Wissensentdeckung in relationalen Datenbanken: Eine Herausforderung für das maschinelle Lernen

Peter Brockhausen und Katharina Morik

Lehrstuhl Informatik VIII, Universität Dortmund

Zusammenfassung. Die bisherigen Ansätze, Wissensentdeckung in sehr großen Datenbanken möglich zu machen, konzentrieren sich auf zwei entgegengesetzte Standpunkte. Die Extreme sind hier die Wahl einfacher Hypothesensprachen oder die geschickte Wahl einer kleinen Datenmenge. Hinzu kommen oft noch weitere Datenreduktionsmethoden. Der von uns verfolgte Ansatz einer kooperativen, balancierten Arbeitsteilung zwischen zwei spezialisierten Systemen für Hypothesengenerierung und Hypothesentest zeigt, daß es sehr wohl möglich ist, nicht nur über allen Daten zu lernen und prädikatenlogische Hypothesen zu entdecken, sondern auch die gegenüber dem Begriffslernen schwierigere Regellernaufgabe zu lösen.

Schlüsselwörter. Induktive logische Programmierung, Kopplung ILP mit RDBMS, Begriffslerner, Regellerner

1 Einleitung

Die Wissensentdeckung in Datenbanken (*KDD*) als ein relativ neues Anwendungsgebiet stellt hohe Anforderungen an maschinelle Lernverfahren, da sie sowohl eine große Effizienz der eingesetzten Lernverfahren verlangt, als auch die erzielten Ergebnisse in besonderem Maße zuverlässig und verständlich sein müssen. Lernen ist schwierig, da die zu lösende Lernaufgabe die Entdeckung aller interessanten, gültigen und nicht redundanten Regeln fordert. Eine zusätzliche Schwierigkeit bei diesem Anwendungsszenario *KDD* kommt durch die Anforderung an die Lernverfahren zustande, sehr große Datenbestände analysieren zu müssen.

Nun ist Datenanalyse für sich genommen kein neues Feld. Durch die spezifischen Anforderungen, die das *KDD* als Anwendungsfeld stellt, ergeben sich aber neue Herausforderungen. Die zu analysierenden Daten entstammen nicht mehr einer strukturierten Versuchsplanung, sondern sie werden für andere Zwecke als die Wissensentdeckung gesammelt oder fallen einfach an. Die Anzahl der Daten oder Beobachtungen ist dadurch häufig immens. Darüberhinaus enthalten die Daten eine Vielzahl unterschiedlicher Merkmale, ohne daß im Detail klar ist, warum welche Merkmale erfaßt wurden. Auch die Anforderungen, die an die Analysten und ihre Werkzeuge gestellt werden, verschieben sich. Im *KDD* steht nicht mehr die Überprüfung einer genau spezifizierten Hypothese im Vordergrund, sondern es werden *möglichst umfassende Antworten auf recht vage formulierte Fragen verlangt.*

Wenn wir uns die klassischen maschinellen Lernverfahren näher ansehen, die zur Zeit im *KDD* zum Einsatz kommen, so ist festzustellen, daß deren bisheriger Einsatzzweck nicht in der Analyse großer Datenmengen bestand. Vielmehr wurden sie zur Unterstützung des Wissenserwerbs oder der Wissenswartung eingesetzt, um nur zwei Gebiete beispielhaft zu nennen. Auch bestanden die benutzten Datenmengen aus wenigen Beobachtungen mit wenigen Merkmalen oder Relationen. Darüberhinaus verlangte jedes Lernverfahren die Daten in einer ihm eigenen Repräsentationssprache. Als Lernaufgabe gab es eine klar definierte Zielgröße. Dieses war häufig ein Begriff, dessen Charakterisierung gelernt werden sollte.

Unter den verschiedenen Ansätzen, die zur Zeit im Data Mining verfolgt werden, lassen sich drei Hauptströmungen identifizieren: „Gehe weg von der Datenbank“; „Gehe weg von den Datenmassen“; und „Vereinfache die Lernaufgabe“. Diese drei Ansätze sind nicht als sich ausschließende Alternativen zu verstehen, sondern sie kommen in den unterschiedlichsten Mischformen vor.

Der erste Ansatz ist gekennzeichnet durch eine Merkmals– bzw. Attributselektion in der Datenbank. Die selektierten Attribute, und damit die Daten, werden exportiert und in die Repräsentation des benutzten Lernverfahrens überführt. Unter den Lernverfahren finden sich die bekannten Verfahren unverändert wieder.

Im Rahmen der zweiten Alternative ist vor allem die Methode zu nennen, eine Stichprobe auszuwählen (Sampling), auf der dann ein Klassifikator gelernt wird. Oder es kommen Windowing–Techniken zum Einsatz, wie sie z. B. C4.5 [24] bietet. Breiman beschreibt in [4] einen Ansatz, der als Kombination dieser beiden Vorgehensweisen aufgefaßt werden kann. Die meisten Algorithmen, die bisher im *KDD* eingesetzt werden und der induktiven logischen Programmierung (*ILP*) zuzuordnen sind, setzen eine Komplexitätsreduktion durch Sampling voraus.

Der dritte und zur Zeit wohl prominenteste Ansatz im *KDD* besteht in der radikalen Vereinfachung der Lernaufgabe. Hiermit sind fast zwangsläufig sehr performante Algorithmen verbunden. Da sind zum einen die Lernverfahren, die Assoziationsregeln lernen, z. B. Apriori oder AprioriTID [1]. Andere schnelle Algorithmen dieser Gruppe fallen unter den Begriff der *Attribut–orientierten Induktion* [5]. Es ist leicht zu zeigen, daß komplexere Abhängigkeiten zwischen Attributen von diesen einfachen Algorithmen nicht gelernt werden können [6].

Diesen drei im Wettstreit miteinander stehenden Alternativen scheint gemein zu sein, daß sie *KDD* durch ihre jeweilige Reduktion erst möglich machen. Ist dieses zwangsläufig so? Müssen wir uns immer für eine dieser drei Möglichkeiten entscheiden und damit ebenso zwangsläufig eine dieser drei Dimensionen derart beschränken, um überhaupt zu Lernergebnissen kommen zu können? Oder ist es möglich, sowohl mit den Daten in der Datenbank direkt umzugehen, als auch aus allen Tupeln zu lernen, und darüberhinaus auch noch die schwierigere Lernaufgabe im *ILP*, das Regellernen, zu lösen?

Diese Herausforderung, die die Wissensentdeckung in Datenbanken für das maschinelle Lernen darstellt, manifestiert sich für uns in vier Punkten, denen wir hier nachgehen. Wir sehen die Repräsentation der Daten in ihrer Beschreibungssprache als fest vorgegeben an. Wir wollen zeigen, daß wir diese Darstellung, die für andere Zwecke als das Lernen entworfen wurde, handhaben können. Dieses heißt konkret: Wir belassen die Daten so in der relationalen Datenbank ORACLE V7, wie wir sie vorfinden. Das heißt auch, daß wir die Anfragesprache *SQL* für den Hypothesentest verwenden müssen.

Schranken, die traditionellen Lernverfahren durch den verfügbaren Hauptspeicher auferlegt werden, durchbrechen wir, indem wir die ganze Datenhaltung dem Datenbanksystem aufbürden. Für Datenbanken sind mehrere hunderttausend Tupel in einer Tabelle bei *normalen* Anfragen kein Problem. Hier sehen wir die zweite Herausforderung darin, zu zeigen, daß dieses auch für Anfragen von Lernverfahren gelten kann.

Weiterhin enthalten Datenbanken oft sehr viele Attribute in verschiedenen Tabellen. Lernverfahren, die eine eingeschränkte Prädikatenlogik als Hypothesensprache verwenden, bieten eine elegante Möglichkeit, diese Relationen und Beziehungen zu modellieren.

Das vierte Problem, dem wir uns stellen wollen, ist die Abkehr von einem fest vorgegebenen Lernziel. Wir wollen weder gezielt eine Hypothese überprüfen, noch einen oder mehrere Zielbegriffe für einen Begriffslerner vorgeben. Vielmehr wird es zu einem Bestandteil des Lernverfahrens, diejenigen Prädikate zu finden, über die es Regeln lernen kann. Dieses erreichen wir durch die Verwendung eines Regellernverfahrens. Dieses Lernverfahren RDT/DB verwendet eine deklarative Beschreibung der Hypothesensprache (deklarativer Bias). Dieses ist für den Benutzer von Vorteil, da er so die Möglichkeit erhält, seine Vorstellungen darüber, wie interessante Hypothesen aussehen könnten, zu formulieren. Die Kombination der beiden Aspekte deklarativer Bias und Regellernverfahren bildet eine solide Grundlage, um dem allgemeinen Ziel im *KDD* näher zu kommen, etwas interessantes zu entdecken, von dem man nur eine grobe Vorstellung hat.

In Abschnitt 2 werden wir anhand eines Beispiels zeigen, daß die Regellernaufgabe schwieriger als die Begriffslernaufgabe ist und daß sie an ihr Lernergebnis stärkere Forderungen stellt, als üblicherweise im *KDD* von Lernaufgaben und deren Ergebnissen verlangt werden. In Abschnitt 3 werden wir das Regellernverfahren RDT/DB darstellen, anschließend erläutern wir die Experimente, die wir mit RDT/DB durchgeführt haben (Abschnitt 4). Zum Schluß diskutieren wir unsere Arbeit und fassen die Ergebnisse zusammen.

2 Lernaufgaben im KDD

Eine Dimension, nach der sich maschinelle Lernverfahren einteilen lassen, ist die Art der Lernaufgabe, zu deren Lösung sie eingesetzt werden. Ohne auf die kleineren Unterschiede einzugehen, die bei näherer Betrachtung rein syntaktischer Natur sind, wird bei den Lernaufgaben unterschieden zwischen dem

- Lernen einer (Menge von) Begriffsbeschreibung(en) bzw. Klassifikatoren und dem
- Lernen von Regeln.

Das zweite Szenario wurde von Helft im Rahmen des *ILP* eingeführt [8]. Für Sprachen $\mathcal{L}$, die Teilmengen der Prädikatenlogik erster Stufe sind, definiert Kietz diese beiden Lernaufgaben folgendermaßen [10].

Definition 1 (Regellernen) *Seien eine Menge E von Beobachtungen in einer Sprache $\mathcal{L}_\mathcal{E}$, Hintergrundwissen B in einer Sprache $\mathcal{L}_\mathcal{B}$ und eine Hypothesensprache $\mathcal{L}_\mathcal{H}$ gegeben. Die Aufgabe beim Lernen von Regelhaftigkeiten besteht im Finden einer Menge H von Hypothesen, $H \subset \mathcal{L}_\mathcal{H}$, für die gilt:*

1. *H ist in allen minimalen Modellen[1] von B und E wahr, $\mathcal{M}^+(B \cup E) \subseteq \mathcal{M}(H)$ (Gültigkeit),*
2. *alle Hypothesen enthalten neue Informationen über die Beobachtungen, $\forall h \in H : \exists e \in E : B,E \backslash \{e\} \not\models e$ und $B,E\backslash\{e\},h \models e$ (Notwendigkeit),*
3. *alle in B und E wahren Hypothesen folgen aus H, $\forall h \in \mathcal{L}_\mathcal{H}$ die gültig und notwendig sind, gilt $: H \models h$ (Vollständigkeit) und*
4. *H ist minimal, d.h. es existiert keine gültige und vollständige Menge von Hypothesen G mit $G \subset H$, (Minimalität).*

Definition 2 (Begriffslernen) *Seien Hintergrundwissen B in einer Sprache $\mathcal{L}_\mathcal{B}$, positive und negative Beispiele $E = E^+ \cup E^-$ in einer Sprache $\mathcal{L}_\mathcal{E}$ und eine Hypothesensprache $\mathcal{L}_\mathcal{H}$ gegeben. Ferner gelte, daß sich kein Beispiel e aus dem Hintergrundwissen B folgern läßt, ($\forall e \in E : \; B \not\models e$), und daß die Beispiele konsistent mit dem Hintergrundwissen sind, ($B,E \not\models \Box$). Die Lernaufgabe für einen Begriffslerner besteht darin, eine Hypothese $H \subset \mathcal{L}_\mathcal{H}$ zu finden, für die gilt:*

1. *Die Hypothese ist konsistent mit dem Hintergrundwissen und allen Beispielen E, $B,H,E \not\models \Box$, (Konsistenz)*
2. *die positiven Beispiele E^+ folgen aus dem Hintergrundwissen und der Hypothese, $B,H \models E^+$ (Vollständigkeit) und*

[1] Minimale Modelle $\mathcal{M}$ kennzeichnen wir mit einem $^+$.

Tabelle 1: Beispieldatenbank über mögliche Kunden und ihre Ehepartner

Möglicher Kunde		
Person	Einkommen	Kunde
Ann Smith	10.000	+
Joan Gray	1.000.000	+
Mary Blyte	20.000	-
Jane Brown	20.000	+
Bob Smith	100.000	+
Tom Blyte	10.000	-
Jack Brown	200.000	+

verheiratet_mit	
Ehemann	Ehefrau
Bob Smith	Ann Smith
Tom Blyte	Mary Blyte
Jack Brown	Jane Brown

3. *die negativen Beispiele E^- folgen nicht aus dem Hintergrundwissen und der Hypothese, $B,H \not\models E^-$ (Korrektheit).*

Ein Beispiel soll die Unterschiede zwischen diesen beiden Definitionen verdeutlichen. Sei die relationale Datenbank aus Tabelle 1 gegeben. Ferner wenden wir einen Begriffslerner und einen Regellerner auf diese Datenbank an. Dabei laute der Zielbegriff für den Begriffslerner *kunde*. Dann finden beide Lernverfahren folgende drei Regeln:

(i) verheiratet_mit(Person,Ehepartner) & kunde(Person)
$\rightarrow$ kunde(Ehepartner)
(ii) verheiratet_mit(Person,Ehepartner) & einkommen(Person,$\geq$100.000)
$\rightarrow$ kunde(Ehepartner)
(iii) einkommen(Person,$\geq$100.000) $\rightarrow$ kunde(Person)

Unser Ziel ist es zu zeigen, daß leichte Variationen der Daten zu unterschiedlichen Mengen von Regeln führen werden, die für diese beiden Typen von Lernverfahren dann noch lernbar sind. Zur Variation der Daten nehmen wir lediglich an, daß einzelne Werte unbekannt sind; ein Zustand, der in realen Datenbanken sehr häufig anzutreffen ist.

Beispiel 1 *Es sei unbekannt, ob Jane Brown eine Kundin ist.*

Für den Regellerner gilt nun, daß Jane Brown nicht mehr in allen minimalen Modellen Kundin ist, folglich können (i) und (ii) nicht mehr gelernt werden. Nur Regel (iii) bleibt davon unberührt. Der Begriffslerner sagt Jane Brown weiterhin als Kundin vorher. Zur Erfüllung seiner Vollständigkeitsbedingung reicht es, wenn (i),(iii) oder (ii),(iii) gelernt werden. Wendet der Begriffslerner hingegen die *Closed–World Assumption* (*CWA*) an, so werden die Regeln (i) und (ii) falsch und damit zurückgewiesen. Dann reicht aber auch (iii) nicht aus, um alle positiven Beispiele abzudecken (Ann Smith). Hiermit schlägt das Begriffslernen fehl.

Tabelle 2: Zusammenfassung der lernbaren Regeln

	Regellerner	Begriffslerner	Begriffsl. mit CWA
Ist Jane ein Kundin?	(iii)	(i),(iii) oder (ii),(iii)	—
Wieviel verdient Jack?	(i),(ii),(iii)	(i),(iii)	(i),(iii)
Ist Jane Kundin und was verdient Jack?	(iii)	—	—

Beispiel 2 *Es sei unbekannt, wie hoch das Einkommen von Jack Brown ist.*

Im Falle des Regellerners werden alle drei Regeln (i) – (iii) gelernt. Jetzt nur mit dem Unterschied, daß Regel (ii) Jane Brown nicht mehr betrifft. Der Begriffslerner lernt die Regeln (i) und (iii). Damit werden alle positiven Beispiele für Kunden abgeleitet. Bei Verwendung der *CWA* ändert sich nichts.

Beispiel 3 *Jetzt seien sowohl der Status von Jane Brown als Kundin, als auch das Einkommen von Jack Brown unbekannt.*

Für den Regellerner gilt, daß Regel (iii) als einzige Regel in allen Modellen gültig bleibt. Im Falle des Begriffslerners gilt nun aber, daß mit keiner Regelmenge alle positiven Beispiele abgeleitet werden können. Damit schlägt das Begriffslernen komplett fehl. In Tabelle 2 haben wir die Ergebnisse der drei Beispiele zusammengefaßt. Es läßt sich festhalten, daß Regellernen dichter an den Daten ist, während Begriffslernen mehr Vorhersagekraft hat. Dies zeigt Beispiel 1.

Die Beispiele illustrieren, daß die Regellernaufgabe schwieriger ist als die Begriffslernaufgabe. So gibt es zum einen Lernaufgaben, die vom Regellernen noch gelöst werden können, vom Begriffslernen hingegen nicht (Beispiel 3). Auf der einen Seite kann das Begriffslernen zwar unter Zuhilfenahme der *CWA* die Gültigkeitsanforderung des Regellernens erfüllen, die Vollständigkeitsanforderung des Regellernens jedoch nicht. Wendet aber das Begriffslernverfahren die *CWA* an, so führt die Vollständigkeitsbedingung des Begriffslernens sehr schnell dazu, daß gar nichts mehr gelernt wird. Stephen Muggleton und Luc De Raedt haben für definite Hornklauseln gezeigt, daß die Begriffslernaufgabe bei Verwendung der *CWA* in der Regellernaufgabe enthalten ist [20]. Jörg–Uwe Kietz hat dieses Ergebnis dahingehend verallgemeinert, daß alle Begriffslernaufgaben durch einen Regellerner gelöst werden können, nicht aber umgekehrt [10]. Folglich ist Regellernen die schwierigere Aufgabe. Diese Schwierigkeit liegt in der Vollständigkeitsbedingung des Regellernens: „Finde alle gültigen und notwendigen Regeln!“

Die Regellernaufgabe in *ILP* geht auch über die Aufgabe der Wissensentdeckung hinaus, wie sie Heikki Mannila in [15] definiert hat:

Definition 3 (Wissensentdeckung) *Sei eine Datenbank E und eine Repräsentationssprache $\mathcal{L}_{\mathcal{H}}$ gegeben. Die Aufgabe der Wissensentdeckung be-*

steht darin, eine interessante und charakteristische Beschreibung H der Daten mit $H \in \mathcal{L}_{\mathcal{H}}$ zu finden, wobei die Interessantheit über ein Prädikat p bestimmt wird, sodaß gilt:

$$H(E,p) = \{h \in \mathcal{L}_{\mathcal{H}} \mid p(E, h(E)) \text{ } \textit{ist wahr}\}$$

Hintergrundwissen findet bei dieser Definition nur Berücksichtigung, wenn es in die Sprache $\mathcal{L}_{\mathcal{E}}$ der Datenbank transformiert und ebendort gespeichert wird. Dieser Ansatz findet sich auch oft im *ILP*, wenn Hintergrundwissen in Form von Grundfakten vorliegen muß. Der eigentliche Unterschied ist darin zu sehen, daß weder eine Notwendigkeit noch eine Minimalität von H gefordert wird. Die geforderte Interessantheit ist formal schwer zu fassen, und so existieren verschiedene Ansätze, diese über ein Akzeptanzkriterium zu messen [27].

3 KDD mit ILP und direktem Datenbankzugriff

Im *KDD* sind *ILP* Lernverfahren von besonderem Interesse, da sie die Entdeckung komplexer Regeln erlauben. Bis jetzt wurden sie aber nicht auf allgemein verwendete, kommerzielle Datenbanksysteme angewendet. Da eine der Zielsetzungen im *KDD* die Analyse der in Benutzung befindlichen Datenbanken ist, haben wir RDT zu RDT/DB hin weiterentwickelt, dem ersten *ILP*–Regellernverfahren, das direkt mit einer kommerziellen ORACLE V7 Datenbank interagiert.

3.1 RDT/DB

Zur Beschränkung des Hypothesenraumes benutzt RDT/DB die gleiche deklarative Beschreibung der Hypothesensprache wie RDT [12]. Diese Beschreibung gibt der Benutzer in Form von Regelschemata an, die auch synonym als Regelmodelle oder Metaprädikate bezeichnet werden. Ein Regelschema ist eine Regel mit Prädikatsvariablen anstelle von Prädikaten. Zusätzlich können in den Literalen bestimmte Positionen gekennzeichnet werden, an denen Konstante gelernt werden sollen. Durch die Angabe der Prädikatsvariablen zusammen mit den zu bestimmenden Konstanten im Kopf des Metaprädikates wird die Definition des Modells eindeutig.

$$mp1(C, P1, P2, Q): \quad P1(X,Y) \;\&\; P2(X,C) \quad \rightarrow Q(Y) \tag{1}$$

Zur Hypothesengenerierung instantiiert RDT/DB die Prädikatsvariablen und die Argumente, die zum Konstantenlernen markiert sind. Die Regelschemata werden gemäß einer erweiterten θ–Subsumtionsbeziehung nach ihrer Allgemeinheit angeordnet. RDT/DB durchsucht diese Halbordnung Top–

Down mit einer Breitensuche. Diese Suchstrategie garantiert in Zusammenarbeit mit der definierten Ordnung ein sicheres Pruning im Hypothesenraum! Folglich lernt RDT/DB die allgemeinsten Regeln.

Eine andere Art von Kontrollwissen, das der Benutzer angeben muß, ist das aus vier Grundbausteinen zusammengesetzte Akzeptanzkriterium: $pos(H)$, die Anzahl der Belegungen, für die Prämisse und Konklusion wahr sind; $neg(H)$, die Anzahl der Belegungen, für die Prämisse und Negation der Konklusion wahr sind; $concl(H)$, die Anzahl der Vorkommen der Konklusion und $negconcl(H)$, die Anzahl der Vorkommen der Negation der Konklusion. Der Benutzer kann dieses Akzeptanzkriterium dazu verwenden, um in unterschiedlich starkem Maße die Zuverlässigkeit von Regeln zu verlangen, oder andersherum ausgedrückt, um unterschiedlich viel Noise in den Regeln zu erlauben.

Für den Fall, daß es zwei Klassen gibt, z. B. fehlerhafte und nicht fehlerhafte Fahrzeuge, drückt das durch Bayes inspirierte Akzeptanzkriterium (2) aus, daß die a posteriori Wahrscheinlichkeit größer gleich der a priori Wahrscheinlichkeit sein soll:

$$\frac{pos(H)}{pos(H) + neg(H)} \geq \frac{concl(H)}{concl(H) + negconcl(H)} \tag{2}$$

Abhängig vom gewählten Akzeptanzkriterium wird für das Pruning ein zusätzliches Kriterium benötigt. Bei einfachen Kriterien reicht hingegen die „Negation“ des Akzeptanzkriteriums zum Pruning aus.

Für RDT/DB haben wir ein Interaktions- und Kommunikationssystem zwischen dem Lernverfahren und der Datenbank ORACLE V7 entwickelt[2]. Unter Zuhilfenahme des Datenlexikons der Datenbank werden die Datenbankrelationen und Attribute auf die Prädikate in RDT/DB's Hypothesensprache abgebildet. Dieser Vorgang geschieht auf Wunsch automatisch, der Benutzer kann ihn aber auch in allen Details beeinflussen. Hier ist zu beachten, daß wir in RDT/DB nur Prädikatsdeklarationen speichern, jedoch keinerlei Umrepräsentationen an den Daten vornehmen! Die Hypothesengenerierung übernimmt das Lernverfahren. Für den Hypothesentest wird die generierte Hypothese in *SQL*–Anfragen übersetzt und über eine Netzwerkverbindung an die Datenbank geschickt.

RDT/DB kann sowohl mit negativen Beispielen umgehen, andererseits aber auch nur aus positiven Beispielen lernen. Durch Deklarationen der Art *not-faulty(ID) = not(faulty(ID))* gibt der Benutzer dem System bekannt, welche Prädikate gegensätzliche Bedeutung haben.

3.2 Analyse des Hypothesenraumes

Die Größe des Hypothesenraumes von RDT/DB hängt nicht von der Anzahl der Tupel ab, sondern von der Anzahl der Regelschemata r; der Anzahl p der

[2]Eine erste, prototypische Implementierung wurde in [14] beschrieben.

Prädikate, die für Instantiierungen der Prädikatsvariablen zur Verfügung stehen; und der maximalen Anzahl k der Literale in einem Regelschema. Beim zusätzliches Lernen von Konstanten müssen alle Werte der zum Konstantenlernen markierten Argumente durchprobiert werden. Sei c die Anzahl der markierten Argumente und i die maximale Anzahl möglicher Werte für diese Argumente. Dann lautet die obere Schranke für den Hypothesenraum von RDT/DB:

$$r \cdot (p \cdot i^c)^k$$

Da k in der Regel eine kleine Zahl ist, damit die gelernten Regeln verständlich bleiben, ist dieses Polynom akzeptabel. Maßgeblich für die Größe des Hypothesenraumes ist folglich die durch den Benutzer vorgegebene Hypothesensprache, die für die Hypothesengenerierung Verwendung findet. Obwohl die tatsächliche Größe des Hypothesenraumes sehr stark mit der gewählten Hypothesensprache variiert, bleibt sie immer endlich. Wir können also die Komplexität in Form der $\mathcal{VC}$–Dimension [28] angeben als:

$$\mathcal{VC}dim(\mathcal{L_H}) \leq log_2\left(r \cdot (p \cdot i^c)^k\right)$$

Die konkreten Zahlen für diese Konstanten können unterschiedlich ausfallen, abhängig davon, wie der Benutzer die Abbildung von Relationen auf Prädikate definiert. Es ist erforderlich, diese Abbildung sehr sorgfältig zu wählen, da sonst schon leicht unterschiedliche Abbildungen zu großen Differenzen in der Größe der Hypothesenräume führen.

Eine einfache und intuitive Art, Datenbankrelationen auf Prädikate abzubilden, ist Abbildung 1[3]:

Abbildung 1: Jede Relation R mit Attributen $A_1, \ldots, A_n$, wird auf ein Prädikat $rn(A_1, \ldots, A_n)$ abgebildet, wobei rn dem Namen von R und jede Argumentstelle einem Attribut entspricht.

Hierbei ist zu beachten, daß Variablen in der Konklusion der Regeln allquantifiziert sind. Wenn sie nicht durch die Prämisse eingeschränkt werden, sind die resultierenden Regeln sehr allgemein. Wenn wir hingegen jedes Attribut einer Relation auf ein Prädikat abbilden, so können wir damit allquantifizierte, unbeschränkte Variablen vermeiden. Darüberhinaus brauchen weniger Regelschemata geschrieben werden, um Regeln zu lernen, die eine ähnliche, aber speziellere Bedeutung haben, wie Regeln, die bei Anwendung von Abbildung 1 gelernt werden können.

Abbildung 2: Für jede Relation R mit Attributen $A_1, \ldots, A_n$, wobei die Attribute $A_j, \ldots, A_l$ den Primärschlüssel bilden, und für jedes $x \in [1, \ldots, n] \backslash [j, \ldots, l]$ bilden wir ein Prädikat $rn\_AX(A_j, \ldots, A_l, A_x)$, wobei AX dem Attributnamen entspricht.

[3] Eine ausführliche Darstellung der nun folgenden Abbildungen zusammen mit ihren Konsequenzen hinsichtlich der Größe des Hypothesenraumes findet sich in [19].

Die Anzahl der resultierenden Prädikate bei Abbildung 2 wird bestimmt durch die Anzahl der Relationen mal der maximalen Anzahl von Attributen einer Relation, jedoch ohne deren Schlüssel. Abbildung 3 reduziert nun die Ausdrucksmächtigkeit auf Aussagenlogik, somit gibt es auch keine zu lernenden Konstanten mehr.

Abbildung 3: Für jedes Attribut A_i, welches kein Schlüsselattribut ist, und welches die Werte $a_1, \ldots, a_n$ besitzt, bilden wir die Menge der Prädikate $rn\_AI\_ai(A_j, \ldots, A_l)$, wobei $A_j, \ldots, A_l$ den Primärschlüssel bilden.

Bei Verwendung der Ergebnisse des Algorithmus Num_Int, oder jedes anderen Algorithmus, der numerische Werte diskretisiert, z. B. [29], kann eine vierte Abbildung angewendet werden.

Abbildung 4: Für jedes Attribut A_i, das kein Primärschlüssel ist und für das Intervalle $<a_{1p}, a_{1q}>, \ldots, <a_{np}, a_{nq}>$ berechnet wurden, wird eine Menge von Prädikaten $rn\_AI\_<a\_ip, a\_iq>\ (A_j, \ldots, A_l)$ gebildet, wobei $A_j, \ldots, A_l$ den Primärschlüssel bilden.

Prädikate sind genau dann wahr, wenn die Werte für das Attribut AI in dem Intervall $<a\_ip, a\_iq>$ liegen. Der Benutzer ist jedoch nicht gezwungen, sich auf eine dieser Abbildungen festzulegen, sondern er kann sowohl diese frei miteinander kombinieren, als auch die Prädikate mit weiteren Attributen ergänzen, z. B. $rn\_AX(A_j, \ldots, A_l, A_x, A_y)$.

Durch die Wahl einer Abbildung und Bestimmung einer Menge von Regelschemata ist der Hypothesenraum festgelegt. Regelschemata können nun für alle Abbildungen angeben werden. Sie beschränken den Hypothesenraum durch die Festlegung von k und c. Die Tiefe einer Variablen kann direkt aus dem Metaprädikat abgelesen werden. All dieses sind wichtige Faktoren für die Lernbarkeit von Klauseln, die nun in der Hand der Benutzer liegen. Ob nun die Variablenbindungen determinierend oder indeterminierend sind — ein weiterer, wichtiger Faktor für die Lernbarkeit — hängt von den Daten ab[4].

Die Systeme Linus [13] und MlSmart [2] führen eine Abbildung von eingeschränkter Prädikatenlogik auf eine *datalog* Repräsentation durch. Die dort verwendeten Abbildungen sind durch die Systeme vorgegeben und können nicht vom Benutzer beeinflußt werden. Im Gegensatz dazu starten wir bei den Datenbankrelationen und bieten dem Benutzer die Möglichkeit, eine funktions–freie Signatur in Prädikatenlogik zu spezifizieren. Die Abbildung von dieser Signatur zurück auf die Datenbank wird dann von Rdt/db automatisch durchgeführt.

[4] Für einen Überblick über Lernbarkeitsresultate im *ILP* siehe [11] und [10].

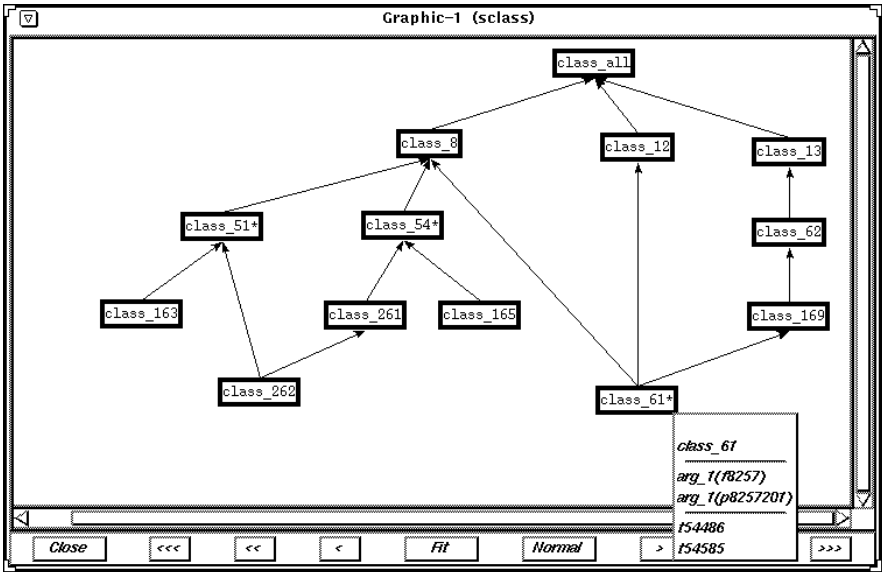

Abbildung 1: Ein Ausschnitt aus der mithilfe von STT aus dem Werkstattbuch berechneten Klassenhierarchie.

3.3 Bildung einer Hierarchie nominaler Attribut–Werte

Im Rahmen der Wissensentdeckung in Datenbanken wird Hintergrundwissen häufig eingesetzt, um nominale Attributwerte zu strukturieren. Die Attributwerte werden dabei meistens dergestalt in einer Hierarchie angeordnet, daß sie von Ebene zu Ebene mehr und mehr abstraktere Werte darstellen. Jedoch liegt zusätzliches Hintergundwissen meistens nur in unstrukturierter Form und außerhalb der Datenbank vor. Es muß folglich erst passend aufbereitet werden, bevor es in der Datenbank dem Lernverfahren zur Verfügung gestellt werden kann. Für diese Aufgabe haben wir das Lernverfahren STT [9],[17] verwendet, das aus Fakten Taxonomien berechnet.

Wir haben textuelles Hintergrundwissen über Teile von Mercedesfahrzeugen als Grundfakten repräsentiert, aus denen dann STT eine Klassenhierarchie induziert hat. So werden Teile aufgrund funktionaler, räumlicher oder ihre möglichen Schadensarten betreffender Zusammenhänge zu Klassen zusammengefaßt [26].

Abbildung 1 zeigt einen Ausschnitt aus der Klassenhierarchie. Insbesondere kann man sich zu jeder Klasse ihre Mitglieder und die Extensionen anzeigen lassen. So besteht die Klasse *class_61* aus den Mitgliedern *arg_1(f8257)* und *arg_1(p8257201)*, und ihre Extension lautet *(t54486, t54585)*. Die Nummern $t54486$ und $t54585$ bezeichnen Fahrzeugteile, das Prädikat $f8257$ gibt einen funktionalen Zusammenhang an, und das Prädikat $p8257201$ beschreibt einen

Tabelle 3: Zusammenfassung der Datensätze

Datensatz	t	j	Attribute	i	Tupel	Fakten
Fahrzeugteile	–	–	604 Prädikate	–	–	20,794
Benziner, manuell	23	3	1 – 3	166	111,995	39,000
Benziner, Autom.	23	3	1 – 3	166	111,995	2,080,000
Diesel, Autom.	23	3	1 – 3	166	111,995	130,000
Benziner, manuell +	16	8	1 – 9	166	700,000	57,351
Alle Fahrzeuge	3	3	3 – 9	700,000	750,000	1,500,820

t — Anzahl der Tabellen
j — max. Anzahl der Tabellen in einer Join–Projektion
Attribute — Anzahl der Attribute in den Tabellen
i — max. Anzahl verschiedener Attributwerte (ohne Schlüssel)
Tupel — max. Anzahl an Tupeln in einer Tabelle
Fakten — Anzahl äquivalenter Fakten

räumlichen Zusammenhang zwischen Fahrzeugteilen. Die verwendeten Codes sind für den Benutzer verständlich, da sie aus seiner Anwendungswelt stammen.

Die gelernte Klassenhierarchie wird in eine binäre Darstellung überführt und der Datenbank hinzugefügt. Zusammenfassend können wir festhalten: Während in der ursprünglichen Datenbank nur die Teilenummern zu finden waren, ist nach dieser Aufbereitung des Hintergrundwissens auch Wissen über funktionale, räumliche und die Schadensarten betreffende Beziehungen in der Datenbank verfügbar. Dieses Wissen wird nun von dem Lernverfahren RDT/DB dadurch berücksichtigt, daß es zum Lernen die Klassen anstelle der ursprünglichen Attributwerte verwendet.

4 Experimente

Im Rahmen eines laufenden Forschungsprojekts der Daimler Benz AG haben wir RDT/DB für die Analyse von Fahrzeugdaten verwendet. Die Datenbank enthält alle Fahrzeugdaten einer Fahrzeugbaureihe, darunter alle Garantie– und Kulanzdaten, die diese Fahrzeuge verursacht haben. Der hier verwendete Datenbankausschnitt hat eine Größe von 2,6 Gigabyte. Er besteht aus 40 Tabellen mit jeweils bis zu 40 Attributen, darunter sind einige Tabellen, die bis zu 750.000 Tupel enthalten. Das Hauptinteresse der Anwender ist es, interessante Regeln zu finden, die Fahrzeuge charakterisieren, die durch ihre Beanstandungen Garantie– oder Kulanzfälle auslösten. In Tabelle 3 sind die Datensätze dargestellt, die wir für die verschiedenen Experimente verwendet haben.

Die Daten der Datensätze sind über 23 Tabellen verteilt, die jeweils bis zu neun Attribute enthalten, die wir für die Lernläufe selektiert haben. Hinzu kommen jeweils noch die Schlüsselattribute. Wenn man diese Daten als Grundfakten repräsentieren würde, wie es bei allen anderen *ILP* Lernverfahren üblich ist, erhält man folgende Zahlen. Eine Beobachtung oder ein Beispiel, d. h. ein Fahrzeug, besteht aus 26 Grundfakten: ein Zielprädikat, das aussagt, ob es sich um ein beanstandetes Fahrzeug handelt; 11 Fakten, die den Bauzustand des Fahrzeugs beschreiben; und bis zu 14 Fakten, die die Sonderausstattungen des Fahrzeugs charakterisieren. Die Gruppe der Benziner mit manuellem Schaltgetriebe umfaßt 1.500 Fahrzeuge, die Gruppe der Benziner mit Automatikgetriebe besteht aus 80.000 Fahrzeugen und die Gruppe der Dieselfahrzeuge mit Automatikgetriebe enthält 5000 Fahrzeuge. Dieses erklärt, wie wir die Anzahl der äquivalenten Fakten in Tabelle 3 berechnet haben.

Wir haben die maximale Anzahl der Tupel in den Tabellen mitangegeben, obwohl dieses nicht die Anzahl der Fahrzeuge in den jeweiligen Gruppen ist. Dieses soll einen Eindruck vermitteln, wie groß die Tabellen sind, in denen das Datenbanksystem nach der Charakterisierung der Beanstandungen suchen muß. Dieses ist von Belang, da zum Beispiel alle Angaben über Motoren in einer Tabelle in der Datenbank zu finden sind, wir diese Originaltabelle aber nicht in drei Tabellen aufgelöst haben. Der Grund hierfür ist unser oben erläutertes Ziel, mit RDT/DB ein Data Mining System zu entwickeln, das mit den Daten lernen kann, so wie sie in der Realität zu finden sind.

Eine andere Eigenheit der Daten, die sich für RDT/DB als wichtig herausgestellt hat, ist die maximale Anzahl der Join–Operatoren, die für einen Hypothesentest ausgeführt werden müssen. Die Anzahl der Joins hat auf der einen Seite keinen Einfluß auf die Komplexität des Hypothesenraumes, auf der anderen Seite trägt sie aber entscheidend zur Komplexität des Hypothesentests bei.

Die Ergebnisse unserer Lernaufgaben sind in Tabelle 4 zusammengefaßt. In der Spalte *Konklusionsprädikate* geben wir an, wieviele Prädikate die Prädikatsvariable in der Konklusion des Metaprädikates instantiieren konnten und für wieviele von ihnen wir Regeln gelernt haben. Da wir in dem Akzeptanzkriterium (2) nur ein $\geq$ gefordert haben, sind die Werte in Klammern in Tabelle 4 die Ergebnisse, für die ein „echt größer“ gilt.

Lernaufgabe 0 gibt die benötigte Zeit für die einmalige Aufbereitung des Hintergrundwissens und die Anzahl der gelernten Klassen an. Die Lernaufgabe 1 besteht darin, eine Charakterisierung der Beanstandungen zu lernen. Um die Regeln interessanter zu gestalten, haben wir hier Hintergrundwissen über die Sonderausstattungen der Fahrzeuge mitberücksichtigt. Regeln, die wir gelernt haben, sehen folgendermaßen aus:

$$niveauregulierung(PKW) \rightarrow beanstandet(PKW) \quad (3)$$

$$motor\_typ(PKW, Typ) \& anz\_zyl(Typ, 6) \rightarrow beanstandet(PKW) \quad (4)$$

In Lernaufgabe 2 haben wir für Benziner mit manuellem Schaltgetriebe

Tabelle 4: Zusammenfassung der Lernergebnisse

L	Datensatz	K	a priori Wahrscheinlichkeit	R	Ergebnis	Lernzeit
0	Fahrzeugteile	—	—	498 Kl.	—	12 h
1	Benziner manuell	1 / 1	48.15 %	17 (10)	56.2% (61.8%)	3 m 15 s
	Benziner Autom	1 / 1	84.48 %	79 (74)	94.67% (95.35%)	3 h 2m
	Diesel Autom	1 / 1	99.11%	15 (3)	99.13% (99.24%)	7 m 29 s
2	Benziner manuell +	4 / 9	9.82%	92	15.47%	49 h 33 m
3	alle Fahrzeuge	6 / 9	62.38%	7	68.83%	13 h 30 m

L — Lernaufgabe
K — Anzahl der Konklusionsprädikate
R — Anzahl gelernter Regeln
Ergebnis — durchschnittliche a posteriori Wahrscheinlichkeit

einen Ausschnitt aus der mit STT gelernten Klassenhierarchie als zusätzliches Wissen berücksichtigt. Von den 498 Klassen, die STT gefunden hat, haben wir neun ausgewählt und der Datenbank hinzugefügt. Bei der Bewertung der gelernten Regeln durch die Anwender wurden diese zusätzlichen Klassen auch besonders gewürdigt, da sie wichtige Zusammenhänge modellieren, die vorher nicht in der Datenbank erfaßt wurden und somit für Auswertungen auch nicht zur Verfügung standen.

Die erweiterte Menge der Prädikate in Lernaufgabe 2 bestand nun aus den 11 Bauzustandsbeschreibungen der Fahrzeuge, den Klassen der fehlerhaften Teile und bis zu vier Sonderausstattungen, d. h. wir haben jene Prädikate entfernt, die nie in gelernten Regeln der Lernaufgabe 1 auftauchten. Zusätzlich haben wir Hintergrundwissen über die Regionen, in denen die Beanstandungen auftraten, der Datenbank hinzugefügt. Eine Beobachtung oder ein Fall bestand nun aus 21 Aussagen über ein Fahrzeug, wieder verteilt über diesmal 16 Tabellen. Da ein Fahrzeugteil in verschiedenen Klassen enthalten sein kann, werden die Variablenbindungen innerhalb der Regeln indeterministisch.

$$\begin{aligned} & garantiefall(X1, X2, Rbetr, X4, X5, X6, X7, Kgr, Teil) \;\; \& \\ & werkstattaufenthalt(X1, X2, Rbetr, X4, X5, X6, X7, PKW) \;\; \& \\ & motor\_typ(PKW, 206) \;\; \& \;\; italien(Rbetr) \\ & \qquad\qquad\qquad \rightarrow class\_419(Kgr, Teil) \end{aligned} \tag{5}$$

Tabelle 5: Lernläufe mit FOIL auf künstlich generierten Daten

Datensatz	K	R	Ergebnis	Lernzeit
Benziner, manuell*	1	10	5.6% cov. 99.7% acc.	42 s
Benziner, Autom*	1	139	73.6% cov. 39.9% acc.	8 h 5 m
Diesel, Autom*	1	39	99.47% cov. 25% acc.	4 m 41 s
Benziner, manuell* +	9	6	78,2% cov. 99,9% acc.	3 h 17m

K — Anzahl der Konklusionsprädikate
R — Anzahl gelernter Regeln

Eine der 92 gelernten Regeln ist in (5) wiedergegeben. Beanstandete Fahrzeugteile werden durch zwei Attribute (Kgr,Teil) kodiert, *Rbetr* bezeichnet die Codenummer der Werkstatt, und 206 ist ein bestimmter Motortyp. Die Regel drückt eine erhöhte Anfälligkeit der zur Gruppe 419 gehörenden Fahrzeugteile aus, wobei die einzelnen Fahrzeuge in Italien beanstandet wurden. Darüberhinaus ist ihr Benzinmotor von dem spezielleren Typ 206.

Die Aufgabenstellung der dritten Lernaufgabe weicht insofern von den anderen ab, als daß hier keine Trennung der Fahrzeuge nach ihren Motoren respektive Getrieben mehr stattfand. Die Lernaufgabe besteht hier darin, mittels der von STT gefundenen Klassen Kostenintervalle zu charakterisieren. Dazu wurden zuvor mit NUM_INT neun disjunkte Intervalle über dem Attribut gelernt, das die durch die 750.000 Beanstandungen aller Fahrzeuge verursachten Kosten enthält.

$$\begin{aligned} &garantiefall(X1, X2, Rbetr, X4, X5, X6, X7, Kgr, Teil)\ \& \\ &class\_35(Kgr, Teil) \\ &\qquad \rightarrow kosten\_0\_500(X1, X2, Rbetr, X4, X5, X6, X7, Kgr, Teil) \end{aligned} \tag{6}$$

Die Regel (6) zeigt eine der sieben gelernten Regeln. Bei diesem Lernlauf haben wir zwei Regeln mit identischen Prämissen, aber unterschiedlichen Kostenintervallen als Konklusionsprädikat gelernt. Dies ist ein Beispiel für Regeln, deren Entdeckung durch einen Begriffslerner sehr unwahrscheinlich ist. Diese zwei Regeln sagen uns, daß die Zugehörigkeit von Teilen zur Klasse 35 zwei unterschiedliche Kostenintervalle wahrscheinlicher macht, als diese normalerweise sind; entsprechend reduzieren sich die Wahrscheinlichkeiten der anderen sieben Kostenintervalle. Die anderen fünf Regeln geben jeweils einen Zusammenhang zwischen genau einer Klasse und einem Intervall an.

Um einen Eindruck zu gewinnen, wie andere *ILP* Lernverfahren sich auf diesen großen Datenmengen bewähren, haben wir den Begriffslerner FOIL [25] und PROGOL [21] auf ähnlichen Daten ausprobiert (Tabelle 5). Wir haben künstliche Daten generiert, die die von RDT/DB benutzten echten Daten so genau wie möglich wiederspiegeln. Die Generierung der künstlichen Daten

— und damit indirekt die Lernaufgabe — haben wir vereinfacht, indem wir die durch RDT/DB gelernten Regeln verwendet haben. So verzichteten wir darauf, Daten für irrelevante Attribute zu erzeugen, die in keinen gelernten Regeln vorkamen. Daher ergeben sich für den Datensatz *Benziner Autom** nur 1.300.000 anstelle der 2.080.000 berechneten Fakten.

Darüberhinaus liefen sowohl FOIL als auch PROGOL auf einer Sparc 20 mit 128 MByte Hauptspeicher, während der echten Datenbank auf gleichem Rechnertyp nur 96 MByte zur Verfügung stand. Da FOIL die einfachere Aufgabe des Begriffslernens löst, hört FOIL auf zu lernen, wenn die Definition des Begriffes *Beanstandung* exakt genug ist. Hingegen muß ein Regellerner selbst bei einer gefundenen, angemessenen Begriffsbeschreibung weiter lernen, um alle generellsten, gültigen und nicht redundanten Regeln zu entdecken.

Wegen dieser Vereinfachungen kann der Vergleich von FOIL mit RDT/DB nur die Unterschiede beim Lernen auf sehr großen Datenmengen aufzeigen. Für die Lernergebnisse von FOIL haben wir jeweils die erreichte *Accuracy* und *Coverage* angegeben. Für drei der vier Lernläufe benötigte FOIL weniger Zeit als RDT/DB. Bemerkenswert ist aber, daß FOIL über fünf Stunden länger für den Datensatz *Benziner Autom** benötigte als RDT/DB. Betrachtet man den Speicherbedarf von FOIL, so zeigt sich sehr deutlich, daß FOIL stark davon abhängig ist, daß alle Daten in den physikalischen Hauptspeicher passen. Diese Grenze wurde bei diesem Datensatz mit 130 MByte gerade überschritten.

Darüberhinaus fällt auf, daß FOIL sehr viel weniger Zeit für den Datensatz *Benziner manuell**+ benötigte. Diese Daten passen alle in den Hauptspeicher, und bei RDT/DB muß man hier für die Datenbanklösung nicht zuletzt wegen der Joins einen hohen Tribut bezahlen. Bleibt noch anzumerken, daß PROGOL nicht in der Lage war, innerhalb von 72 Stunden Rechenzeit irgendein Resultat zu liefern. Auf dem Datensatz *Benziner Autom** brach es schon nach 10 Minuten mit einem Fehler ab.

5 Diskussion

In diesem Papier haben wir einen Ansatz präsentiert, um Regeln in eingeschränkter Prädikatenlogik in großen Datenbanken zu lernen. Wir haben die verschiedenen Möglichkeiten diskutiert, wie die Prädikate unserer Hypothesensprache auf die Datenbankrelationen abgebildet werden können. Die Analyse der Konsequenzen, die diese Abbildungen auf die Größe des resultierenden Hypothesenraumes haben, führt zu der Erkenntnis, daß es erstrebenswert ist, wenn das Regellernverfahren durch weitere, einfache Algorithmen unterstützt wird. Die Umsetzung in einen Multistrategieansatz [18] konnten wir hier nur am Beispiel der Integration von aufbereitetem Hintergrundwissen anreißen.

Die Frage, ob wie in unserem Fall der Anwender die der Lernaufgabe angemessenen Ebenen (hier: neun Klassen) aus der gelernten Klassenhierarchie auswählen soll, bleibt offen. Wir vertreten hier wie Brachman und Anand die Position, daß der Anwender direkt in den *KDD* Prozeß involviert sein soll [3]. Denn es ist für den Anwender einfacher, aus einer graphischen Darstellung wie Abb.1 die interessanten Bereiche auszuwählen, anstatt seine Vorstellungen durch einen weiteren, deklarativen Bias bzgl. relevanter Klassen zu formulieren.

Weiterhin zeigt die von FOIL benötigte Zeit für den Fall, daß die Tupel nicht mehr in den physikalischen Hauptspeicher passen (fünf Stunden länger als RDT/DB), daß sehr viel Zeit durch den direkten Zugriff auf eine Datenbank eingespart werden kann. Hier zahlt sich der Hypothesentest durch eine Datenbanklösung aus.

Der direkte Zugriff auf eine gegebene Datenbank ist auch sehr verschieden von dem von MLSMART verfolgten Ansatz [2]. Bei gegebener Datenbank kann die Art der Datenspeicherung nicht mehr beeinflußt werden. Darüberhinaus wurde diese zu anderen Zwecken als Lernen entworfen und kann auch nur abgefragt werden. MLSMART hingegen benutzt eine selbstentwickelte Datenbank, in der es neue Tabellen kreiert, Zwischenergebnisse in eigenen Tabellen speichert oder Aggregationen der Daten erzeugt. Dadurch ändert sich beim Lernen fortlaufend die Datenbank[5].

RDT/DB schließt die Lücke zwischen den Anforderungen, die das Lernverfahren stellt, und den gegebenen Daten. Es benutzt Deklarationen der Art der Abbildung zwischen der Datenbank und der Hypothesensprache. RDT/DB kontrolliert, ob Hypothesen unerfüllbar sind. Es transformiert den deklarativen Bias automatisch in die Anfragesprache der Datenbank. Diese Dienste, die RDT/DB dem Anwender bietet, entsprechen einer Reihe von Vorverarbeitungsschritten, die sonst bei der Anwendung herkömmlicher Lernverfahren notwendig sind, bevor diese überhaupt mit dem Lernen beginnen können.

Der Algorithmus des Regellernverfahrens RDT ist besonders gut für Data Mining Aufgaben im *KDD* Prozeß geeignet, da seine Komplexität nicht von der Anzahl der Tupel sondern von der Art der Hypothesensprache abhängt. Der deklarative, syntaktische Bias ist besonders nützlich zur Beschränkung des Hypothesenraumes, wenn man über sehr großen Datenmengen lernen möchte.

Wir können somit sagen, daß wir unser in der Einleitung formuliertes Ziel erreicht haben. Wir haben über einer sehr großen Tupelmenge, die über mehrere Tabellen in einer gegebenen Datenbank verteilt war, die schwierige Regellernaufgabe im *ILP* erfolgreich gelöst. Folglich lautet unser Fazit: Es geht! Allerdings haben wir in unseren Experimenten aber auch die Grenzen unseres Ansatzes für den Fall gesehen, haben, wenn die Tupelmenge zu klein ist. Hier bedarf es noch einiger Forschung, die genauen Bedingungen zu finden, unter denen die Datenbank Vorteile gegenüber alternativen Ansätzen wie Sampling oder Datenhaltung durch das Lernverfahren hat.

[5] Ob der von MLSMART verfolgte Ansatz besser ist, bleibt fraglich. Zum einen sind in [2] keine Experimente mit großen Datenmengen zu finden. Viel gravierender ist aber die Analyse von Pazzani und Kibler, daß MLSMART in doppelt exponentieller Zeit läuft [22].

Danksagung. Unser Dank gilt dem Daimler–Benz Forschungszentrum Ulm für die Unterstützung der hier präsentierten Arbeit (Vertragsnr. 094 965 129 7/0191).

Literatur

[1] R. Agrawal und R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, 1994.

[2] F. Bergadano, A. Giordana, und L. Saitta. *Machine Learning: an integrated framework and its applications*. Ellis Horwood, 1991.

[3] R. J. Brachman und T. Anand. The process of knowledge discovery in databases: A human-centered approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, und R. Uthurusamy, Hrsg., *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[4] L. Breiman. Pasting bites together for prediction in large data sets and on-line. Technical report, Statistics Departement, University of California, Berkeley, CA 94708, November 1996.

[5] Y. Cai, N. Cercone, und J. Han. Attribute–oriented induction in relational databases. In G. Piatetsky-Shapiro und W. Frawley, Hrsg., *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.

[6] S. Dzeroski. Inductive logic programming and knowledge discovery in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, und R. Uthurusamy, Hrsg., *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[7] C. Franzel. Auffinden interessanter Wertebereiche in Datenbankattributen. Diplomarbeit, Fachbereich Informatik, Universität Dortmund, Oktober 1996.

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

[9] J.-U. Kietz. Incremental and reversible acquisition of taxonomies. In J. M. Boose, editor, *Proceedings of EKAW–88*, GMD, Sankt Augustin, 1988. Auch als KIT-Report 66, TU Berlin.

[10] J.-U. Kietz. *Induktive Analyse relationaler Daten*. Dissertation, TU Berlin, 1996.

[11] J.-U. Kietz und S. Dzeroski. Inductive logic programming and learnability. *SIGART–Bulletin*, 5(1):22–32, 1994.

[12] J.-U. Kietz und S. Wrobel. Controlling the complexity of learning in logic through syntactic and task–oriented models. In S. Muggleton, Hrsg., *Inductive Logic Programming*. Academic Press, 1992.

[13] N. Lavrač und S. Džeroski. *Inductive Logic Programming — Techniques and Applications*. Ellis Horwood, 1994.

[14] G. Lindner und K. Morik. Coupling a relational learning lagorithm with a database system. In Y. Kodratoff, G. Nakhaeizadeh, und C. Taylor, Hrsg., *Statistics, Machine Learning, and Knowledge Discovery in Databases*, MLnet Familiarization Workshops. MLnet, April 1995.

[15] H. Mannila. Aspects of data mining. In Y. Kodratoff, G. Nakhaeizadeh, und C. Taylor, Hrsg., *Statistics, Machine Learning and Knowledge Discovery in Databases*, MLnet Familiarization Workshops. MLnet, April 1995.

[16] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, und T. M. Mitchell, Hrsg., *Machine Learning — An Artificial Intelligence Approach*. Morgan Kaufmann, 1983.

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

[18] K. Morik und P. Brockhausen. A multistrategy approach to relational knowledge discovery in databases. In R. S. Michalski und J. Wnek, Hrsg., *Proceedings of the Third International Workshop on Multistrategy Learning*, 1996. AAAI Press.

[19] K. Morik und P. Brockhausen. A multistrategy approach to relational knowledge discovery in databases. *Machine Learning Journal*, 1997. to appear.

[20] S. Muggleton und L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[21] S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13:245–286, 1995.

[22] M.J. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.

[23] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro und W. Frawley, Hrsg., *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.

[24] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993.

[25] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[26] M. Siebert. Erwerb funktionaler, räumlicher und kausaler Beziehungen von Fahrzeugteilen aus einer technischen Dokumentation. Diplomarbeit, Fachbereich Informatik, Universität Dortmund, Januar 1997.

[27] A. Silberschatz und A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, December 1996.

[28] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer–Verlag, 1995.

[29] D. Wettscherek und T. G. Dietterich. An experimental comparison of the nearest-neighbour and nearest-hyperrectangle algorithms. *Machine Learning*, 19(1):5 – 27, April 1995.