UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

LEHRSTUHL VIII
KÜNSTLICHE INTELLIGENZ

Logikbasiertes Lernen in relationalen Datenbanken

LS–8 Report 12

Guido Lindner

Dortmund, 30. September 1994

Universität Dortmund
Fachbereich Informatik

UNI DO

University of Dortmund
Computer Science Department

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

ISSN 0943-4135

Anforderungen an:

Universität Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

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

ISSN 0943-4135

Requests to:

University of Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

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

Logikbasiertes Lernen in relationalen Datenbanken

LS–8 Report 12

Guido Lindner

Dortmund, 30. September 1994

Universität Dortmund
Fachbereich Informatik

Zusammenfassung

Die Verbindung von Verfahren des maschinellen Lernens mit relationalen Datenbanken ist für verschiedene Bereiche, z.B. Knowledge Discovery, Deduktive Datenbanken und maschinelles Lernen selbst, von steigendem Interesse. Dieser Artikel beschreibt die Anwendung des logikorientierten Lernverfahrens RDT auf relationle Datenbanken. In diesem Rahmen wird die Frage der Repräsentation der Datenbank für das logikbasierte Lernen diskutiert. In ersten experimentellen Test über dem im maschinellen Lernen bekannten KRK–Sachbereich und einer Anwendung aus dem Bereich der Roboternavigation werden einige Vorteile und Nachteile von RDT/DB deutlich.

1 Einleitung

Im Rahmen der Wissensentdeckung in relationalen Datenbanken versucht man immer mehr auch Verfahren des maschinellen Lernens einzusetzen. Hier bieten sich logikbasierte Verfahren aufgrund ihrer Leistungsfähigkeit an. Im logikbasierten Lernen liegt das Lernergebnis in eingeschränkter Prädikatenlogik vor und ist von einem System interpretierbar. Andererseits besteht auch von der Seite des maschinellen Lernens ein großes Interesse, durch eine Anbindung an Datenbanken sich größere Beispielmengen zugänglich und handhabbar zu machen.

Im nächsten Abschnitt werden deshalb erst einmal die Berührungspunkte der verschiedenen Disziplinen, die ein Interesse am Einsatz von Lernverfahren in relationalen Datenbanken haben, beschrieben. Im weiteren wird die Anwendung des logikbasierten Lernverfahrens RDT [Kietz und Wrobel, 1992] auf relationale Datenbanken gezeigt. Im Abschnitt 3 wird das Verfahren vorgestellt, und der folgende Abschnitt beschreibt besondere Punkte der Anpassung von RDT an relationale Datenbanken. Der Abschnitt 6 zeigt dann die Unterschiede zwischen RDT, angewandt auf relationale Datenbanken (RDT/DB), und dem ursprünglichen RDT auf.

Im Abschnitt 7 werden erste experimentelle Vergleichstests zwischen RDT und RDT/DB, an Hand von zwei Testszenarien, beschrieben. Zum einen wurde hierfür der im maschinellen Lernen bekannte Sachbereich KRK [Quinlan, 1983] verwendet, und zum anderen eine reale Anwendung aus dem Bereich der Roboternavigation.

2 Induktives Lernen und relationale Datenbanken

2.1 Maschinelles Lernen im Knowledge Discovery

Im Rahmen des Knowledge Discovery (KDD) erhält das Maschinelle Lernen (ML) eine immer größere Bedeutung. Aus der Definition für Knowledge Discovery von Frawley, Piatetsky-Shapiro und Matheus wird die Ähnlichkeit der Zielsetzung 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.

Aus der Definition 1 sieht man die dem Lernen aus Beispielen ähnliche Zielsetzung. Auch hier geht man von einer Sprache $\mathcal{L}_H$ zur Beschreibung der Hypothese und einer Menge von Beispielen (Fakten) aus. Das Ergebnis eines maschinellen Lernvorgangs ist eine Hypothese, die den Kriterien des Benutzers genügt. Diese Hypothese könnte man auch als eine Menge von Schemata im Sinne des Knowledge Discovery ansehen. In diesem Sinne bezeichnen Matheus, Chan und Piatetsky-Shapiro Knowledge Discovery 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 Schließen 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/Data Mining 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], CN2 [Clark und Niblett, 1989], ...

2.2 Induktives Lernen in deduktiven Datenbanken

Eine weitere Anwendung und damit eine weitere Motivation, ein Verfahren des induktiven Lernens auf relationale Datenbanken anzuwenden, ist die Möglichkeit, diese 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 2.1 und 2.2 genannten Verfahren handelt es sich ausschließlich um Attributwerte-Verfahren. Hier ist es nun von Interesse, ein logikorientiertes Verfahren wie RDT in Zusammenhang mit relationalen Datenbanken einzusetzen.

3 Das Lernverfahren RDT

RDT [Kietz und Wrobel, 1992] ist Bestandteil des Modellierungssystems MOBAL [Morik *et al.*, 1993] und wurde an der GMD entwickelt.

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

Bevor nun auf die Besonderheiten beim Lernen mit RDT über relationale Datenbanken eingegangen wird, soll an dieser Stelle das Verfahren RDT aus der Sicht des maschinellen Lernens charakterisiert werden.

RDT ist ein Verfahren aus dem *inductive logic programming* (ILP). Die Lernaufgabe von RDT kann wie folgt beschrieben werden:

Gegeben: Hintergrundwissen und eine Menge positiver und negativer Beispiele für einen zu lernenden Begriff C in funktionsfreier Klausellogik.

Ziel: Finde eine Hypothese H in funktionsfreier Klausellogik, die einem vom Benutzer definierten Akzeptanzkriterium genügt.

Das Akzeptanzkriterium setzt sich aus den folgenden Faktoren zusammmen:

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 $H : P(X_1, \ldots, X_m) \rightarrow q(Y_1, \ldots, Y_n), \forall\ Y_i : Y_i \in (X_1, \ldots, X_m), 1 \leq i \leq n$.

Dann sind mögliche Faktoren des Akzeptanzkriteriums:

- Anzahl der positiven Beispiele, die durch die Hypothese abgedeckt sind:

  $\mid pos(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge q(c_1, \ldots, c_n)\} \mid$

- Anzahl der negativen Beispiele, die durch die Hypothese abgedeckt sind:

  $\mid neg(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge not(q(c_1, \ldots, c_n))\} \mid$

- Anzahl der Instanzen, für die noch nicht bekannt ist, ob die Schlußfolgerung erfüllt ist:

  $\mid pred(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m) \wedge unknown(q(c_1, \ldots, c_n))\} \mid$

- Anzahl aller Beispiele der Hypothese:

  $\mid total(H) \mid := \mid \{(c_1, \ldots, c_n) \mid P(c_1, \ldots, c_m)\} \mid = \mid pos(H) \cup neg(H) \cup pred(H) \mid$

- 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:

  $\mid uncover(H) \mid := \mid concl(H) \backslash pos(H) \mid$

Da bei RDT die Hypothesensprache eingeschränkte Prädikatenlogik ist, wird somit ein sehr großer Hypothesenraum beschrieben. Bei RDT wird der Hypothesenraum durch die Vorgabe von Schemata von Regeln 1. Stufe eingeschränkt. Diese definieren die syntaktische Form der möglichen Regeln. Ein Regelschema[2] ist also eine Regel, die anstatt der Sachbereichsprädikate Prädikatvariablen enthält.

[2] Ich verwende hier die Begriffe Regelmodelle, Regelschemata und Metaprädikate synonym.

Die Regelschemata werden nach einer erweiterten Form der θ–Subsumtion [Plotkin, 1970], ihrer Allgemeinheit nach partiell geordnet. RDT beginnt mit dem generellsten Regelschema und geht dann zu den spezielleren (Top–Down–Strategie). Hierdurch ergibt sich eine weitere Einschränkung des Hypothesenraums. Spezialisierungen von akzeptierten oder von nach dem Akzeptanzkriterium als zu speziell bewerteten Hypothesen müssen nicht mehr getestet werden. Die Reihenfolge, in der die Prädikatvariablen instanziiert werden, erfolgt nach der *relation chain* der Attributvariablen der einzelnen Prädikatvariablen.

Die *relation chain* beschreibt die Verknüpfung der Attributvariablen in den Regelmodellen. Es werden nur Prädikatvariablen instanziiert, die in einer solchen Verknüpfungsbeziehung mit den bereits belegten Prädikatvariablen stehen (siehe Beispiel Sortentaxonomie).

Weiterhin können von RDT zur Instanziierung der Prädikatvariablen die Prädikattopologie und die Sortentaxonomie ausgenutzt werden.

Prädikatentopologie
Prädikate können strukturiert, in Form eines Graphen, dessen Knoten Prädikate zugeordnet sind, repräsentiert werden. Eine Hypothese ist mit einer Prädikattopologie kompatibel, wenn die Prädikate der Prämisse in einem Vorgängerknoten oder im Knoten des Prädikats der Konklusion sind. Daraus folgt, daß die Suche nach Instanzen für die Prädikatvariablen auf wenige Topologieknoten beschränkt werden kann. Eine solche Topologie berechnet in MOBAL die Systemkomponente PST ([Klingspor, 1991],[Morik *et al.*, 1993, S.149-168]).

Sortentaxonomie
Es ist effektiv, nur sortenverträgliche Hypothesen zu testen. An einem abstrakten Beispiel soll der Sachverhalt deutlich gemachen werden.
Gegen sei folgendes Regelschema:

```
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 nur Prädikate ausgewählt werden, die die Sorte von `X` als als Sorte ihres ersten Arguments haben. Die Sortenbeziehungen werden in MOBAL durch die Systemkomponente STT berechnet ([Morik *et al.*, 1993, S.109-148])

Nachdem nun RDT beschrieben wurde, wird in der Abbildung 1 die Vorgehensweise noch einmal anschaulich skizziert.

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 der Datenbank in geeigneter Form als Prädikate dargestellt werden. Diese Transformation muß eine injektive Abbildung sein, rdt(Q):-

- Konstruiere partielle Ordnung über die Metaprädikate (MP) für **Q**.
- Gehe TOP–DOWN durch die Ordnung der MP.
  - Instanziiere ein weiteres Prädikat `P` in MP, das folgendem genügt:
    1. Stelligkeitskompatibel
    2. Prädikatentopologiekompatibel
    3. Sortenkompatibel
  - Redundanztest θ–Subsumtion mit bisher akzeptierten oder zu speziellen Hypothesen.
  - Berechnung der Faktoren für das Akzeptanzkriterium.
  - Auswertung des Akzeptanzkriteriums.

Abbildung 1: Vorgehensweise in RDT

damit man bei den Lernläufen wieder auf die DB zugreifen kann.

Problemstellung:
Gegeben sei eine Menge von Relationsschemata R und r∈R über die Attribute $a_1, ..., a_n$.
Finde eine geeignete Prädikatendarstellung für r.

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] zusammengefaßt. So konnte man schnell und einfach existierende attribut–orientierte Lernverfahren einsetzen.

Eine solche Vorgehensweise ist für ein logikbasiertes Verfahren wie RDT sicherlich nicht geeignet, da man die einzelnen Relationen der Datenbanken verliert.

Zur Umsetzung der Relationen in Prädikate gibt es zahlreiche Möglichkeiten, von denen die folgenden drei in Betracht gezogen wurden.

Repräsentation I:

Man übernimmt den Tabellennamen als Prädikatnamen und die Attribute der Relation als Prädikatattribute.

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

Repräsentation II:

Eine andere Möglichkeit ist die Zerlegung der Relation in $n + 1$ Prädikate der Form:

Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow \forall(a_x(R, a_j, ..., a_l, a_x))$,
$R(a_j, ..., a_l)$,
$a_j, ..., a_l$ sind Primärschlüsselattribute der Relation R, und x läuft im Intervall $[1, ..., n]$ ohne $j \ldots l$.

Repräsentation III:

Die dritte Form ist ähnlich der Repräsentation II. Auch hier erhält man $n+1$ Prädikate, nur mit dem Unterschied, daß die Relationsbezeichnung (Tabellenname) mit in den Prädikatnamen gezogen wird.

Abbildungsvorschrift: R über $a_1, ..., a_n \rightarrow \forall R\_a_x(a_j, ..., a_l, a_x)$,
$R(a_j, ..., a_l)$,
$a_j, ..., a_l$ sind Primärschlüsselattribute der Relation R, und x läuft im Intervall $[1, ..., n]$ ohne $j \ldots l$.

4.2 Diskussion der Repräsentationen

Der Vorteil der Repräsentation I liegt in der sehr einfachen Überführungsvorschrift. Dem stehen allerdings zwei gravierende Nachteile gegenüber. Der erste Punkt ist, daß nicht mehr auf einzelne Attribute der Relation zugegriffen und somit nur die ganze Relation als Zielbegriff verwendet werden kann. Es dürfte aber gerade von Interesse sein, auch über einzelne Attribute zu lernen. Der zweite Nachteil ergibt sich aus der angestrebten Anwendung.

In dem logikorientierten Lernverfahren RDT wird die Beschreibung für einen Begriff gesucht. Als Beschreibungssprache dienen dabei die dem System bekannten Prädikate. Ist nun ein Attribut „frei" wählbar bei der Beschreibung des zu lernenden Begriffs, so erhöht sich das Kriterium *total* von RDT um den Faktor der Anzahl der Sortenelemente. Diese hätte zur Folge, daß das Kriterium *pred* als Akzeptanzkriterium von RDT nicht mehr handhabbar wäre. Man verliert dadurch ein Kriterium.

An einem Beispiel soll die Problematik deutlich gemacht werden:

Gegeben ist ein Metaprädikat der Form

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

und zwei Datenbankrelationen r1 (zweistellig) und r2 (sechsstellig). Sei nun $m1$ vollinstanziiert:

$$r1(X1,X3) \; \& \; r1(Y1,Y3) \rightarrow r2(X1,Y1,X2,Y2,X3,Y3)$$

Die Variablen X2 und Y2 der Konklusion sind nicht in der Prämisse gebunden. Das Produkt aus |X2 |, |Y2 | und der Anzahl der möglichen Belegungen von r2 ergibt die maximale Anzahl der möglichen Belegungen für r2.

Repräsentation	freie Variablen	# Prädikate pro Relation	doppelte Prädikate
I	ja	1	nein
II	nein	$n+1$	ja
III	nein	$n+1$	nein

Tabelle 1: Vergleich der Repräsentationen I, II und III

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

In den Repräsentationen II und III wird diese Problematik im wesentlichen vermieden. Allerdings kann im schlechtesten derselbe Fall eintreten wie in der Repräsentation I. Dies allerdings nur, wenn in der DB keine oder alle Attribute als Primärschlüssel deklariert wurden.

Der Nachteil der Repräsentation II ist die Möglichkeit, daß verschiedene Relationen auf dasselbe Prädikat abgebildet werden können. Zum Beispiel könnte es neben einer Relation Personal noch eine Relation Kunden geben, die zbeide die Attribute Name und Vorname als Primärschlüssel haben. In Repräsentation II erhält man somit möglicherweise zwei Prädikate ort/4.

Ein gemeinsamer Schwachpunkt der Repräsentationen II und III ist die Anzahl der entstehenden Prädikate, da die Anzahl der Prädikate direkt von der Attributanzahl der Relation abhängt.

Die Repräsentation III ist die geeigneste Darstellung zur Anwendung von RDT. Die Repräsentation III stellt im allgemeinen Fall sicher, daß im Gegensatz zur Repräsentation I keine freien Variablen in der Konklusion einer Regel auftreten können und Prädikate nicht doppelt dargestellt werden. Jeweils einen dieser Punkte erfüllen die anderen Repräsentationen nicht. In Tabelle 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 soll die Problematik veranschaulicht werden.

Gegeben sind folgende zwei Metaprädikate[3]:

$$\mathrm{m1(P1,P2,C):P1(}\underline{\mathrm{S,T,U,V}}\mathrm{)}\ \&\ \mathrm{P2(}\underline{\mathrm{S,T,V,X}}\mathrm{)} \rightarrow \mathrm{C(}\underline{\mathrm{S,T}}\mathrm{,U)}$$

$$\mathrm{m2(P1,P2,P3,C):P1(}\underline{\mathrm{S,T,U,V}}\mathrm{)}\ \&\ \mathrm{P2(}\underline{\mathrm{S,T,V,X}}\mathrm{)}\ \&\ \mathrm{P2(}\underline{\mathrm{S,T,X,Z}}\mathrm{)}$$
$$\rightarrow \mathrm{C(}\underline{\mathrm{S,T}}\mathrm{,U)}$$

RDT/DB würde m2 nicht als mögliche Hypothese betrachten, wenn m1 akzeptiert wurde, da m1 bzgl. der θ–Subsumption genereller ist als m2. Damit dieser Fall, nicht auftritt muß das Ende der Verkettung im Konklusionsbegriff gebunden sein. Eine Lösung ist, die

[3] Die Schlüsselattribute sind unterstrichen.

Repräsentationen I und III zu kombinieren. Das Problem der ungebundenen Variablen tritt hierbei nicht mehr auf, da die Schlüsselattribute des Konklusionsbegriffes gebunden sind.

Mit dieser gewählten Repräsentation ist der in der Datenbank dargestellte Sachbereich überrepräsentiert, was sich aber aufgrund der automatischen Umrepräsentation nicht vermeiden läßt. Der Benutzer hat allerdings auch die Möglichkeit, nicht gewünschte Prädikate wieder zu löschen.

5 Umsetzung des Akzeptanzkriteriums in SQL – Anfragen

Um das Verfahren RDT auf relationale Datenbanken anwenden zu können, muß man geeignete SQL–Anfragen bestimmen, die die einzelnen Faktoren des Akzeptanzkriteriums berechnen.

Im folgenden sei H definiert wie in Abschnitt 3. Die Prädikate der Hypothese H sind von der Form $R\_a_x(PS, a_x)$, bzw. $R(PS)$, 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 Prämisen P in SQL–Bedingungen überführt.

Die Bestandteile des Akzeptanzkriteriums können wie folgt angesetzt werden:

- $\mid pos(H) \mid \Rightarrow$
  select count($ps(q)$) 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.

  $\mid neg(H) \mid \Rightarrow$
  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.

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

- $\mid total(H) \mid \Rightarrow$
  select count(*) from $tabelle(P)$
  where $(v(P(c_1, \ldots, c_m))$

- $\mid concl(H) \mid \Rightarrow$
  select count($ps(q)$) from $tabelle(q)$;
- $\mid uncover(H) \mid = \mid concl(H) \mid - \mid pos(H) \mid$

6 Das Verfahren RDT/DB

6.1 Einschränkungen des Hypothesenraums

RDT/DB gleicht in der Grundstruktur RDT. Wie bei RDT wird in RDT/DB der Hypothesenraum durch die vorgegebenen Regelmodelle definiert und nach der θ-Subsumtion geordnet.

Unterschiede zu RDT gibt es bei der Instanziierung der Regelschemata. Wie in RDT wird in RDT/DB weiterhin die *relation chain* und die Prädikatentopologie für die Belegung der Prädikatvariablen benutzt. Die Sortenbeziehung ist in RDT/DB durch die schwächere Einschränkung der Datentypkompatibilität ersetzt worden. Weiterhin wird eine Art der redundaten Instanziierung in RDT/DB vermieden, die sich aus dem gewählten Repräsentationsformalismus, Abschnitt 4, ergibt.

6.1.1 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[4]. 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.

6.1.2 Datentypkompatibilität

Eine weitere Möglichkeit der Hypothesenraumeinschränkung neben den Regelmodellen und der Prädikatentopologie 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 3 (Datentypkompatibel)
Sei $m(P_1, \ldots, P_n, C)$ ein teilinstanziiertes Regelmodell, $P(A_1, \ldots, A_k)$ ein zu instanziie-

[4] Hierbei wird von ungeordnete Mengen ausgegangen.

rendes 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:

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

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

6.1.3 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 7.1) 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 können 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 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 wird.

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 4 (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, daß 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.

6.2 θ–Subsumtionstests in RDT/DB

In den ersten Lernläufen[5] 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.

Garey und Johnson haben 1979 schon gezeigt, daß diese Problematik NP–vollständig ist [Garey und Johnson, 1979]. In einem neueren Beweis zeigen Kietz und Lübbe, daß der θ–Subsumtionstest schon bei verbundenen Hornklauseln NP–hart ist. Aus diesem Grund wird vermutet, daß es keine effizienten Algorithmen für dieses Problem gibt [Kietz und Lübbe, 1994].

[5] Siehe Testergebnisse in Kapitel 7.

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.

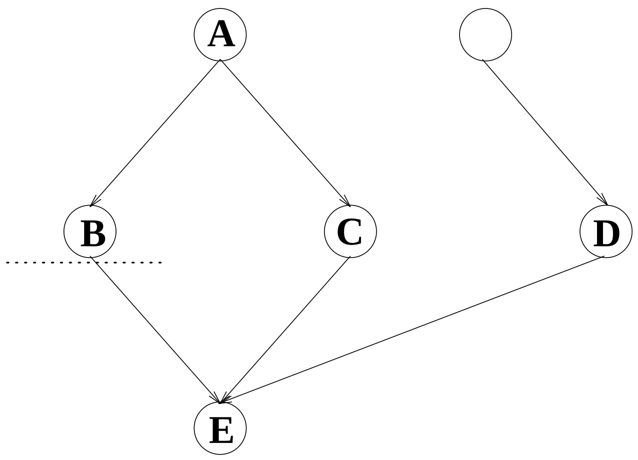

Abbildung 2: Instanziierungsgraph

An einem Ausschnitt aus einem allgemeinen Instanziierungsgraph (siehe Abbildung 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 mit der aktuellen Hypothese im θ–Subsumtionstest verglichen werden, wird B jetzt nicht in die Menge der zu vergleichenden Hypothesen 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 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 zu spezielle Hypothesen 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.

6.3 Negative Beispiele in RDT/DB

In relationalen Datenbanken sind keine negativen Beispiele gegeben, diese können aber für das Lernen von großer Bedeutung sein, da ein negatives Beispiel oft stärker abgrenzt. Durch negative Beispiele vermeidet man es zu allgemeine Regeln zu lernen. An dieser Stelle wird ausgenutzt, daß RDT/DB als externes Tool in das System MOBAL mit eingebunden ist. Der Benutzer kann so über das System MOBAL negative Beispiele, die beim Lernen über die Datenbank berücksichtigt werden (siehe Abschnitt 5), eingeben.

6.4 Algorithmus

Bevor im nächsten Abschnitt die ersten experimentellen Tests mit RDT/DB vorgestellt werden, wird im folgenden die Vorgehensweise von RDT/DB skizziert.

rdt_db(Q):-

- Konstruiere partielle Ordnung über die Metaprädikate (MP) für **Q**.
- Gehe TOP–DOWN durch die Ordnung der MP.
  - Instanziiere ein weiteres Prädikat `P` in MP das folgendem genügt:
    1. Stelligkeitskompatibel
    2. Prädikatentopologiekompatibel
    3. Datentypkompatibel
    4. Test auf redundant instanziierte Prädikate
  - Redundanztest θ–Subsumtion mit bisher akzeptierten oder zu speziellen Hypothesen.
  - Berechnung der Faktoren für das Akzeptanzkriterium.
  - Auswertung des Akzeptanzkriteriums.

7 Lernläufe mit RDT/DB

7.1 Tests im Lernszenario KRK

7.1.1 Sachbereich

Das KRK–Schachendspiel 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.

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 Vergleichstests von LINUS/FOIL [Lavrač und Džeroski, 1992], und RDT/FOIL [Lindner und Robers, 1994] verwendet. Aus der Sicht des Knowledge Discovery ist diese Anwendung allerdings nicht akzeptabel. Hier wird besonderer Wert auf die „reale“Anwendung gelegt. Um diesem Punkt gerecht zu werden wurde ein Testszenario aus dem Bereich der Roboternavigation (siehe Abschnitt 7.3) für weitere experimentelle Tests mit RDT/DB ausgewählt.

7.1.2 Testaufbau und Lernergebnisse im KRK–Sachbereich

Im KRK–Sachbereich wurden zwei Testreihen durchgeführt. In der ersten Testreihe soll der direkte Vergleich zwischen RDT und RDT/DB im Vordergrund stehen und die Anwendbarkeit von RDT/DB auf größere Daten gezeigt 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.

Testaufbau und Ergebnisse KRK-I

Die Relationen entsprechen der in der Beispieldatenbank in Abschnitt 7.1 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).
```

Die Ergebnisse der Lernläufe sind in der Tabelle 2 zusammengefaßt.

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

Lernverfahren	Beispielanzahl	Regeln	Lernzeit
RDT	335	> 600	> 20 Std.
RDT/DB*[6]	335	120	2 Std. 28 Min.
RDT/DB	3504	191	12 Std. 32 Min.
RDT/DB*	3504	191	8 Std. 5 Min.

Tabelle 2: 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.

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.

Testaufbau und Lernergebnisse KRK-II In diesem Beispiel sind die einzelnen Schachfiguren in eigenen Tabellen in der Datenbank abgelegt (siehe Tabellen 3 bis 6). Weiterhin ist der Zielbegriff 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, daß solche Datenmengen nicht in der MOBAL–Wissensbasis verarbeitet werden können.

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

Tabelle 3: white_king

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

Tabelle 4: black_king

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

Tabelle 5: white_rook

ID
267434
...

Tabelle 6: illegal

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) $| pos(H) | > 1500$
  b) $| pos(H) | > 700$
- 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 $| pos(H) | > 700$ 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.

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

Lernergebnisse

Lernlauf Ia): In der Abbildung 3 sind die von RDT/DB gelernten Regeln aus dem Lernlauf I dargestellt. Der Benutzer muß nun, wie im Lernszenario beschrieben, entscheiden, welche Regeln in der Wissensbasis von MOBAL bleiben sollen.

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. Betrachtet werden sollen folgende zwei Regeln:

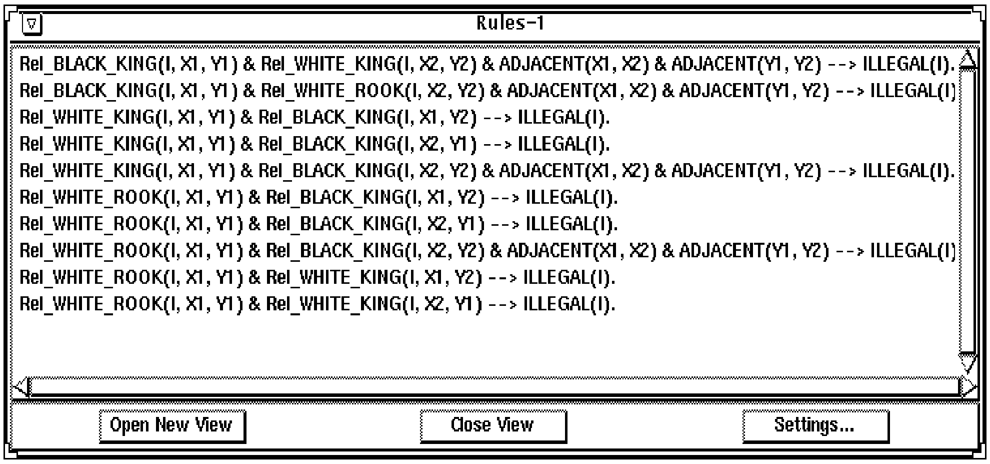

Abbildung 3: Gelernte Regeln aus dem Lernlauf I

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: Lernergebnisse zu illegal aus KRK II

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, daß `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 Tabelle 7 sind die Lernzeiten der Testläufe zusammengefaßt.

Diskussion der Ergebnisse

Zur Analyse der Lernergebnisse werden zwei Bewertungskriterien, Vollständigkeit und Korrektheit, eingeführt.

- **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 experimentellen Vergleich von RDT und FOIL über KRK–Sachbereich verwendet. Hier wurde auch dieselbe Repräsentation verwendet. In der Tabelle 8 sind die Werte für die Vollständikeit und Korrektheit der Lernläufe zusammengefaßt.

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

Tabelle 8: Vollständigkeit und Korrektheit

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 wurden, sehr nahe kommen. Die Tests mit FOIL und RDT wurden mit einer Beispielmenge von bis zu 500 Stück durchgeführt. Diese Tests wurden von Ursula Robers und Guido Lindner beschrieben [Lindner und Robers, 1994]. In diesem Test[7] 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

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

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, daß 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).`

7.2 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 derselben Rechnerumgebung auf einer SPARC STATION ELC gestartet wurde.

7.3 Tests im Lernszenario Roboternavigation

7.3.1 Sachbereich

Das Lernszenario ist aus dem aktuellen Forschungsprojekt B-Learn II[8] entnommen. In dem Lernszenario geht es um die Bewegung eines Roboters in einem einfachen Raum (siehe Bild 4), in dem sich dieser Roboter durch die Messungen seiner 24 Sensoren orientieren soll. Hierzu werden die Messungen auf verschiedenen Ebenen zu Definitionen von Begriffen abstrahiert. Eine genaue Beschreibumg der verschiedenen Abstraktionsebenen geben Morik und Rieger in [Morik und Rieger, 1993]. Die Messungen der einzelnen Sensoren werden in Intervalle unterteilt, die einen linearen Messungsverlauf beschreiben. Diese Intervalle werden *basic features* genannt. Für den Trace 24 und den Sensor 5 erhält man folgende Sequenz von *basic features* (siehe [Klingspor, 1994]).

- `no_measurement(t24,0r,s5,1,14,_)`
- `decreasing    (t24,0r,s5,14,15,-22)`

[8] Das Projekt B-Learn II (P7274) wird von der EG und dem Forschungsministerium von NRW unterstützt

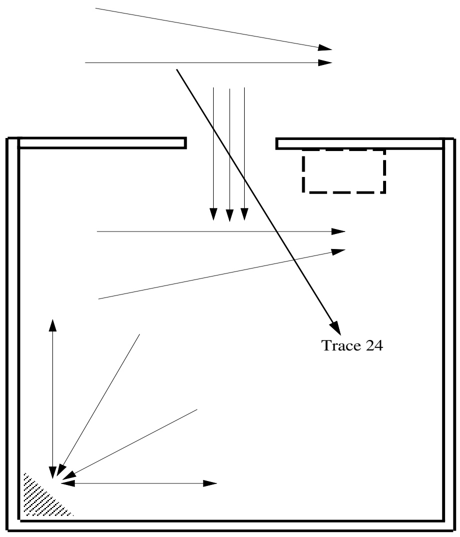

Abbildung 4: Roboterbewegung in einem Raum

- `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[9], die relative Orientierung des Sensors, die Sensorbezeichnung, Start- und Endzeit der Messung und den Gradienten. 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 beschreibt.

Lernziel ist, Abfolgen von *basic features* zu lernen, die ein *sensor feature* beschreiben, d.h ein *sensor features* beschreiben das Verhalten eines Sensors vom Zeitpunkt t1 bis tn. Insgesamt gibt es 4 *sensor features*. Die Argumente der *sensor features* sind die Tracenummer, die Sensorbezeichnung, die Start- und Endzeit der Abfolge und die Bewegungsrichtung.

7.4 Testaufbau

Neben den *basic features* und den *sensor features* wurden fünf Metaprädikate für die Lernläufe verwendet, von denen hier drei exemplarisch angeben sind:

```
m1(P1,P2,M,C)      :P1(T,S,X1,X2) & P2(T,S,X2,X3) & M(O)
                    --> C(T,S,X1,X3,O)
```

[9] Schlüsselattribute sind unterstrichen.

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

Im Lernszenario Roboternavigation war das Ziel eine „reale" Anwendung durchzuführen. Eine solche Anwendung, ausgehend von einer nicht überschaubaren und interpretierbaren Menge an Meßdaten, ist besonders für das Knowledge Discovery interessant.

7.5 Lernergebnisse

In der Tabelle 9 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.

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

Tabelle 9: Lernergebnisse zum Sachbereich Roboternavigation

Die Lernzeiten, die die Verfahren jeweils für die einzelnen Lernziele benötigten, sind der Tabelle 10 zu entnehmen.

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 10: Lernzeiten im Sachbereich Roboternavigation

7.6 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.

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 6.1.2) und einen Test auf redundante Prädikate (siehe Abschnitt 6.1.3) 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 2.1.

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 Mehraufwand bedeuteten, der sich bei größeren Faktenbasen relativiert. In den Tests hat sich gezeigt, daß die θ–Subsumtionstests für die zu speziellen Hypothesen zeitaufwendiger ist als ein direkter Test gegen die Beispieldaten. 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 beschäftigt. 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 andere 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 großem Interesse.

Verbesserungen
In RDT/DB wurde auf die Ausnutzung der Sortenbeziehungen verzichtet (vgl. Abschnitt 6.1.1). Diese kann bestimmt in vielen Fällen sehr effektiv 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.

9 Danksagung

Mein besonderer Dank gilt Joachim Hertzberg und Volker Klingspor, die mir mit ihren Ratschlägen, Kommentaren und Korrekturlesen immer eine gute Quelle für Verbesserungen waren.

Literatur

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

[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.

[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.

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

[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 Machine 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.

[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.

[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.

[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.

[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.

[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.

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