die Faktoren für die Hypothese.

   Seien n die möglichen Belegungen der MOBAL–Prädikate, so werden für jeden Faktor des Akzeptanzkriteriums n Datenbankanfragen gestellt. Für $pos(H)$ ergibt sich z.B. folgende Berechnungsformel: $\sum_{i=1}^{n} pos(H, Wert_i)$, wobei $Wert_i$ eine der möglichen Belegungen der Prädikate aus der MOBAL–Wissensbasis ist.

An einem Beispiel soll der Sachverhalt veranschaulicht werden.

Beispiel: Gegeben ist der Sachbereich KRK aus Abschnitt 2.4.1. Hier sei allerdings die Relation adjacent als Hintergrundwissen in MOBAL gegeben, so daß folgendes Szenario gegeben ist:

Datenbankprädikat:

ILLEGAL_WHITE_KING_X/2: $< ID >, < X >$
ILLEGAL_WHITE_KING_Y/2: $< ID >, < Y >$
ILLEGAL_BLACK_KING_X/2: $< ID >, < X >$
ILLEGAL_BLACK_KING_Y/2: $< ID >, < Y >$

MOBAL–Prädikat als Hintergrundwissen:

adjacent/2: $< X >, < Y >$

Regelmodell: Gegen ist ein Regelmodell

```
m1(P1,P2,P3,P4,P5,C):P1(ID,X1) & P2(ID,Y1) & P3(ID,X2) &
P4(ID,Y2) & P5(X1,X2) & P5(Y1,Y2) --> C(ID),
```

das teilinstanziiert ist:

```
ILLEGAL_WHITE_KING_X(ID,X1)&ILLEGAL_WHITE_KING_Y(ID,Y1)&
ILLEGAL_BLACK_KING_X(ID,X2)&ILLEGAL_BLACK_KING_Y(ID,Y2)&
P5(X1,X2) & P5(Y1,Y2) → ILLEGAL(ID)
```

Bis zu dieser Belegung konnte die Berechnung der Faktoren des Akzeptanzkriteriums durch einfache Datenbankanfragen, wie in Abschnitt 5.1 beschrieben, erfolgen. Wird nun für P5 das Prädikat adjacent aus der MOBAL–Wissensbasis instanziiert, kann die Berechnung der Faktoren nicht mehr mit den Anfragen aus Abschnitt 5.1 durchgeführt werden. Für jede mögliche Belegung von adjacent müssen die Faktoren des Akzeptanzkriteriums berechnet und die Summe gebildet werden. Für das Prädikat adjacent gibt es 22 Belegungen. Da adjacent zweimal in dem Regelmodell instanziiert ist, gibt es $22 * 22 = 484$ Belegungen. Das bedeutet, daß 484 Datenbankanfragen pro zu berechnendem Akzeptanzkriteriumfaktor durchgeführt werden müssen.

Aufgrund der Vielzahl zu stellender Datenbankanfragen wurde auf die Nutzung der Prädikate aus der MOBAL–Wissensbasis als Hintergrundwissen für RDT/DB verzichtet.

5.2.4 Faktenbasis für negative Beispiele

In relationalen Datenbanken existieren keine negativen Beispiele. Die Idee bei Datenbanken ist, den aktuell gültigen Sachverhalt festzuhalten. Es werden somit nur positive Beispiel in einer Datenbank abgelegt.

In vielen Fällen beim maschinellem Lernen ein negatives Beispiel nützlicher ist als weitere positive Beispiele. Negative Beispiele werden benötigt, um zu generelle Hypothesen auszuschließen. Es stellt sich die Frage, wie man beim Lernen über relationale Datenbanken negative Beispiele berücksichtigen kann.

Hier würde sich anbieten, eine kleine Anzahl zu den Datenbankprädikaten von dem Benutzer in die MOBAL–Wissensbasis eintragen zu lassen und beim Lernen mit RDT/DB zu berücksichtigen. Durch eine entsprechend starke Gewichtung im Akzeptanzkriterium könnte eine kleine Anzahl von negativen Beispielen ausreichend sein.

Die Anzahl der negativen Beispiele wird nicht durch das System beschränkt. Es ist dem Benutzer, überlassen die nötige Anzahl an negativen Beispielen zu bestimmen. Dem Benutzer muß bei der Auswahl bzw. der Eingabe der negativen Beispiele nur klar sein, daß die Anzahl der negativen Beispiele stark den Aufwand der Berechnungen beeinflußt. Eine große Anzahl von negativen Beispielen erhöht die Anzahl der Bedingungen in der Datenbankanfrage, und dies bedeutet einen höheren Rechenaufwand in der Datenbank. Die Problematik ist mit der im Abschnitt 5.2 vergleichbar.

5.2.5 MOBAL–*built-in*–Prädikate in RDT/DB

Zur Instanziierung der Regelmodelle können prinzipiell alle bekannten Prädikate verwendet werden. Zusätzlich gibt es in MOBAL eine Anzahl von vordefinierten Prädikaten, sogenannte *built-in*–Prädikaten. Diese Prädikate können in RDT in derselben Weise genutzt werden, wie alle anderen Prädikate. Die zur Verfügung stehenden Prädikate sind:

eq/2: $< X >, < Y >$: X gleich Y.

ne/2: $< X >, < Y >$: X ungleich Y.

lt/2: $< X >, < Y >$: X kleiner als Y.

gt/2: $< X >, < Y >$: X größer als Y.

le/2: $< X >, < Y >$: X kleiner gleich Y.

gt/2: $< X >, < Y >$: X größer gleich Y.

add/3: $< X >, < Y >, < Z >$: Die Summe von X und Y ist gleich Z.

sub/3: $< X >, < Y >, < Z >$: Die Differenz von X und Y ist gleich Z.

prod/3: $< X >, < Y >, < Z >$: Das Produkt von X und Y ist gleich Z.

div/3: $< X >, < Y >, < Z >$: Der Quotient von X und Y ist gleich Z.

Die Umsetzung dieser Prädikate ist auch in SQL möglich, so daß die *built-in*–Prädikate aus MOBAL auch in RDT/DB benutzt werden können.

5.3 Einschränkungen des Hypothesenraums in RDT/DB

Die Hypothesenraumeinschränkung von RDT/DB ist in einigen Punkten anders als bei RDT. Während die Hypothesenraumeinschränkung durch Regelmodelle und die Prädikatentopologie beibehalten wurde, wurde auf die Sortenkompatibilität, wie aus dem Abschnitt 5.3.1 hervorgeht, verzichtet. In den Abschnitten 5.3.2 und 5.3.3 werden zwei Hypothesenraumeinschränkungen vorgestellt, die sich aus der Anwendung von RDT auf relationale Datenbanken ergeben.

5.3.1 Sortenbeziehungen

In RDT/DB wurde auf die Ausnutzung der Sortenbeziehungen zur Hypothesenraumeinschränkung verzichtet, da die Verwaltung der Sorten einer Datenbank zu umfangreich wäre und die Mengenvergleichsoperation wegen der hohen Anzahl von Elementen in den Sorten sehr laufzeitintensiv sind. Eine Mengenvergleichsoperation läßt sich in $O(n^2)$ durchführen[2]. Im Hinblick darauf, daß eine ähnliche Hypothesenraumeinschränkung durch Überprüfung der Datentypen erreicht werden kann, fiel die Entscheidung, auf die Sortenbeziehungen zu verzichten, nicht schwer. Die Einschränkung des Hypothesenraums durch die Datentypen ist eine schwache Sorteneinschränkung. Wenn in zukünftigen Datenbanksystemen benutzerdefinierte Integritätsbedingungen voll Unterstützt werden, sollte diese Art der Hypothesenraumeinschränkung auch wieder in RDT/DB verwendet werden.

5.3.2 Datentypkompatibilität

Eine weitere Möglichkeit der Hypothesenraumeinschränkung neben den Regelmodellen und der Prädikatentoplogie ergibt sich beim Lernen über relationale Datenbanken aus der Kenntnis des Datentyps jedes einzelnen Attributes eines Prädikats.

Es können also nur Prädikate, die datentypkompatibel zu den bisher instanziierten Prädikaten sind, zu einem Ergebnis führen.

Definition 15 (Datentypkompatibel) *Sei $m(P_1, \ldots, P_n, C)$ ein teilinstanziiertes Regelmodell, $P(A_1, \ldots, A_k)$ ein zu instanziierendes Prädikat in m für $P_i(V_1, \ldots, V_k)$ und $L = [(V_j, D_l), \ldots]$ eine Liste von Paaren bestehend aus bereits gebundener Variablen und deren Datentyp. Weiterhin sei $dt(X)$ eine Funktion, die den Datentyp eines Attributs bestimmt.*
P ist datentypkompatibel zu m, wenn gilt:

$$\forall r : P_i(\ldots, V_r, \ldots) \ \wedge \ (V_r, D_s) \in L \Rightarrow D_s = dt(A_r)$$

Die in dem Datenbanksystem ORACLE 7 vorhandenen Datentypen sind im Abschnitt 2.2, Tabelle 2.1 beschrieben.

An einem abstrakten Beispiel soll der Sachverhalt deutlich gemacht werden.

Gegeben ist folgendes Regelmodell:

```
m_example(P1,P2,C):P1(Y) & P2(X,Y) --> C(X)
```

Nach der *relation chain* wird erst P2 instanziiert, da nur die Variable `X` durch den Zielbegriff gebunden ist. Für P2 können, nach der Bedingung der Datentypkompatibilität, nur Prädikate instanziiert werden, deren erstes Argument den gleichen Datentyp haben wie das Argument des Zielbegriffs.

Die Bedingung der Datentypgleichheit kann in einem bestimmten Fall abgeschwächt werden. Bei den Datentypen `CHAR` und `VARCHAR2` ist es ausreichend, daß sie typenverträglich sind. Der Datentyp `CHAR` ist eine Zeichenkette mit fester Länge; wird eine kürzere Zeichenkette eingegeben so füllt

[2] Hierbei wird von ungeordnete Mengen ausgegangen.

ORACLE 7 die restlichen Stellen mit Leerzeichen auf. Die beiden Datentypen werden durch die Schnittstelle von PROLOG zu ORACLE 7 vergleichbar (vgl. Abschnitt 6.1). Die Schnittstelle ist so realisiert, daß Leerzeichen, die vor und nach der Zeichenkette stehen, nicht berücksichtigt werden.

5.3.3 Redundante Prädikate

Eine weitere datenbankspezifische Einschränkung des Hypothesenraums ergibt sich aus der Erkennung redundanter Prädikate durch Ausnutzung der Primärschlüsseleigenschaft. An einem Beispiel aus dem KRK–Sachbereich (siehe Abschnitt 2.4.1) soll der Sachverhalt dargestellt.

Gegeben ist folgendes Regelmodell:

```
m_krk1(P1,P2,P3,C):P1(E,X1) & P2(E,Y1) & P3(E,X1)  --> C(E)
```

Angenommen, P1 ist mit dem Prädikat ILLEGAL_WHITE_ROOK_X belegt. Da es sich bei den zu lernenden Regeln immer um konjunktive Verknüpfungen handelt, sind Prädikate, die mehr als einmal in der Prämisse einer Regel auftreten und deren Attribute mit den gleichen Variablen belegt sind, redundant in der Prämisse. Für das Beispiel heißt das, wenn ILLEGAL_WHITE_ROOK_X auch für P3 instanziiert würde, wäre P3 redundant belegt.

Beim Lernen in relationalen Datenbanken mit der in Kapitel 4 ausgewählten Repräsentation können weitere Instanziierungen in der Prämisse ausgeschlossen werden, weil sie redundant sind. Hier kann die Eigenschaften der Primärschlüssel zum Lernen ausgenutzt werden.

In der Prädikatendarstellung der Datenbank gibt es drei Arten von Prädikaten:

- Prädikate, die die Relation beschreiben und wo die Attribute des Prädikats der Primärschlüssel der Relation ist,
- Prädikate, die die gesamte Relation beschreiben und
- Prädikate, die einzelne Spalten der Tabelle repräsentieren.

Prädikate, bei denen der Prädikatsname gleich dem Tabellennamen ist, müssen nach dem oben beschriebenen Kriterium auf Redundanz getestet werden, während es bei den anderen Prädikaten ausreicht, die Variablen der Primärschlüsselattribute zu vergleichen. Dies bedeutet, daß im ersten Fall alle Attribute des Prädikats getestet werden müssen, während bei den anderen Prädikaten nur ein Teil der Attribute getestet werden.

An dem Beispiel soll dies nochmal erläutert werden. P2 soll mit ILLEGAL_WHITE_ROOK_X belegt werden. Dies wäre eine redundante Instanziierung, da die Variablen der Primärschlüsselattribute identisch sind. Aus der Primärschlüsseleigenschaft folgt, daß dann auch `X1 = Y1` gilt.

Formal läßt sich die Redundanz der Prädikate in einen Regelmodell folgendermaßen definieren:

Definition 16 (Prädikatredundanz)
Sei $m(P_1, \ldots, P_n, C)$ ein teilinstanziiertes Regelmodell, $P(A_1, \ldots, A_k)$ ein zu instanziierendes Prädikat in m und $ps(Q)$ eine Funktion, die die Primärattribute des Prädikats Q bestimmt. $P(A_1, \ldots, A_k)$ ist redundant in m, gdw. $\exists P_i \in m(P_1, \ldots, P_n, C) : ps(P_i) = ps(P)$.

Der Vollständigkeithalber muß an dieser Stelle gesagt werden, das es auch andere Arten von Prädikatredundanzen in Regeln gibt, die aber im allgemeinen nur mit sehr hohem Aufwand berechnet werden können, z.B. durch θ–Subsumtionstests.

5.4 Spezielle Funktion in RDT/DB

In den ersten Lernläufen[3] mit RDT/DB wurde deutlich, daß der θ–Subsumtionstest, der zwischen der aktuell betrachteten Hypothese und allen bisher zu speziellen sowie allen akzeptierten Hypothesen durchgeführt wird, sehr laufzeitintensiv ist.

Das Problem des θ–Subsumtionstests ist NP–hart, so daß vermutet wird, das es keine effiziente Algorithmen für dieses Problem gibt [Kietz und Lübbe, 1994].

Aus dieser Problematik heraus entstand die Idee, die aktuell betrachtete Hypothese nur gegen die bisher akzeptierten Hypothesen zu testen. Dies hat die Auswirkung, daß Hypothesen, die spezieller sind als bereits bekannte zu spezielle Hypothesen, nicht durch den θ–Subsumtionstest eliminiert werden, sondern erst durch den Test gegen die Datenbank.

An einem Ausschnitt aus einem allgemeinen Instanziierungsgraph (siehe Abbildung 5.2), in dem die Knoten Instanzen eines Regelmodells sind und die Kanten deren Ordnung, bzgl. $\leq_\theta$ ausdrücken, soll die Idee anschaulich dargestellt werden.

Annahme: Der Knoten B ist eine zu spezielle Instanziierung des Regelmodells.

Wenn nur die akzeptierten Hypothesen in LEAVES eingetragen werden, wird B jetzt nicht in LEAVES eingetragen. Wird nun im Rahmen der weiteren Instanziierungen E über C erreicht, wird E nicht durch einen θ–Subsumtionstest als zu speziell erkannt, sondern erst bei der Auswertung des Akzeptanz-, bzw. des Pruningkriteriums. Dasselbe gilt, wenn man über D wieder zum Knoten E kommt.

Ein solches Vorgehen verringert den Zeitaufwand, der für den θ–Subsumtionstest benötigt wird, da die aktuell betrachtete Hypothese mit

[3] Siehe Testergebnisse in Kapitel 7.

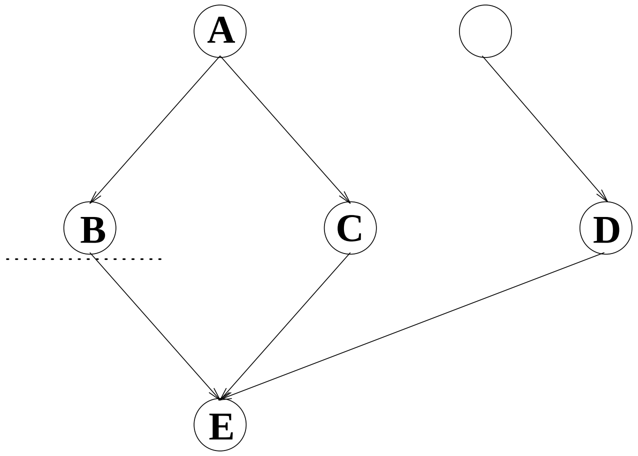

Abbildung 5.2: Instanziierungsgraph

viel weniger Hypothesen verglichen werden muß. Die zusätzlichen Berechnungen der Faktoren des Akzeptanzkriteriums sind vernachlässigbar.

Als Schlußfolgerung aus dieser Beobachtung wurde in RDT/DB ein zusätzlicher Parameter eingefügt, über den der Benutzer bestimmen kann, ob die zu speziellen Hypothesen in LEAVES eingetragen werden und damit, ob diese im θ–Subsumtionstest berücksichtigt werden oder nicht. Die experimentellen Tests in Kapitel 7 ohne den θ–Subsumtionstest für zu spezielle Hypothesen, zeigen ein verbessertes Laufzeitverhalten bis zu einem Faktor 20.

Kapitel 6

Programmarchitektur

6.1 Kopplung RDT/DB mit ORACLE

In diesem Abschnitt werden die verschiedenen Kopplungsarten und die in dieser Arbeit in Betracht gezogenen technischen Kopplungsmöglichkeiten zur Verbindung von Prologprozessen und dem Datenbanksystem ORACLE 7 beschrieben. Weiterhin werden die Architektur von RDT/DB aufgezeigt und die wichtigsten Parameter von RDT/DB beschrieben. Zum Abschluß dieses Kapitels werden noch einige Hinweise zur Benutzung von RDT/DB in MOBAL gegeben.

6.1.1 Kopplungsarten

Bei der Kopplung von wissensbasierten Systemen gibt es zwei Arten von Kopplungen [Ceri *et al.*, 1990, S. 41 - 64]:

1. die dichte Kopplung und
2. die lose Kopplung[1].

Lose Kopplung zwischen Prolog und Datenbanken

Bei einer losen Kopplung werden die von dem Prologprozeß benötigten Daten beim Start des Prozesses in die Prolog–Wissensbasis geladen. Während des Programmablaufs werden keine Anfragen an die Datendank gestellt. Dies bedeutet, daß zu Beginn des Prologprogramms alle benötigten Daten in den Hauptspeicher transferiert werden müssen. In der Abbildung 6.1 ist das Schema einer losen Kopplung dargestellt. Ein anderer Begriff für lose Kopplung ist statische Kopplung. Der Vorteil einer solchen Kopplung ist, daß während des Programmablaufs keine Verbindung zur Datenbank mehr aufgebaut werden muß. Der schwerwiegende Nachteil dieser Lösung

[1] Die Begriffe lose und dichte Kopplung werden in der Literatur nicht einheitlich benutzt. In dieser Arbeit werden die Begriffe nach S.Ceri u.a. [Ceri *et al.*, 1990] benutzt.

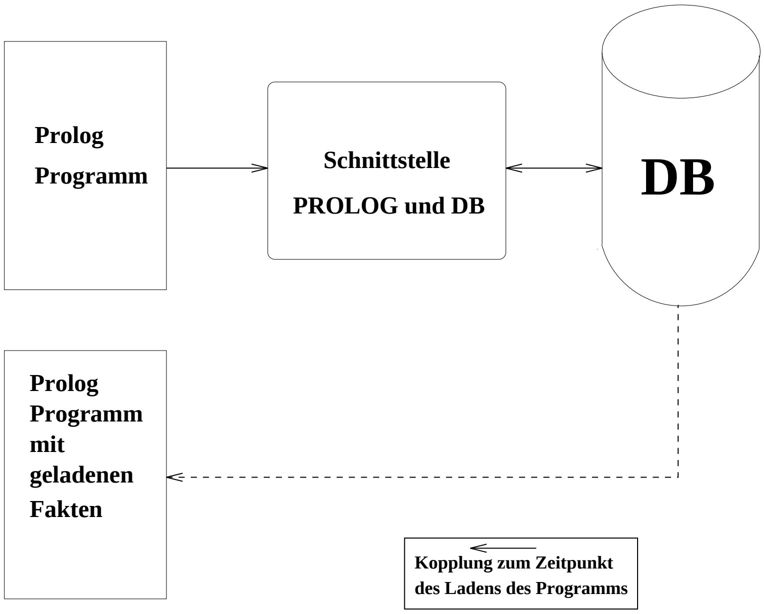

Abbildung 6.1: Schema einer losen Kopplung

ist allerdings die Tatsache, daß alle möglicherweise benötigten Daten aus der Datenbank in der Prolog–Wissensbasis, und damit im Hauptspeicher gehalten werden müssen.

Dichte Kopplung zwischen Prolog und Datenbanken

Im Gegensatz zu einer losen Kopplung werden bei einer dichten Kopplung die von dem Prologprogramm benötigten Daten während des Programmablaufs über die Schnittstelle von der Datenbank abgefragt. Eine solche Kopplung wird auch dynamische Kopplung genannt. Es werden also nur selektiv die wirklich benötigten Daten von der Datenbank abgefragt. Die Abbildung 6.2 zeigt ein allgemeines Schema einer dichten Kopplung. Der Vorteil dieser Lösung ist, daß auf den jeweils aktuellen Daten gearbeitet wird und die Speicherressourcen nicht unnötig beansprucht werden. Dem gegenüber stehen als Nachteil die längeren Zugriffszeiten, da jedesmal über die Schnittstelle zur Datenbank gegangen werden muß.

6.1.2 Technische Kopplungsmöglichkeiten

Für die Kopplung von RDT/DB, das in PROLOG implementiert ist, und der Datenbank ORACLE 7 standen zwei Möglichkeiten zur Diskussion.

1. UNIX Pipes
2. TCP/IP

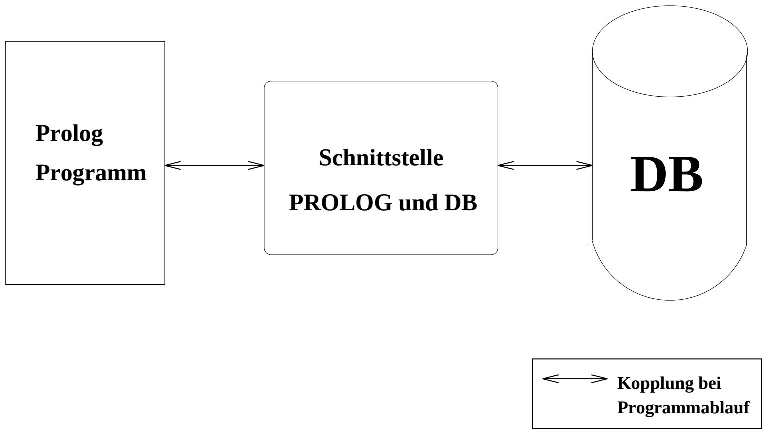

Abbildung 6.2: Schema einer dichten Kopplung

Unix–PIPES

Unter UNIX gibt es ein Konstrukt, das die Kommunikation zwischen verschiedenen Prozessen unterstützt, sogenannte PIPES. PIPES lenken die Ausgabe eines Programms zur Eingabe eines anderen Programms. Der hierzu verwendete Puffer kann als eine effiziente Implementierung einer temporären Datei betrachtet werden. Für eine bi–direktionale Kommunikation müssen zwei PIPES angelegt werden, da für die PIPES die Sender und Empfänger fest vorgegeben werden [Gulbins, 1988, S.58-59,108]. Das PIPE–Konstrukt unterstützt aber keine Kommunikation in Rechnernetzen. Dies bedeutet, daß das Datenbanksystem und MOBAL, da RDT/DB ein TOOL von MOBAL ist, auf demselben Rechner laufen müssen.

TCP/IP

TCP/IP ist eine standardisierte Kommunikationsvorschrift zwischen zwei Prozessen in lokalen Netzen. Ursprünglich wurde TCP/IP für UNIX an der Universität von Kalifornien in Berkeley entwickelt. Heute ist es ein Standard für viele verschiedene Computersysteme. Die Palette reicht vom PC bis zum Großrechner. Als ein solcher Standart wird TCP/IP auch von QUINTUS PROLOG und ORACLE 7 unterstützt. Einige Vorteile von TCP/IP sind:

- TCP/IP ermöglicht eine schnelle Datenübertragung.
- Die Spezifikation ist vollständig und verfügbar.
- TCP/IP ist aufgrund seiner Verbreitung auf vielen Computersystemen portabel und auch in heterogenen Netzen einsetzbar.

6.1.3 Realisierte Kopplung

Für die Anwendung von RDT auf relationale Datenbanken wurde eine dichte Kopplung gewählt. Die Entscheidung für eine solche dynamische Kopplung liegt schon in der Zielsetzung begründet. Ziel dieser Arbeit war es, die Datenbank als Faktenbasis zugänglich zu machen. Dies bedeutete auch die spezifischen Eigenschaften der Datenbank auszunutzen und nicht einfach eine Datenbank in die MOBAL–Wissensbasis zu kopieren. Eine komplette Kopie wäre notwendig, da nicht vorhergesehen werden kann, welche Daten für die einzelnen Lernläufe relevant sind.

Als technische Realisierung wurde TCP/IP gewählt. Dies hatte zwei Gründe:

1. Datenbanksysteme sind heutzutage so groß, daß sie auf einen eigenen Rechnern laufen, der nicht durch andere Nutzungen, wie MOBAL, blockiert werden soll.

2. Der andere Grund ist die Übertragbarkeit. Das Datenbanksystem ORACLE 7 ist auf fast allen Betriebssystemebenen verfügbar, somit können auch andere ORACLE–Datenbanken, die auf anderen Betriebssystemebenen arbeiten, zum Lernen mit RDT/DB genutzen werden.

6.1.4 Die Schnittstelle

Die Schnittstelle[2] zwischen ORACLE 7 und PROLOG ist eine Implementierung in C und in RDT/DB eingebunden .

Die Anfragen von RDT/DB an die Datenbank erfolgt durch die Übergabe von Zeichenketten (einer Liste von Zeichen[3]) an die Schnittstelle. Diese werden mittels des TCP/IP Protokolls an die Datenbank weitergeleitet. Das Ergebnis einer solchen Anfrage wird in Form einer Liste über die Schnittstelle an RDT/DB zurückgegeben.

Ein besonderer Punkt ist die Behandlung von Datenbankobjekten der Typen `VARCHAR` und `CHAR`. Objekte dieser Typen können führende und abschließende Leerzeichen besitzen. Diese Leerzeichen werden durch die Schnittstelle abgetrennt.

6.2 Programmarchitektur von RDT/DB

Als externes Tool besteht RDT/DB aus drei Komponenten:

1. einer Initialisierungskomponente,

2. einer Programmkomponente und

[2] Wurde am Lehrstuhl VIII des FB Informatik an der Universität Dortmund entwickelt.
[3] Keine Strings, hier besteht ein Unterschied in QUINTUS PROLOG

3. einer Parameterkomponente.

In der Komponente zur Initialisierung von RDT/DB sind die Programmteile enthalten, die zum Laden und Speichern des Sachbereiches ausgeführt werden. Die Programmkomponente enthält den eigentlichen Programmcode von RDT/DB, während in der Parameterkomponente die Parameter mit ihren Wertebereichen definiert werden. Die Architektur von RDT/DB und Verbindung zu MOBAL und der Datenbank ist in der Abbildung 6.3 dargestellt.

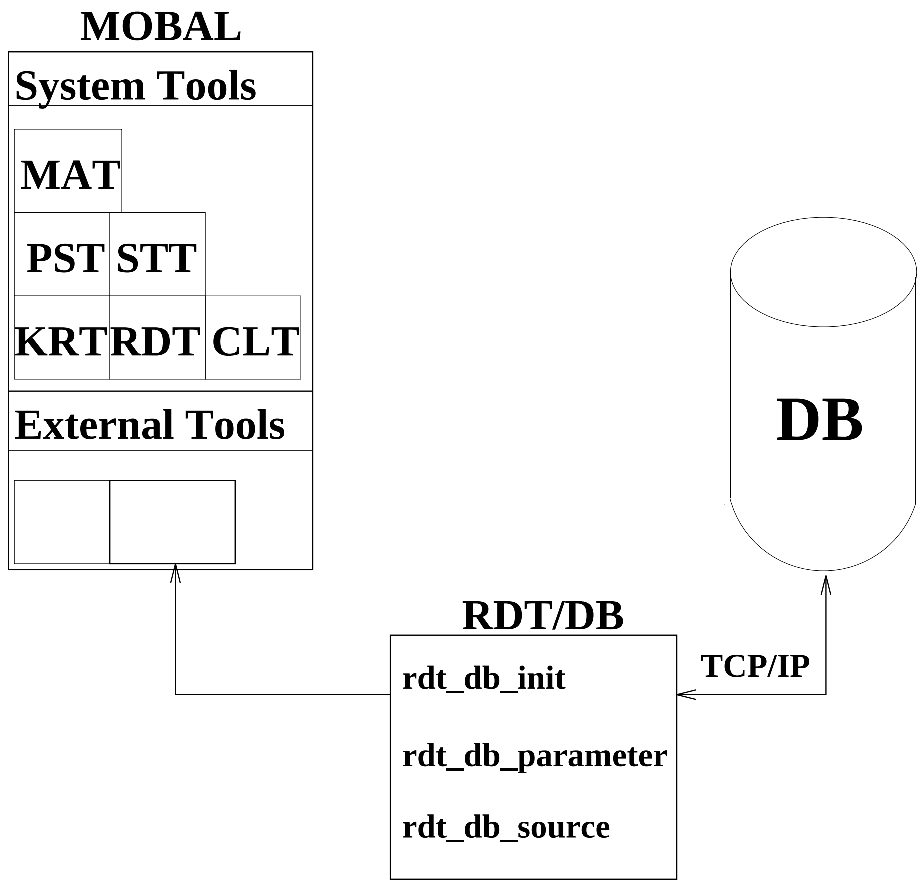

Abbildung 6.3: Verbindung von RDT/DB zu MOBAL und der Datenbank

6.2.1 Parameter

In der Parameterkomponente sind alle Parameter von RDT/DB definiert. Diese Definition umfaßt den zulässigen Wertebereich der Parameter, sowie ihre Voreinstellung. Im folgenden werden die wichtigsten Parameter von RDT/DB vorgestellt:

Die folgenden Parameter enthalten die Informationen, die zur Herstellung einer Verbindung notwendig sind:

rdt_db_oracle_sid: Die SID, *system id*, gibt an auf welche Datenbank RDT/DB zugriffen werden soll.

rdt_db_oracle_server: Dieser Parameter enthält die Bezeichnung des Datenbankservers.

rdt_db_oracle_user: Hier kann der Benutzer bestimmen unter welcher Benutzerkennung die Verbindung zur Datenbank aufgebaut werden soll.

rdt_db_oracle_passwd: Dieser Parameter enthält das Paßwort der Benutzerkennung.

Die nun folgenden Parameter sind aus dem orginal RDT übernommen:

rdt_db_list_of_metapredicates: Hier kann der Benutzer den Lernlauf auf bestimmte Regelmodelle begrenzen. Die Voreinstellung ist eine leere Liste, d.h. es werden alle in MOBAL bekannten Regelmodelle zum Lernen genutzt.

rdt_db_topologie_restrictions: Dieser Parameter ist ein Schalter. Der Benutzer entscheidet hier, ob die Prädikatentopologie in den Lernläufen berücksichtigt wird oder nicht. Die Voreinstellung ist $\boxed{\text{yes}}$.

rdt_db_variable_type_restrictions: Mit diesem Schalter wird bestimmt, ob die Ordnung $\leq_P$ für die Reihenfolge der Intanziierung der Prädikatsvariablen genutzt wird oder nicht. Die Voreinstellung ist hier $\boxed{\text{on}}$.

rdt_db_constant_learning: Mit diesem Schalter kann der Benutzer bestimmen, ob nur die besten Konstanten, in Bezug auf die Häufigkeit ihres Auftreten, oder alle Konstanten getestet werden sollen. Die Voreinstellung ist hier $\boxed{\text{best}}$.

rdt_db_min_no_of_pos_examples: Hier gibt der Benutzer die minimale Anzahl der positiven Beispielen an, die die Hypothese erfüllen muß.

rdt_db_inductive_leap_percentage: Dieser Parameter bestimmt, wieviel Prozent der bekannten Belegungen des Lernziels abgedeckt werden müssen.

rdt_db_max_no_of_exceptions: Dieser Parameter legt fest, wieviele Ausnahmen RDT/DB bei einer Hypothese toleriert.

rdt_db_max_percentage_of_exceptions: Hier kann der Benutzer festlegen, wieviele negative Beispiele, prozentual von allen bekannten Beispielen, durch eine Hypothese abgedeckt sein dürfen.

rdt_db_other_criteria: Mit diesem Parameter können weitere Bedingungen für das Akzeptanzkriterium festgelegt werden.

Zum Abschluß gibt es noch den neu eingeführten Parameter, der aus den Überlegungen in Abschnitt 5.4 resultierte.

rdt_db_spez_hypo_test: Dieser Schalter bestimmt, ob zu spezielle Hypothesen in LEAVES eingetragen werden oder nicht. Die Voreinstellung ist no.

6.3 Benutzerhinweise für RDT/DB

In diesem Abschnitt werden die wichtigsten Punkte angesprochen, die bei der Nutzung von RDT/DB berücksichtigt werden müssen.

Einbinden in MOBAL: Über den Menüpunkt Load Tool ... kann der Benutzer RDT/DB zu den schon in MOBAL enthaltenen Lernverfahren dazuladen.

Notationen: In MOBAL werden alle Datenbankprädikate groß geschrieben. Dies hat zur Folge daß Eingaben in der Prolognotation, d.h. in Anführungszeichen, gemacht werden müssen.

Parameter: Nach dem Einbinden von RDT/DB in MOBAL können die Parameter über den Menüpunkt Parameter Display kontrolliert und editiert werden.

Aufruf von RDT/DB: RDT/DB wird in MOBAL über den Menüpunkt External Tools gestartet. Nach dem Aufruf erscheint ein Fenster, in dem das Konklusionsprädikat (Lernziel) eingegeben werden muß.

Verfügbarkeit: Der Prolog Programmcode ist am Lehrstuhl VIII verfügbar.

Beispiel zur Programmnutzung

Nach dem Starten von MOBAL lädt man RDT/DB über den Menüpunkt Load Tool ... als externes Tool zu den Lernverfahren von MOBAL dazu. Ist RDT/DB in MOBAL integrier, sollte zuerst über den Menüpunkt Parameter Display die gewünschten Parameter eingestellt werden.

In Abbildung 6.4 ist das Dialog–Fenster dargestellt, in dem die Parameter gesetzt werden können. Werden die Parameter geänder, die die Informationen zum Verbindungsaufbau zur Datenbank beinhalten, so wird die Struktur der Datenbank ausgelesen und als Prädikate in die MOBAL–Wissensbasis eingetragen. Dies geschieht nach der Abbildungsvorschrift aus Abschnitt 4.2.4.

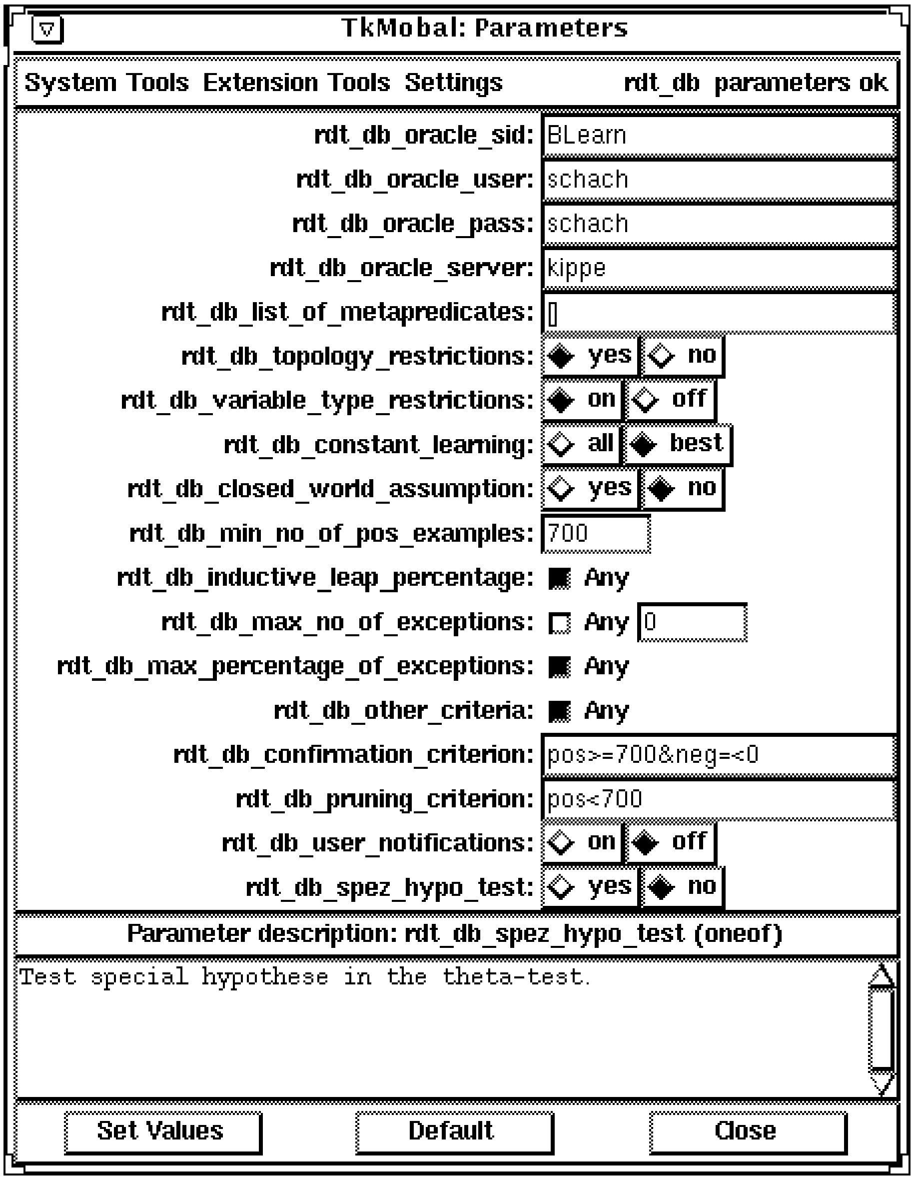

Abbildung 6.4: RDT/DB Parameter in MOBAL

Zum Starten eines Lernlaufs wählt man nun über den Menüpunkt External Tools RDT/DB aus. Hierauf erscheint ein Dialog–Fenster (siehe Abbildung 6.5), in dem das Lernziel eingegeben wird.

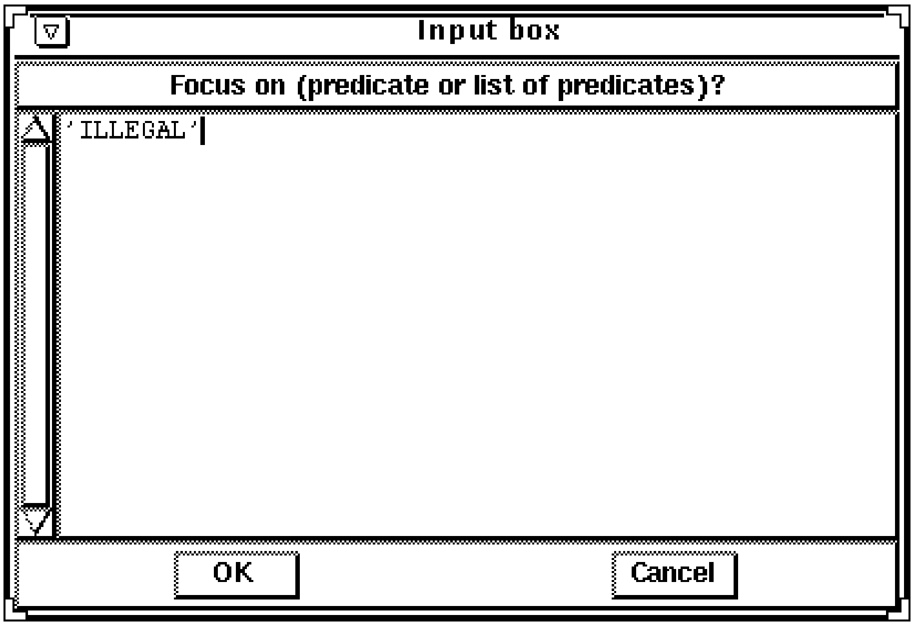

Abbildung 6.5: Fenster zur Eingabe des Lernziels

Das Lernergebnis wird in die Regelmenge von MOBAL eingefügt und kann von dem Benutzer durch Aufruf des Regelfensters betrachtet werden. Hier besteht auch die Möglichkeit gelernte Regeln aus der MOBAL–Wissensbasis zu löschen.

Kapitel 7

Tests mit RDT/DB

In diesem Kapitel werden einige Tests mit RDT/DB in den Szenarien KRK und Roboternavigation, die schon im Abschnitt 2.4 vorgestellt wurden, beschrieben. In dem KRK–Sachbereich soll in erster Line gezeigt werden, daß sich mit RDT/DB nun wesentlich größere Datenmenge bearbeiten lassen als mit RDT und daß Lernen mit RDT/DB in relationalen Datenbanken möglich ist. Hierzu wurde neben der KRK-Datenbank aus dem Abschnitt 2.4.1 eine weitere Datenbank mit einer anderen Struktur aufgebaut. Als drittes Testbeispiel wurde das Roboternavigations–Szenario ausgewählt. Die Auswahl des Roboternavigations Szenarios wurde schon in Abschnitt 2.4.2 mit der realen Anwendung begründet. Leider war die Anzahl der zur Verfügung stehenden Beispiel für die Tests nicht sehr groß. Trotz dieses Umstands gelten die Gründe für die Auswahl des Sachbereichs.

7.1 KRK

Im KRK–Sachbereich wurden zwei Testreihen durchgeführt. Im Abschnitt 7.1.1 wurde die Beispieldatenbank aus Abschnitt 2.4 als Faktenbasis genutzt. In dieser Testreihe soll der direkte Vergleich zwischen RDT und RDT/DB im Vordergrund stehen und die Anwendbarkeit von RDT/DB auf größere Datengezeigt werden. In der zweiten Testreihe liegt der Schwerpunkt in der Anwendung von RDT/DB auf große Datenmengen und dem Vergleich der Ergebnisse aus diesen Tests mit früheren Tests, die mit RDT und FOIL über diesen Sachbereich durchgeführt wurden. Für die zweite Testreihe wurde der Sachbereich in der Datenbank anders strukturiert.

7.1.1 KRK-I

Testaufbau im Lernszenario KRK-I

Die Relationen entsprechen der in der Beispieldatenbank in Abschnitt 2.4.1 vorgestellten Darstellung. Um mit RDT/DB und RDT lernen zu können, wurden folgende Regelmodelle vorgegeben:

```
m1(P1,P2,P3,P4,C)   :P1(E,X1) & P2(E,Y1) & P3(E,X1) & P4(E,Y2) &
                     ne(Y1,Y2) --> C(E).
m2(P1,P2,P3,P4,C)   :P1(E,X1) & P2(E,Y1) & P3(E,X1) & P4(E,Y1)
                     --> C(E).
m3(P1,P2,P3,P4,P5,C):P1(E,X1) & P2(E,Y1) & P3(E,X2) & P4(E,Y2) &
                     P5(X1,X2)& P5(Y1,Y2)  --> C(E).
```

Testergebnisse und Schlußfolgerung

Die Ergebnisse der Lernläufe sind in der Tabelle 7.1 zusammengefaßt.

Lernverfahren	Beispielanzahl	Regeln	Lernzeit
RDT	335	> 600	> 20 Std.
RDT/DB*[1]	335	120	2 Std. 28 Min.
RDT/DB	3504	191	12 Std. 32 Min.
RDT/DB*	3504	191	8 Std. 5 Min.

Tabelle 7.1: Lernergebnisse zu illegal

Dieser erste Test zeigt, daß RDT/DB gößere Datenmengen im Sinne des maschinellen Lernens bearbeiten kann. Trotz der fast zehnfachen Menge an Beispielen benötigte RDT/DB in beiden Einstellungen weniger Zeit als RDT. Auch in dem direkten Vergleich auf denselben Beispieldaten war RDT/DB* deutlich schneller. Die Vielzahl mehr an Regel ergeben sich bei RDT aus einem sehr schwachen Akzeptanzkriterium und der großen Anzahl von Verteilungsmöglichkeiten der Prädikate auf die Prädikatsvariablen der Regelmodelle. Hieraus ergeben sich bei RDT eine Vielzahl von redundanten Instanziierungen von Prädikaten, die bei RDT/DB nicht auftretten. RDT/DB nutzt hier die bekannten Primärschlüsseleigenschaften aus (vgl. Abschnitt 5.3.3).

Das Akzeptanzkriterium für die Tests mit 335 Beispielen war die Voreinstellung von RDT. In den Tests mit 3504 Beispielen wurde das Akzeptanzkriterium um den Faktor 10 hochgesetzt.

7.1.2 KRK-II

Testaufbau und Lernergebnisse

In diesem Beispiel sind die einzelnen Schachfiguren in eigenen Tabellen in der Datenbank abgelegt (siehe Tabellen 7.2 bis 7.5). Weiterhin ist der Zielbegriff

[1]Ohne θ–Subsumtionstest für zu spezielle Hypothesen.

illegal in einer Tabelle gespeichert. Das Hintergrundwissen adjacent ist wie in KRK-I gegeben. Die Beispiele wurden mittels eines Zufallsgenerators erzeugt. In der Datenbank waren insgesamt 12886 Beispiele enthalten. Diese Testreihe wurde nur mit RDT/DB durchgeführt, da frühere Erfahrungen gezeigt haben, das solche Datenmengen nicht in der MOBAL–Wissensbasis verarbeitet werden können.

white_king_x	white_king_y	ID
2	6	267434
...	...	...

Tabelle 7.2: white_king

black_king_x	black_king_y	ID
7	4	267434
...	...	...

Tabelle 7.3: black_king

white_rook_x	white_rook_y	ID
3	4	267434
...	...	...

Tabelle 7.4: white_rook

ID
267434
...

Tabelle 7.5: illegal

In der Prädikatendarstellung stehen RDT/DB zum Lernen folgende Prädikate zur Verfügung:

ADJACENT/2: $< x >, < y >$.

BLACK_KING/1: $< id >$.

BLACK_KING_X/2:
$< id >, < x >$.

BLACK_KING_Y/2:
$< id >, < y >$.

Rel_BLACK_KING/3:
$< id >, < x >, < y >$.

WHITE_KING/1: $< id >$.

WHITE_KING_X/2:
$< id >, < x >$.

WHITE_KING_Y/2:
$< id >, < y >$.

Rel_WHITE_KING/3:
$< id >, < x >, < y >$.

WHITE_ROOK/1: $< id >$.

WHITE_ROOK_X/2:
$< id >, < x >$.

WHITE_ROOK_Y/2:
$< id >, < y >$.

Rel_WHITE_ROOK/3:
$< id >, < x >, < y >$.

ILLEGAL/1: $< id >$.

Aus der automatischen Abbildung der Relationen in Prädikate, können auch Prädikate entstehen, die keine Verwendung in den Lernläufen finden. Offensichtlich nicht benötigte Prädikate können vom Benutzer auch wieder gelöscht werden.

Für die Lernläufe wurden folgende Regelmodelle vorgegeben:

```
krk2_m1(P1,P2,C): P1(I,X1,Y1) & P2(I,X1,Y2) --> C(I).
krk2_m2(P1,P2,C): P1(I,X1,Y1) & P2(I,X2,Y1) --> C(I).
krk2_m3(P1,P2,C): P1(I,X1,Y1) & P2(I,X1,Y1) --> C(I).
krk2_m4(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X2,Y2) & P3(X1,X2) &
        P3(Y1,Y2) --> C(I).
krk2_m5(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X2,Y1) & P3(I,X3,Y1) &
```

```
        lt(X2,X1) & lt(X3,X1) & ne(X2,X3) --> C(I).
krk2_m6(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X2,Y1) & P3(I,X3,Y1) &
        lt(X1,X2) & lt(X1,X3) & ne(X2,X3) --> C(I).
krk2_m7(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X1,Y2) & P3(I,X1,Y3) &
        lt(Y2,Y1) & lt(Y3,Y1) & ne(Y2,Y3) --> C(I).
krk2_m8(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X1,Y2) & P3(I,X1,Y3) &
        lt(Y1,Y2) & lt(Y1,Y3) & ne(Y2,Y3) --> C(I).
krk2_m9(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X2,Y2) & P3(I,X2,Y3) &
        ne(X1,X2) --> C(I).
krk2_m10(P1,P2,P3,C): P1(I,X1,Y1) & P2(I,X2,Y2) & P3(I,X3,Y2)&
         ne(Y1,Y2) --> C(I).
```

Folgende Testläufe wurden durchgeführt.

- Lernlauf I mit den Regelmodellen `krk2_m1` bis `krk2_m4` und keinen negativen Beispielen. Dieser Versuch wurde mit zwei verschiedenen Akzeptanzkriterien durchgeführt:

  a) $\mid pos(H) \mid > 1500$

  b) $\mid pos(H) \mid > 700$

- Lernlauf II mit den Regelmodellen `krk2_m3` bis `krk2_m10` und keinen negativen Beispielen. Da in diesem Test speziellere Regeln gelernt werden sollen, wurde das Akzeptanzkriterium auf $\mid pos(H) \mid > 700$ abgeschwächt.

- Lernlauf III mit allen Regelmodellen und vier negativen Beispielen in der MOBAL–Wissensbasis. Die negativen Beispiele beschreiben einen Spezialfall, der nur durch wenige Beispiele vertreten ist. Es handelt sich hierbei um die Situation, wenn die drei Spielfiguren in einer Linie stehen und der weiße König zwischen dem schwarzen König und dem Turm steht. Die negativen Beispiele waren:

  1. `not(ILLEGAL(252327))`
  2. `not(ILLEGAL(444541))`
  3. `not(ILLEGAL(376717))`
  4. `not(ILLEGAL(532373))`

  Das Akzeptanzkriterium aus dem Lernlauf II wurde um die Bedingung das keine negativen Beispiele abgedeckt werden dürfen erweitert.

Lernergebnisse

Lernlauf Ia): In der Abbildung 7.1 sind die von RDT/DB gelernten Regeln aus dem Lernlauf I dargestellt. Der Benutzer muß nun, wie im Lernszenario in Abschnitt 1.2 beschrieben, entscheiden, welche Regeln in der Wissensbasis von MOBAL bleiben sollen.

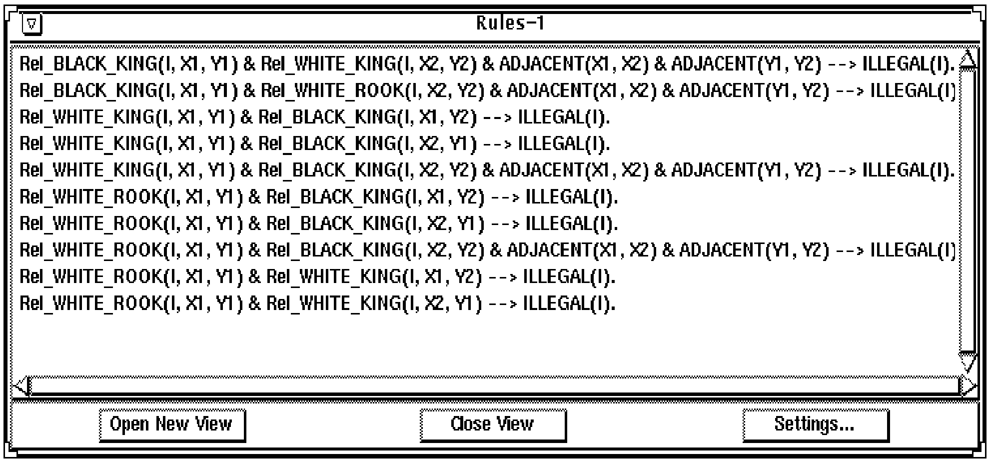

Abbildung 7.1: Gelernte Regeln aus den Lernlauf I

Bei einer genaueren Betrachtung der gelernten Regeln stellt der Benutzer Redundanzen in der Regelmenge fest. Solche redundante Regeln können von dem Benutzer dann gelöscht werden. Diese semantischen Redundanzen können von MOBAL oder RDT/DB nicht aufgedeckt werden. An zwei solcher Regeln soll deutlich gemacht werden, warum dies nicht möglich ist. Betrachte werden sollen folgende zwei Regeln:

1. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_KING(I,X2,Y2) & ADJACENT(X1,X2) & ADJACENT(Y1,Y2) --> ILLEGAL(I).`

2. `Rel_WHITE_KING(I,X1,Y1) & Rel_BLACK_KING(I,X2,Y2) & ADJACENT(X1,X2) & ADJACENT(Y1,Y2) --> ILLEGAL(I).`

Der Benutzer weiß in diesem Fall, das `adjacent` eine Äquivalenzrelation ist und kann deshalb entscheiden, daß die Regeln redundant sind. Würde statt des Prädikats `adjacent` das *built-in* Prädikat `lt` instanziiert so wären die Aussagen nicht redundant. So kann alleine aus dem Aufbau der Regeln nicht auf die Redundanz von Regeln geschlossen werden.

In der folgenden Aufzählung ist eine aus der Regelmenge in Abbildung 7.1 resultierende Regelmenge ohne Redundanzen aufgeführt.

1. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_KING(I,X2,Y2) & ADJACENT(X1,X2) & ADJACENT(Y1,Y2) --> ILLEGAL(I).`

2. `Rel_WHITE_ROOK(I,X1,Y1) & Rel_BLACK_KING(I,X1,Y2) --> ILLEGAL(I).`

3. `Rel_WHITE_ROOK(I,X1,Y1) & Rel_BLACK_KING(I,X2,Y1) --> ILLEGAL(I).`

Lernlauf Ib): In diesem Test wurden 15 Regeln gelernt, von denen sechs in die Wissensbasis übernommen werden konnten. Die ersten drei sind dieselben wie im Lernlauf Ia). Hinzu kommen die folgenden Regeln:

1. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_KING(I,X1,Y1)`
   `--> ILLEGAL(I).`

2. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X1,Y1)`
   `--> ILLEGAL(I).`

3. `Rel_WHITE_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X1,Y1)`
   `--> ILLEGAL(I).`

Lernlauf II: Insgesamt wurden in diesem Test 18 Regeln gelernt. Nach der Kontrolle des Benutzers ergibt sich ein Lernergebnis bestehend aus folgenden sieben Regeln:

1. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_KING(I,X1,Y1)`
   `--> ILLEGAL(I).`

2. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X1,Y1)`
   `--> ILLEGAL(I).`

3. `Rel_WHITE_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X1,Y1)`
   `--> ILLEGAL(I).`

4. `Rel_BLACK_KING(I,X1,Y1) & Rel_WHITE_KING(I,X2,Y2) &`
   `ADJACENT(X1,X2) & ADJACENT(Y1,Y2) --> ILLEGAL(I).`

5. `Rel_WHITE_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X2,Y2) &`
   `Rel_BLACK_KING(I,X3,Y3) & lt(X2,X1) & lt(X3,X1) & ne(X2,X3)`
   `--> ILLEGAL(I).`

6. `Rel_WHITE_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X2,Y2) &`
   `Rel_BLACK_KING(I,X3,Y3) & ne(X1,X2) --> ILLEGAL(I).`

7. `Rel_WHITE_KING(I,X1,Y1) & Rel_WHITE_ROOK(I,X2,Y2) &`
   `Rel_BLACK_KING(I,X3,Y3) & ne(Y1,Y2) --> ILLEGAL(I).`

Lernlauf III: In der Abbildung 7.2 sind die von beiden Algorithmen gelernten Regeln abgebildet.

Nach der Kontrolle des Lernergebnisses durch den Benutzer, bleiben dieselben sieben Regeln über wie im Lernlauf II.

In der Tabelle 7.6 sind die Lernzeiten der Testläufe zusammengefaßt.

Test	Beispielanzahl	Lernzeit RDT/DB	Lernzeit RDT/DB*
Ia)	12886	1 Std. 22 Min.	34 Min.
Ib)	12886	32 Min.	33 Min.
II	12886	1 Std. 6 Min.	1 Std. 9 Min.
III	12886	55 Min.	39 Min.

Tabelle 7.6: Lernergebnisse zu illegal aus KRK II

7.1.3 Diskussion der Ergebnisse

Bevor die Ergebnisse im einzelnen analysiert werden, sollen an dieser Stelle zwei Bewertungskriterien, Vollständigkeit und Korrektheit, vorgestellt werden.

- **Vollständigkeit:** Vollständigkeit ist die vollständige Ableitung aller tatsächlichen Extensionen eines Begriffs aus dem Sachbereich.

$$\frac{\mid tat\_ext(c) \cap ext(c) \mid}{\mid tat\_ext(c) \mid}$$

  wobei $ext(c)$ die Menge aller Beispiele ist, die durch das Lernergebnis als c klassifiziert werden und tat_ext ist die Menge aller Beispiele, die dem Begriff c in der Realität angehören.

- **Korrektheit:** Werden keine Extensionen im Widerspruch zur tatsächlichen Extension abgeleitet, so ist eine Hypothese korrekt.

$$1 - \frac{\mid not(tat\_ext(c)) \cap ext(c) \mid}{\mid ext(c) \mid}$$

Diese Kriterien sind eindeutig berechenbar und daher leicht zu bewerten und vergleichen. Die Kriterien wurden auch schon im experimentellem Vergleich von RDT und FOIL über KRK–Sachbereich verwendet. Hier wurde auch dieselbe Repräsentation verwendet. In der Tabelle 7.7 sind die Werte für die Vollständikeit und Korrektheit der Lernläufe zusammengefaßt.

Das schlechte Ergebnis im Lernlauf Ia), bezüglich der Korrektheit und Vollständigkeit, war aufgrund der kleinen Regelmenge zu erwarten. Hieraus folgt, daß das Akzeptanzkriterium zu hoch gewählt war. Das Ergebnis des Lernlaufs Ib) ist im Bezug auf die Vollständigkeit optimal, während der Wert für die Korrektheit immer noch schlecht ist. Dies bedeutet, daß die Regeln zu generell sind.

In den Lernläufen II und III wurden erheblich bessere Ergebnisse erzielt, die den Ergebnissen, die z.B. mit FOIL in derselben Repräsentation erreicht

Lernlauf	Vollständigkeit	Korrektheit
Ia)	0.47	0.46
Ib)	1	0.58
II/III	0.87	0.93

Tabelle 7.7: Vollständigkeit und Korrektheit

wurden, sehr nahe kommen. Die Tests mit FOIL und RDT wurden mit einer Beispielmengen von bis zu 500 Stück durchgeführt. Diese Tests wurden von Ursula Robers und Guido Lindner beschrieben [Lindner und Robers, 1994]. In diesem Test[2] erzielte FOIL für die Korrektheit einen Wert von 0.98 und einen Vollständigkeitswert von 0.93. Das Ergebnis von RDT war in diesem Test besser. Die von RDT gelernten Regeln hatten einen Korrektsheitswert von 0.98 und einen Vollständigkeitswert von 1.

Die Unterschiede in den Werten der Bewertungskriterien zu den Lernläufen II und III im Vergleich mit RDT, sind nach einer Betrachtung der Lernergebnisse der Lernläufe II und III leicht erklärbar. In diesen Lernläufen wurde nur eine Regeln zu dem Fall gelernt, daß alle drei Spielfiguren in einer Zeile oder Spalte stehen. Dieser Fall wird aber erst durch vier Regeln vollständig beschrieben. Durch ein schwächeres Akzeptanzkriterium würde auch RDT/DB die fehlenden Regeln lernen.

Bemerkenswert ist im Vergleich dieser Tests weiterhin, daß RDT für den verglichenen Lernlauf 1 Stunde und 57 Minuten benötigte, während RDT/DB für 12886 Beispiele 1 Stunde und 6 Minuten benötigte. Dieser Sachverhalt macht deutlich, daß die Bearbeitung von großen Datenmengen mit RDT/DB möglich ist.

Der Lernlauf III unterscheidet sich von allen anderen Tests, da dort negative Beispiele berücksichtigt wurden. In diesem Test waren für spezielle Situationen, negative Beispiele in der MOBAL–Wissensbasis vorgegeben. Diese Beispiele sollten verhindern, daß zu allgemeine Regeln für den Fall, das die drei Spielfiguren in einer Zeile oder Spalte stehen gelernt werden. Durch die negativen Beispiele, die nach dem Akzeptanzkriterium nicht abgedeckt sein durften, wurden die folgenden zu allgemeinen Regeln auch nicht gelernt:

- `Rel_WHITE_ROOK(I,X1,Y1) & Rel_BLACK_KING(I,X1,Y2)`
  `--> ILLEGAL(I).`

- `Rel_WHITE_ROOK(I,X1,Y1) & Rel_BLACK_KING(I,X2,Y1)`
  `--> ILLEGAL(I).`

[2] Dieser Test wurde mit 500 Beispielen durchgeführt, von denen 30% negative Beispiele waren.

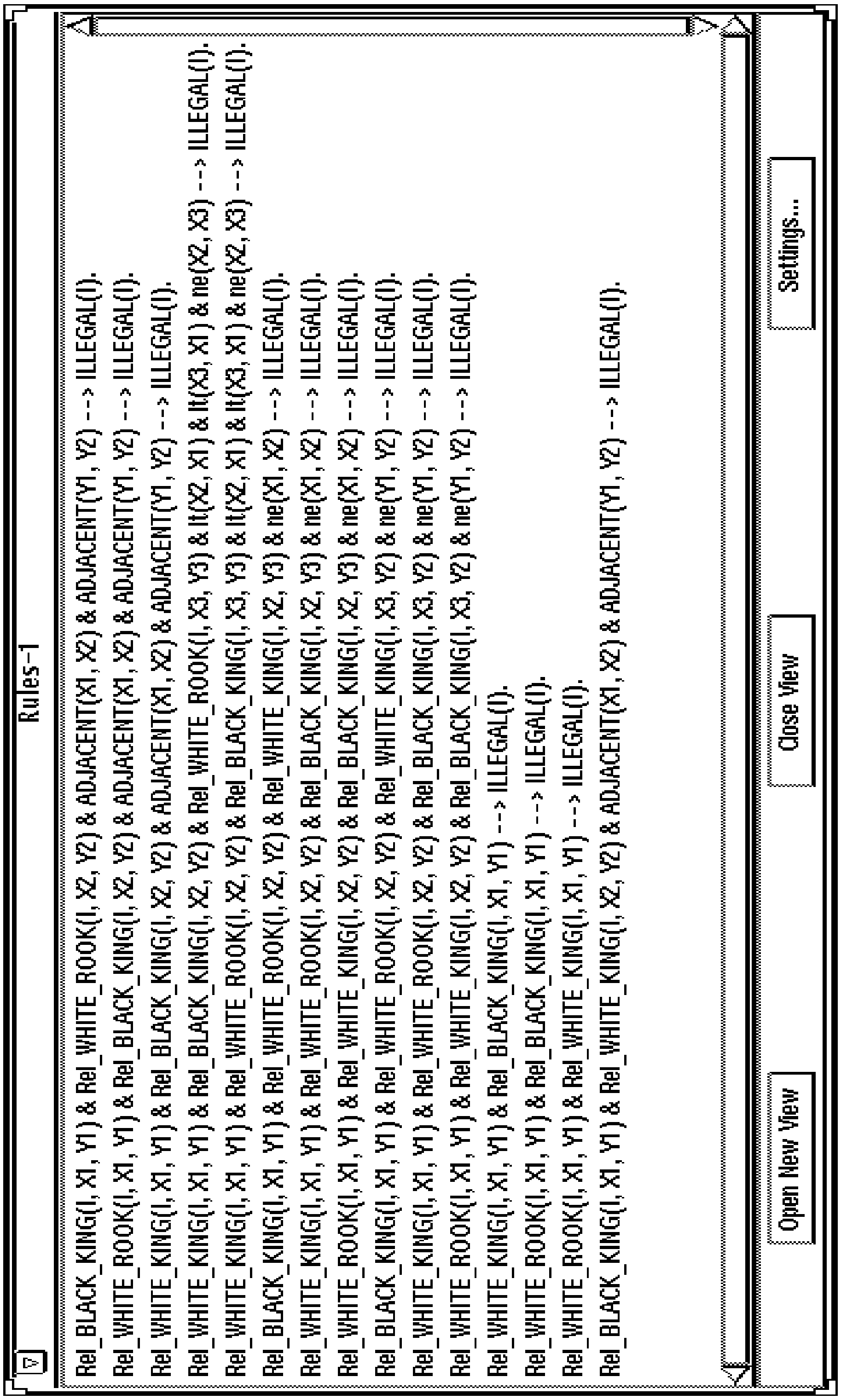

Abbildung 7.2: Gelernte Regeln aus den Lernlauf III

7.1.4 Schlußfolgerung aus den KRK–Tests

Die Tests im KRK–Bereich haben gezeigt, daß mit RDT/DB wesentlich größere Datenmengen verarbeiten lassen. Die Lernzeiten von RDT/DB sind im Vergleich mit RDT sogar besser, obwohl die Beispielmengen in KRK II z.B. für RDT/DB um den Faktor 24 größer waren. Die Lernzeiten der einzelnen Tests sind insofern vergleichbar, daß MOBAL immer in der selben Rechnerumgebung auf einer SPARC STATION ELC gestartet wurde.

7.2 Roboternavigation

Dieser Abschnitt beschreibt den Aufbau und die Lernergebnisse aus dem Sachbereich Roboternavigation, der in Abschnitt 2.4.2 vorgestellt wurde.

Zu Beginn soll das Lernziel wiederholt werden. Lernziel ist es, Abfolgen von *basic features* zu lernen, die ein *sensor feature* beschreiben, d.h. ein *sensor feature* beschreibt das Verhalten eines Sensors vom Zeitpunkt t1 bis tn. Diese *sensor features* werden zur Interpretation der Umgebung genutzt.

7.2.1 Testaufbau

Neben den *basic features* und den *sensor features* wurden fünf Regelmodelle für die Lernläufe verwendet, von denen drei exemplarisch angeben sind. Die insgesamt vorgegebenen Regelmodelle beschreiben eine Verkettung von bis zu 6 Prädikaten.

```
m2(P1,P2,M,C)       :P1(T,S,X1,X2) & P2(T,S,X2,X3)
                     --> C(T,S,X1,X3,0).
m3(P1,P2.P3,M,C)    :P1(T,S,X1,X2) & P2(T,S,X2,X3) &
                     P3(T,S,X3,X4) --> C(T,S,X1,X4,0).
m4(P1,P2.P3,P4,M,C):P1(T,S,X1,X2) & P2(T,S,X2,X3)
                     & P3(T,S,X3,X4) & P4(T,S,X4,X5)
                     --> C(T,S,X1,X5,0).
```

Die Tests wurden mit RDT, RDT/DB und RDT/DB ohne θ–Subsumtionstests der zu speziellen Hypothesen durchgeführt. Die Anzahl der Fakten, die zu den einzelnen Lernzielen zur Verfügung standen, können aus der Tabelle 7.8 abgelesen werden.

7.2.2 Lernergebnisse

In der Tabelle 7.8 ist die Anzahl der Regeln zu den einzelnen Lernzielen dargestellt, die von allen drei Verfahren erreicht wurden. Die gelernten Regeln waren bei allen drei Verfahren identisch.

Die Lernzeiten, die die Verfahren jeweils für die einzelnen Lernziele benötigten, sind der Tabelle 7.9 zu entnehmen.

Lernziel	Anzahl der Fakten	Anzahl der Regeln
s_line	3137	64
s_jump	2665	48
s_convex	2516	35
s_concave	2482	8

Tabelle 7.8: Lernergebnisse zum Sachbereich Roboternavigation

Lernziel	Lernzeit RDT	Lernzeit RDT/DB	Lernzeit RDT/DB*
s_line	111 Std. 46 Min.	110 Std. 34 Min.	7 Std. 38 Min.
s_jump	52 Std. 16 Min.	65 Std. 16 Min.	3 Std. 51 Min.
s_convex	24 Std. 12 Min.	31 Std. 6 Min.	2 Std. 20 Min.
s_concave	1 Std. 45 Min.	1 Std. 39 Min.	20 Min.

Tabelle 7.9: Lernzeiten im Sachbereich Roboternavigation

7.2.3 Schlußfolgerung

Der Sachbereich Roboternavigation wurde ausgewählt um eine reale Anwendung zu zeigen. RDT/DB erzielte in diesem Sachbereich von den gelernten Regeln das gleiche Ergebnis wie RDT, das in diesem Projekt eingesetzt wurde. Die Lernzeiten von beiden Verfahren RDT und RDT/DB mit dem θ–Subsumtionstest für zu spezielle Hypothesen unterscheiden sich nicht wesentlich. Die Differenzen in den Lernzeiten lassen sich noch mit den äußeren Einflußfaktoren, z.B Auslastung des lokalen Netzes während der Tests, erklären. Anders ist es mit Lernzeiten von RDT/DB ohne θ–Subsumtionstest für zu spezielle Hypothesen. Hier zeigt die Modifikation von RDT/DB eine sehr große Auswirkung. Die Tests waren bis zu einem Faktor 20 schneller als mit den beiden anderen Algorithmen.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Lernen in relationalen Datenbanken

Ein Ziel dieser Arbeit war, relationale Datenbanken als Beispielmenge für das Lernverfahren RDT zugänglich zu machen, um mit dem Verfahren über relationale Datenbanken zu lernen.

Um dieses Ziel zu erreichen, wurde eine Repräsentation der Datenbankstruktur in eine Prädikatendarstellung entwickelt und das Verfahren auf die spezifischen Datenbankeigenschaften angepaßt. Durch die Nutzung der Primärschlüsseleigenschaften konnte der Hypothesenraum zusätzlich durch einen Test der Datentypen (siehe Abschnitt 5.3.2) und einem Test auf redundante Prädikate (siehe Abschnitt 5.3.3) eingeschränkt werden, während auf eine Ausnutzung der Sortenbeziehungen verzichtet wurde. Weiterhin konnte das Verfahren durch die Idee, die zu speziellen Hypothesen nicht in dem θ–Subsumtionstest zu berücksichtigen, verbessert werden.

Aus den Tests in Kapitel 7 wird deutlich, daß RDT/DB in relationalen Datenbanken in dem getesteten Umfang gut lernen kann. Insbesondere hat sich die Variante von RDT/DB ohne θ–Subsumtionstest für zu spezielle Hypothesen in den Tests als sehr effektiv erwiesen, so daß dieses Vorgehen auch für RDT zu empfehlen ist.

Weiterhin erfüllt RDT/DB die Anforderungen an ein KDD–Verfahren nach der Definition 1 aus dem Abschnitt 1.1.1.

8.2 MOBAL und Datenbanken

Mit der Anbindung von RDT sollte die Frage beantwortet werden, ob die Anbindung einer Komponente von MOBAL (RDT) sinnvoll ist, oder ob der Aufwand durch die Kommunikation über ein Netz und der Verwaltung der Daten in einem Datenbank–Managementsystem zu groß ist. Aus den Tests im Sachbereich der Roboternavigation kann man erkennen, daß die Zugriffe auf die Fakten — beide Faktenbasen waren gleich groß —, in der Datenbank nur einen geringen mehr Aufwand bedeuteten, der sich bei größeren Faktenbasen relativiert. In den Tests hat sich gezeigt, daß weiterhin die θ–Subsumtionstests der zeitaufwendigste Teil in RDT/DB ist. Die Vorüberlegungen, daß der Aufwand durch die Kommunikation über ein Netz und die Verwaltung der Daten in einem Datenbanksystem zu groß sein könnten, hat sich nicht bestätigt.

8.3 Ausblick

Diese Arbeit ist die erste am Lehrstuhl VIII, die sich mit der Thematik der Anwendung von maschinellem Lernen auf relationale Datenbanken beschäfftigt. In Zukunft ist die Anwendung von weiteren Verfahren der induktiven logischen Programmierung geplant, so daß dann auch Vergleiche zwischen diesen Verfahren und RDT/DB durchgeführt werden können. Zum Zeitpunkt dieser Arbeit standen leider weder anderen Verfahren für eigene Tests, noch Beispieldatenbanken, die für andere Verfahren als Testmenge genutzt wurden, zur Verfügung. Solche Vergleichstests wären mit Sicherheit sehr interessant.

Nach den Erfahrungen aus den Tests mit RDT/DB wäre auch eine Anwendung von RDT/DB in dem Testszenario der Roboternavigation auf sehr große Mengen von Sensordaten, im Hinblick auf eine reale Anwendung, von großen Interesse.

Verbesserungen
In RDT/DB wurde auf die Ausnutzung der Sortenbeziehungen verzichtet (vgl. Abschnitt 5.3.1). Diese kann bestimmt in vielen Fällen sehr effektive sein, so daß an dieser Stelle RDT/DB sinnvoll erweitert werden könnte. Dies sollte auf alle Fälle geschehen, wenn das Datenbanksystem ORACLE7 benutzerdefinierte Integritätsbedingungen voll unterstützt.

Fazit
Zur Zeit gibt es nur wenige Verfahren aus dem Bereich der induktiven logischen Programmierung, die auf relationale Datenbanken angewendet werden können. Die guten Erfahrungen dieser Arbeit sind eine weitere Motivation, andere Verfahren auf relationale Datenbanken anzuwenden.

Literaturverzeichnis

[Bobrowski, 1993] S. Bobrowski. *ORACLE Server Begriffe*. ORACLE Corporation, Juni 1993.

[Ceri *et al.*, 1990] S. Ceri, G. Gottlob und L. Tanca. *Logic Programming and Databases*. Springer Verlag, Berlin, New York, 1990.

[Clark und Niblett, 1989] P. Clark und T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261–284, 1989.

[Date, 1990] C.J. Date. *An introduction to database systems*, Band I der *The systems programming series*. Addison Wesley, 5 Auflage, 1990.

[Džeroski und Lavrač, 1993] S. Džeroski und N. Lavrač. Inductive Learning in Deductive Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):939–949, Dezember 1993.

[Finkenzeller *et al.*, 1989] H. Finkenzeller, U. Kracke und M.Unterstein. *Systematischer Einsatz von SQL–ORACLE*. Addison Wesley, 1989.

[Frawley *et al.*, 1991] W. Frawley, G. Piatetsky-Shapiro und C. Matheus. Knowledge Discovery in Databases: An Overview. In G. Piatetsky-Shapiro und W. Frawley (Hrsg.), *Knowledge Discovery in Databases*, S. 1–27, Cambridge, Mass., 1991. AAAI/MIT Press.

[Gulbins, 1988] J. Gulbins. *UNIX Version 7 bis System V.3*. Springer Verlag, 1988.

[Holsheimer und Siebes, 1993] M. Holsheimer und A. Siebes. Data Mininig: The Search for Knowledge in Databases. CS-R9406, CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherland, 1993. ISSN 0169-118X.

[Kietz und Lübbe, 1994] J. U. Kietz und M. Lübbe. An Efficient Subsumption Algorithm for Inductive Logic Programming. In *Proceedings of the 11th Conference of Maschine Learning*, 1994.

[Kietz und Wrobel, 1992] 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, 1992.

[Kietz, 1988] Jörg-Uwe Kietz. Incremental and Reversible Acquisition of Taxonomies. *Proceedings of EKAW-88*, S. 24.1–24.11, 1988. Also as KIT-Report 66, Technical University Berlin.

[Klingspor, 1991] Volker Klingspor. Abstraktion von Inferenzstrukturen in MOBAL. Diplomarbeit, Univ. Bonn, 1991.

[Klingspor, 1994] V. Klingspor. GRDT: Enhancing Model-Based Learning for Its Application in Robot Navigation. Forschungsbericht 518/94, Universität Dortmund, Fachbereich Informatik, 1994. ISSN 0933-6192.

[Lavrač und Džeroski, 1992] N. Lavrač und S. Džeroski. Inductive Learning of Relations from Noisy Data. In S. Muggleton (Hrsg.), *Inductive Logic Programming*, Kapitel 25, S. 495–516. Academic Press, San Diego, CA 92101, 1992.

[Lavrač und Džeroski, 1994] Nada Lavrač und Sašo Džeroski. *Inductive Logic Programming — Techniques and Applications*. Nummer 148 in Artificial Intelligence. Ellis Horwood, Hertfortshire, 1994.

[Lindner und Robers, 1994] G. Lindner und U. Robers. Experimentelle Analyse zweier logik –basierter Lernverfahren. Forschungsbericht 6, Universität Dortmund, Fachbereich Informatik, Lehrstuhl VIII, 1994. ISSN 0943-4135.

[Lockemann und Schmidt, 1987] P.C. Lockemann und J.W. Schmidt. *Datenbank–Handbuch*. GI Informatik Handbücher. Springer Verlag, 1987.

[Matheus *et al.*, 1993] C.J. Matheus, P. Chan und G. Piatetsky-Shapiro. Systems for Knowledge Discovery in Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):903–913, Dezember 1993.

[Michalski *et al.*, 1986] Ryszard Michalski, Ivan Mozetic, Jan Hong und Nada Lavrač. The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In *Procs. of the National Conference on Artificial Intelligence*, S. 1041 – 1045, San Mateo, CA, 1986. Morgan Kaufmann.

[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 und Rieger, 1993] Katharina Morik und Anke Rieger. Learning Action-Oriented Perceptual Features for Robot Navigation. In Attilio Giordana (Hrsg.), *Learning Robots - Proceedings of ECML Workshop*. 1993.

[Morik, 1989] Katharina Morik. Sloppy Modeling. In Katharina Morik (Hrsg.), *Knowledge Representation and Organization in Machine Learning*, S. 107–134. Springer Verlag, Berlin, New York, 1989.

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

[Muggleton, 1992] Stephen Muggleton. *Inductive Logic Programming*. Nummer 38 in APIC series. Academic Press, London, 1992.

[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, 1971] 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.

[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, 1986] R.J. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

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

[Schlageter und Stucky, 1983] G. Schlageter und W. Stucky. *Datenbanksysteme: Konzepte und Modelle*. Teubner Verlag, Stuttgart, 1983.

[Shapiro, 1981] E.Y. Shapiro. An Algorithm that Infers Theories from Facts. In *Proc of the seventh IJCAI-81*, S. 446–451, 1981.

[Ullmann, 1988] J.D. Ullmann. *Clasical Database Systems*, Band I der *Principles of Databases and Knowledge-Base Systems*. Computer Science Press, 1988.

[Ullmann, 1989] J.D. Ullmann. *The New Technologies*, Band II der *Principles of Databases and Knowledge-Base Systems*. Computer Science Press, 1989.

[Weber, 1993] R. Weber. SQL2-Norm und SQL3-Projekt. *Informatik Spektrum*, 16(2):95, April 1993.