Suchraumbeschränkung für relationales Data Mining

Von der Fakultät für Informatik, Elektrotechnik
und Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Irene Weber
aus Gosheim (Württ.)

Hauptberichter:	Prof. Dr. E. Lehmann
Mitberichter:	Prof. Dr. S. Wrobel
Tag der mündlichen Prüfung:	17. Mai 2004

Institut für Intelligente Systeme der Universität Stuttgart

2004

Zusammenfassung

Data Mining und Knowledge Discovery in Databases (KDD) sind Forschungs- und Anwendungsgebiete, die sich mit der Extraktion von nutzbarem Wissen aus Daten befassen. Dazu werden unter anderem Methoden des maschinellen Lernens eingesetzt. Die Induktive Logikprogrammierung ist ein Teilgebiet des Maschinellen Lernens, dessen Gegenstand das Lernen aus multi-relational und prädikatenlogisch repräsentierten Daten ist. Dies macht ILP besonders interessant für KDD und Data Mining. Der Einsatz von ILP-Methoden für KDD und Data Mining wird neuerdings auch als Relationales Data Mining bezeichnet.

Zu den Anwendungsgebieten von KDD und Data Mining gehören die Teilgruppenanalyse und das Finden interessanter Zusammenhänge. Diese Arbeit hat sich zum Ziel gesetzt, Methoden zur sicheren Beschränkung des Suchraums bei der Suche nach interessanten Teilgruppen im Rahmen der ILP zu erarbeiten und zu evaluieren. Sie leistet damit einen Beitrag zur technischen Seite des relationalen Data Mining.

Die Arbeit bietet im einzelnen: (1) eine Formalisierung der Teilgruppenanalyse im Rahmen der ILP, (2) Optimumschätzfunktionen zur Verteilungsungewöhnlichkeit und zur Folgerungsstärke, (3) die Erweiterung des bekannten Apriori-Suchverfahrens um *req*- und *excl*-Bedingungen, (4) einen ILP-Sprachbias für die Teilgruppenanalyse, der die Anwendung der Teilmengenbedingung des Apriori-Suchverfahrens zur Beschränkung eines ILP–Suchraums erlaubt, (5) einen SQL–Sprachbias für die Teilgruppenanalyse in multi-relationalen Datenbanken, (6) einen Ansatz zur Integration der Suchraumbeschränkung anhand von Taxonomien in einen Apriori-artigen Suchalgorithmus, (7) eine Methode zur Behandlung diskretisierter numerischer Attribute, die die Suchraumbeschränkung anhand von Allgemeinerbeziehungen zwischen Intervallen vereinheitlicht mit der Suchraumbeschränkung anhand von Taxonomien, (8) Experimente zur Wirksamkeit der verschiedenen Möglichkeiten zur Suchraumbeschränkung, (9) die Anwendung der entwickelten Ansätze auf ein echtes Data Mining-Problem und ausführliche Vergleiche mit verwandten Arbeiten.

Die Experimente wurden mit einer prototypischen Implementation der in dieser Arbeit entwickelten Ansätze durchgeführt. Dabei haben sich Teilmengenbedingung und Optimumschätzfunktionen als wirkungsvolle und zuverlässige Methoden zur Beschränkung des Suchraums erwiesen, während der Beitrag der Taxonomien zur Suchraumbeschränkung zwischen verschiedenen Anwendungen stark schwankte und in einigen Fällen nur gering war. Ein wichtiges Ergebnis der Versuche ist, daß die Teilmengenbedingung als Suchbeschränkung für multi-relationale Datenbanken und ILP-Sprachen genauso wirkungsvoll sein kann wie für ein-relationale Datenbanken.

Summary

Knowledge discovery and data mining are concerned with the discovery of valid, novel, potentially useful, and understandable patterns in data. Most data mining algorithms require that the data are represented as a single, attribute-value table. In contrast, relational data mining techniques are applicable directly to multi-relational databases. This thesis develops and investigates pruning techniques that are applicable for pattern discovery within a restricted ILP setting where all patterns describe subgroups of a fixed population of individuals. This variant of pattern discovery is also known as relational subgroup discovery. The main contribution of the thesis is an Apriori-like search algorithm for relational subgroup discovery. Formerly, Apriori-like search was applied only in attribute-value (i. e., non-relational) settings.

In particular, the contributions of the thesis are
(1) a formal description of subgroup discovery in the framework of ILP (see also [Wro97, Deh98, Web00, FL01],
(2) optimum estimates for the interestingness criteria *distributional unusualness* [Wro97] and *implication intensity* [FDPB95],
(3) the extension of the well-known Apriori algorithm [MTV94] by *req-* and *excl*-conditions that allow to constrain the set of patterns searched by Apriori,
(4) an ILP language bias that allows the application of an Apriori-like algorithm for relational subgroup discovery,
(5) an SQL–based language bias for relational subgroup discovery via queries to a relational database management system,
(6) a novel approach for integrating pruning based on structured attributes in an Apriori-like search algorithm,
(7) an approach for integrating pruning based on dicreetized numerical attributes in an Apriori-like search algorithm in a manner uniform to (6),
(8) an experimental evaluation of the various pruning methods (namely, optimum estimates, Apriori-like pruning, use of structure in attributes for pruning),
(9) the application of the approach for data mining in a real-world financial database,
(10) and extensive comparisons with related work.

A more detailed english abstract can be found in Appendix B. Parts of the work have been published in [Web02, Web00, Web99b, Web99a, Web98b, Web98a, Web97].

Dank

Diese Dissertation ist in der Abteilung Intelligente Systeme am Institut für Informatik[1] der Universität Stuttgart unter der Leitung von Herrn Prof. Dr. Egbert Lehmann entstanden. Herr Prof. Dr. Lehmann hat mir bei der Themenwahl und der Ausgestaltung der Arbeit Freiräume gelassen und die Arbeit in allen Phasen mit großem Interesse begleitet und gefördert. Schließlich danke ich ihm für die Übernahme des Hauptberichts und Herrn Prof. Dr. Stefan Wrobel für die Übernahme des Mitberichts.

Den Kollegen in der Abteilung danke ich für ein anregendes und freundliches Arbeitsumfeld. Dr. Birgit Tausend und Dr. Irene Stahl haben durch ihre Arbeit unserer Abteilung in der Induktiven Logikprogrammierung einen guten Namen verschafft und mir den Einstieg in dieses Gebiet sehr erleichtert. Sie haben auch Anteil am Aufbau des internationalen Forschungsnetzwerks, das die vorliegende Arbeit ganz wesentlich beeinflußt hat. Das Network of Excellence in Inductive Logic Programming ILPnet2, finanziert von der Europäischen Kommission unter Kontrakt INCO 977102, hat meine Konferenzteilnahme finanziell unterstützt.

Dr. Rüdiger Wirth von der DaimlerChrysler Forschung danke ich für zahlreiche Diskussionen wie auch für die hervorragende Zusammenarbeit bei verschiedenen Seminaren und Vorlesungen und bei der gemeinsamen Betreuung von Studien- und Diplomarbeiten, die mir immer wieder die praktische Relevanz der Forschung vor Augen geführt haben.

Mein Dank gilt auch Dr. Stefan Rapp und Prof. Dr. Grzegorz Dogil vom Lehrstuhl experimentelle Phonetik am Institut für maschinelle Sprachverarbeitung für die Überlassung der Phonetik-Datenbank, die ich für meine Untersuchungen verwenden konnte.

Am Institut für parallele und verteilte Höchstleistungsrechner möchte ich Herrn Dr. Holger Schwarz von der Abteilung Anwendersoftware danken für etliche interessante Diskussionen. Die Abteilung hat mich freundlicherweise zeitweilig mit Rechenleistung und dem Zugriff auf ihr Datenbankmanagement-System unterstützt.

Zum Erfolg dieser Arbeit haben weiterhin die Studenten, vor allem Stefan Escher, Thomas Klenk, Martin Bauer und Oliver Welte, beigetragen, bei denen ich mich ebenfalls bedanken möchte.

Mein Dank gilt auch meiner Familie. Meiner Schwiegermutter Sigrid Rapp danke ich für ihre Mithilfe bei der Kinderbetreuung. Auf sie war auch kurzfristig immer Verlaß. Auch meinen Eltern danke ich für ihre Unterstützung und besonders dafür, daß sie mir eine akademische Ausbildung ermöglicht haben. Mein Mann Stefan Rapp hat die Freuden und Mühen mit mir geteilt, die unsere wachsende Familie und unsere wissenschaftlichen Tätigkeiten mit sich gebracht haben, und dabei nie die Zuversicht verloren. Dafür, und für seine Mithilfe bei der Fertigstellung der Arbeit danke ich ihm sehr.

[1] jetzt Institut für Intelligente Systeme an der Fakultät für Informatik, Elektrotechnik und Informationstechnik

Inhaltsverzeichnis

Kapitel 1

Einleitung

In fast allen menschlichen Tätigkeitsbereichen kommen heute Computer zum Einsatz. In Verwaltung, Industrie und Handel, im Gesundheitswesen und in der Wissenschaft, im Bereich der Presse und der öffentlichen Medien werden Daten am Computer erfaßt und verwaltet. Internet und World Wide Web gewinnen an Bedeutung und nicht zuletzt dienen Computer als Schreibgerät. Vorgänge, die am Computer bearbeitet werden, werden meist auch protokolliert und die anfallenden Daten dauerhaft gespeichert, zumal die Speichermedien leistungsfähiger und billiger, die Datenbanken zuverlässiger und einfacher bedienbar werden. Dabei sammeln sich überwältigende Datenmengen an. Ihre Besitzer haben erkannt, daß diese Daten einen Schatz von Erfahrungen und Informationen darstellen, die sich als Grundlage für Verbesserungen bestehender Geschäftsprozesse und für in die Zukunft reichende Entscheidungen nutzen lassen.

Um aus den Datenmassen und -strömen nützliches und verwertbares Wissen zu ziehen, ist eine radikale Informationsreduktion und -abstraktion notwendig, im Ausmaß vergleichbar mit dem, was die menschliche Sinneswahrnehmung und das menschliche Gedächtnis leisten, um aus der Flut einströmender Umgebungsreize und gespeicherter Erfahrungen aktuell verwertbare Eindrücke auszufiltern. Die traditionelle Datenanalyse, bei der die Analysten gezielt Hypothesen über die Daten formulieren und durch Inspektion der Daten zu verifizieren suchen, kann den gestiegenen Anforderungen nicht mehr genügen. Aus dem Bedürfnis nach mächtigeren und stärker automatisierten Verfahren hat sich seit Ende der 80er Jahre aus den Wurzeln vor allem der Statistik, des Maschinellen Lernens und der Datenbank-Techniken Knowledge Discovery in Databases und Data Mining als ein eigenständiges Forschungsgebiet herausgebildet.

1.1 Knowledge Discovery in Databases und Data Mining

Unter Knowledge Discovery in Databases (KDD, Wissensentdeckung in Datenbanken) versteht man

> ... den Prozeß, (statistisch) gültige, neue, möglicherweise nützliche und letztendlich verständliche Struktur in Daten zu identifizieren.
>
> [Fay98, FPSS96]

Als Data Mining (DM) bezeichnet man den zentralen Schritt innerhalb dieses Prozesses, der darin besteht, ausgewählte Analyse- und Suchalgorithmen auf die aufbereiteten Daten anzuwenden. Häufig werden die beiden Begriffe KDD und DM auch synonym verwendet.

Der Begriff *Data Mining* (wörtlich übersetzt „in Daten schürfen“) tauchte in den sechziger Jahren ursprünglich in der Statistikforschung auf, neben den synonym verwendeten Ausdrücken *data fishing* und *data dredging* (*to dredge* heißt unter anderem „mit dem Schleppnetz fischen, ausbaggern“) [PS98, Fay97]. Diese Begriffe, die durchaus abwertend

gemeint sind, veranschaulichen den Unterschied zwischen traditioneller Datenanalyse und den neuen Methoden:

> ... traditional data analysis is assumption-driven in the sense that a hypothesis is formed and validated against the data, whereas data mining in contrast is discovery-driven in the sense that patterns are automatically extracted from the data, which requires substantial search efforts.
>
> [Han99]

Eine wichtige Anwendung des Data Mining ist das Entdecken interessanter Muster in Datenbanken mit dem klassischen Einsatzbereich Warenkorbanalyse. Die Datenbank besteht dabei aus einer Menge von Warenkörben, die auch als Transaktionen bezeichnet werden und beispielsweise Einkäufe im Supermarkt oder Bestellungen im Versandhandel beschreiben. Die darin gesuchten Muster sind Assoziationsregeln, die Zusammenhänge folgender Art beschreiben: wenn bestimmte Artikel A, B, C zusammen in einem Warenkorb vorkommen, so enthält er mit einer gewissen Sicherheit auch die Artikel X, Y, Z. Der erste Schritt zur Gewinnung von Assoziationsregeln besteht darin, häufige Assoziationen von Artikeln zu bestimmen; das sind Mengen von Artikeln, die in einer großen Zahl von Warenkörben zusammen vorkommen.

Beispiel 1 *Im Online-Buchhandel kann eine Assoziationregel so lauten:*

> *Viele der Kunden, die das Buch* Harry Potter and the Order of the Phoenix, Bd.5 *von J. K. Rowling bestellt haben, haben gleichzeitig auch die Bücher* Harry Potter and the Goblet of Fire, Bd.4 *von J. K. Rowling,* The Ersatz Elevator *von Lemony Snicket und* The Bonesetter's Daughter *von Amy Tan bestellt.*

Für einen Buchversender sind solche Zusammenhänge in seinen Daten interessant, weil er daraus automatisch persönliche Buchempfehlungen für seine Kunden generieren kann. Wenn beispielsweise ein Kunde das Buch Harry Potter and the Order of the Phoenix, Bd.5 *bestellt, gefallen ihm möglicherweise auch die drei anderen in der Assoziationsregel genannten Bücher und können ihm daher besonders empfohlen werden, um ihn zu weiteren Bestellungen zu animieren.*

1.2 Induktive Logikprogrammierung

Viele Verfahren und Methoden, die im Rahmen der KDD eingesetzt und weiterentwickelt werden, stammen aus dem Maschinellen Lernen, einem Teilgebiet der Künstlichen Intelligenz. Der kognitionswissenschaftlich ausgerichtetete Zweig des Maschinellen Lernens erforscht menschliches Lernen mit dem Ziel, es auf dem Computer nachzubilden. Von größerer Bedeutung für die KDD ist die ingenieursmäßig orientierte Richtung des Maschinellen Lernens, die auf Computer zugeschnittene Algorithmen zur automatischen Gewinnung von Wissen aus Daten entwickelt. Diese Algorithmen setzen meist Daten in Attribut-Wert-Repräsentation voraus. Solche Daten bestehen aus einer Menge sogenannter Beispiele, von denen jedes durch einen Vektor von Merkmalen über denselben Attributen beschrieben ist. Die Merkmalsvektoren lassen sich zu einer Tabelle anordnen.

Beispiel 2 (aus [Dže96b]) *Die Tabelle Möglicher-Kunde in Tafel 1.1 repräsentiert Beispiele von möglichen Kunden durch Merkmalsvektoren über den Attributen Name, Alter, Geschlecht, Gehalt und Kunde.*
Angenommen, der Besitzer dieser Daten interessiert sich dafür, Kunden von Nicht-Kunden zu unterscheiden. Wenn er nur die Daten der Tabelle Möglicher-Kunde untersucht, kann er die folgenden Zusammenhänge entdecken:

$$wenn \qquad gehalt(Person) \geq 100\,000, \qquad dann \qquad kunde(Person) = ja.$$

Tafel 1.1 Die Relation *Möglicher-Kunde*.

Person	Alter	Geschlecht	Gehalt	Kunde
Ann Smith	32	w	10 000	ja
Joan Gray	53	w	500 000	ja
Mary Blythe	27	w	20 000	nein
Jane Brown	55	w	20 000	ja
Bob Smith	30	m	100 000	ja
Jack Brown	50	m	200 000	ja

Tafel 1.2 Die Relation *Verheiratet*.

Mann	Frau
Bob Smith	Ann Smith
Jack Brown	Jane Brown

(Eine Person ist Kunde, wenn ihr Gehalt 100 000 *Euro übersteigt.)*

$$\begin{array}{lll} wenn & geschlecht(Person) = w & und \quad alter(Person) \geq 32, \\ dann & kunde(Person) = ja. & \end{array}$$

(Eine Person ist Kunde, wenn sie weiblich und mindestens 32 Jahre alt ist.)

Die Ausdrucksstärke der Attribut-Wert-Repräsentation entspricht der Aussagenlogik. Die Induktive Logikprogrammierung (ILP) ist ein Teilgebiet des Maschinellen Lernens, das sich im Unterschied dazu mit dem Lernen aus Logikprogrammen und relationalen und deduktiven Datenbanken befaßt, deren Ausdrucksstärke der Prädikatenlogik entspricht. Relationale Datenbanken und relationale Datenbankmanagementsysteme sind heute weit verbreitet. Relationale Datenbanken repräsentieren die Daten in mehreren Relationen, zwischen denen vielfältige Beziehungen bestehen können.

Beispiel 3 (zu Beispiel 2) *Die Datenbank mit Kundeninformationen enthält außer der Tabelle Möglicher-Kunde in Tafel 1.1 auch die Relation Verheiratet in Tafel 1.2. Wenn auch diese Relation ausgewertet wird, ergeben sich zusätzlich die folgenden Regeln:*

$$\begin{array}{lll} wenn & verheiratet(Mann, Person) & und \quad gehalt(Mann) \geq 100\,000, \\ dann & kunde(Person) = ja. & \end{array}$$

(Eine Person ist Kunde, wenn sie mit einem Mann verheiratet ist, dessen Gehalt 100 000 *Euro übersteigt.)*

$$\begin{array}{lll} wenn & verheiratet(Mann, Person) & und \quad kunde(Mann), \\ dann & kunde(Person) = ja. & \end{array}$$

(Eine Person ist Kunde, wenn sie mit einem Mann verheiratet ist, der Kunde ist.)

$$\begin{array}{lll} wenn & verheiratet(Person, Frau) & und \quad kunde(Frau), \\ dann & kunde(Person) = ja. & \end{array}$$

(Eine Person ist Kunde, wenn sie mit einer Frau verheiratet ist, die Kunde ist.)

Tafel 1.3 Die Verbundrelation *Möglicher-Kunde-2*.

Person	...	Gehalt	Kunde	Gatte	...	G-Gehalt	G-Kunde
Ann Smith	...	10 000	ja	Bob Smith	...	100 000	ja
Joan Gray	...	500 000	ja	n/a	...	n/a	n/a
Mary Blythe	...	20 000	nein	n/a	...	n/a	n/a
Jane Brown	...	20 000	ja	Jack Brown	...	200 000	ja
Bob Smith	...	100 000	ja	Ann Smith	...	10,000	ja
Jack Brown	...	200 000	ja	Jane Brown	...	20 000	ja

Aus mehreren Relationen bestehende Datenbanken lassen sich im allgemeinen nicht in ein-relationale Attribut-Wert-Darstellungen überführen, ohne daß die transformierte Datenbank zu unvernünftigem Umfang anschwillt.

Beispiel 4 (zu Beispiel 2) *Versucht man, die Tabellen Möglicher-Kunde und Verheiratet zu einer Relation zu verbinden, so enthält diese Relation viele unbekannte Werte (n/a) und Mehrfachrepräsentationen von Einträgen wie die Beispiel-Verbundrelation in Tafel 1.3.*

Die Anwendung der ILP auf Data Mining-Aufgaben wird neuerdings auch als Relationales Data Mining bezeichnet. [DL01]. Der Vorzug des relationalen Data Mining gegenüber herkömmlichen Ansätzen liegt darin, daß die ILP Methoden und Algorithmen bietet, die direkt auf relationale Datenbanken anwendbar sind und keine Transformation in ein einrelationales Format erfordern. Die überlegene Ausdrucksstärke der in der ILP verwandten prädikatenlogischen Repräsentationen erlaubt, Hypothesen über die Daten zu formulieren und auszuwerten, die in der Aussagenlogik nicht darstellbar sind. Viele aussagenlogische Aufgabenstellungen lassen sich als Spezialfälle der ILP beschreiben.

Als individuenzentriert bezeichnet man Data Mining-Anwendungen, die auf das Lernen über eine Population von Individuen ausgerichtet sind, beispielsweise über die Personen, die in der Relation $Kunde$ beschrieben sind. Das Lernen aus Attribut-Wert-Repräsentationen ist immer individuenzentriert, weil jede Zeile einer Attribut-Wert-Tabelle ein Individuum beschreibt und jedes Individuum nur mit Merkmalen aus der ihm entsprechenden Zeile beschrieben wird. Im Unterschied dazu erlaubt die ILP auch nicht-individuenzentrierte Anwendungen, zum Beispiel das Lernen von Logikprogrammen wie $quicksort(List1, List2)$ oder Familienbeziehungen wie $vorfahr\_von(Person1, Person2)$.

Beispiel 5 *Gegeben sei eine Datenbank, die Personen und ihre Verwandschaftverhältnisse repräsentiert. Eine Beschreibung der Beziehung $vorfahr\_von(Person1, Person2)$ soll gelernt werden. Beispiele für diese Beziehung sind Personenpaare; zur Beschreibung der Beziehung $vorfahr\_von(Person1, Person2)$ sind aber Beziehungen der beteiligten Personen $Person1$ und $Person2$ zu anderen, nicht zum Paar gehörenden Personen relevant. So lassen sich weder einzelne Personen noch Personenpaare als die eine Individuenpopulation angeben, die im Zentrum der Lernbemühungen steht.*

Individuenzentrierte Anwendungen der ILP (wie in Beispiel 3) sind direkte Verallgemeinerungen des Lernens aus Attribut-Wert-Repräsentationen. Daher lassen sich bestimmte Vorgehensweisen und Algorithmen des Lernens aus Attribut-Wert-Repräsentationen auf die individuenzentrierte ILP übertragen. Die individuenzentrierte ILP bildet den formalen Rahmen dieser Arbeit.

1.3 Finden interessanter Muster und Teilgruppenanalyse

1.3.1 Aufgabenstellung

Eine allgemeine Aufgabe, die im Rahmen des KDD zu lösen ist, ist die Suche nach interessanten Mustern in Datenbanken. Sie ist festgelegt durch

- die Datenbank,
- eine Menge oder Sprache von Mustern, die in der Datenbank vorkommen können, und
- ein Akzeptanzkriterium, das angibt, ob ein Muster der vorgegebenen Sprache in der gegebenen Anwendung potentiell interessant ist.

Beispiel 6 (zu Beispiel 1) *Im Buchversandhandel liegt eine Datenbank mit Bestellungen vor. Die darin gesuchten Muster sind Assoziationen von Büchern. Die möglichen Assoziationen sind Teilmengen der Menge aller bestellbaren Bücher. Eine Assoziation ist interessant, wenn sie in mindestens 0.05% der Bestellungen vorkommt.*

Die Teilgruppenanalyse ist eine besondere Form des Findens interessanter Muster; man könnte sie als individuenzentrierte Version dieser Aufgabenstellung betrachten. Bei der Teilgruppenanalyse beschreiben die gesuchten, potentiell interessanten Muster Teilmengen einer vorgegebenen, festen Population von Fällen (Beispielen oder Individuen).

Beispiel 7 (zu Beispiel 1) *Im vorhergehenden Beispiel kann die Gesamtheit aller in der Datenbank beschriebenen Bestellungen als Population betrachtet werden; eine Assoziation von Büchern beschreibt die Teilgruppe der Bestellungen, in denen sie vorkommt.*

Um festzustellen, ob ein bestimmtes Muster das Akzeptanzkriterium erfüllt, muß es mittels einer Datenbankanfrage mit der Datenbank abgeglichen werden. Automatische Verfahren zur Suche nach interessanten Mustern durchsuchen die vorgegebene Mustermenge systematisch nach akzeptablen Mustern und stellen die dazu nötigen Anfragen an die Datenbank. Der Aufwand, den die Suche nach interessanten Mustern verursacht, hängt ab von der Anzahl der benötigten Datenbankanfragen und von den Kosten der einzelnen Datenbankanfragen.

1.3.2 Vollständige Suche

Die Aufgabenstellung, alle interessanten Muster in der Datenbank zu entdecken, erfordert eine vollständige Suche durch die vorgegebene Mustermenge, die garantiert alle Muster entdeckt, die das Akzeptanzkriterium erfüllen. Im Unterschied dazu setzen unvollständige Suchverfahren sogenannte Heuristiken ein, die die Suche schneller auf vielversprechende Muster ausrichten, ohne den gesamten Musterraum zu untersuchen. Heuristiken sind auf Erfahrung oder Intuition beruhende Regeln, die gelegentlich zu falschen Entscheidungen führen können. Heuristische Suchverfahren verzichten also zugunsten der Suchgeschwindigkeit oft auf garantiert optimale oder vollständige Ergebnisse. Doch auch die vollständige Suche bietet Ansatzpunkte, den Suchaufwand zu begrenzen. Das Vorgehen, im voraus Bereiche des Suchraums zu identifizieren, die sicher keine akzeptablen Muster enthalten, ohne sie tatsächlich durchsuchen zu müssen, und diese Bereiche von der weiteren Suche auszuschließen, bezeichnet man als sichere Beschränkung der Suchraums.

1.3.3 Subsumtion

Um die Suche systematisch und effizient zu organisieren und den Suchraum sicher beschränken zu können, wird die Mustersprache durch Subsumtionsbeziehungen zwischen

den Mustern zu einem Musterraum geordnet. In individuenzentrierten Anwendungen, deren Untersuchungsgegenstand eine feststehende Population von Beispielen oder Individuen ist, bietet die extensionale Subsumtion eine naheliegende (Halb-)Ordnung für die Mustermenge. Danach subsumiert ein Muster M_1 ein anderes Muster M_2, wenn M_1 alle Individuen beschreibt, für die die Beschreibung M_2 zutrifft. Man sagt dann auch, das subsumierte Muster M_2 ist spezieller als das subsumierende Muster M_1, und M_1 heißt allgemeiner als M_2.

Beispiel 8 (zu Beispiel 1) *Das Buchversender-Beispiel beschreibt eine individuenzentrierte Anwendung mit einer Menge von Bestellungen als Untersuchungsgegenstand. Eine Bestellung ist eine Menge von Büchern. Ein Muster M wird ebenfalls als Menge von Büchern repräsentiert. Ein Muster M deckt eine Bestellung B ab, wenn jedes Buch von M in der Bestellung B vorkommt.*

Angenommen, der Buchversender interessiere sich dafür, häufige Assoziationen der drei Bücher Harry Potter and the Order the Phoenix, Bd.5 (HP5), Robinson Crusoe (RC) und Harry Potter and the Goblet of Fire, Bd.4 (HP4) in den Bestellungen zu entdecken. Mögliche Muster sind hier:

$$M_1 = \{\textit{HP5}\}$$
$$M_2 = \{\textit{RC}\}$$
$$M_3 = \{\textit{HP5, RC}\}$$

Das Muster M_1 subsumiert das Muster M_3, weil alle Bestellungen, in denen M_3 vorkommt, auch M_1 enthalten. Genauso subsumiert M_2 das Muster M_3.

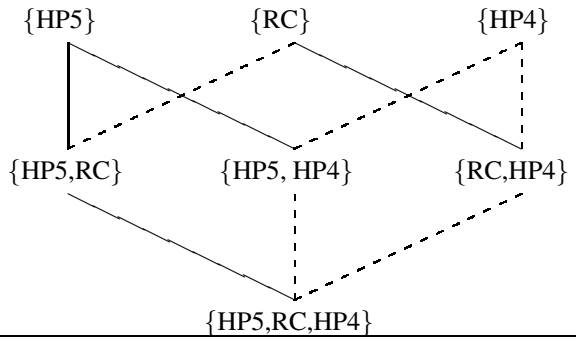
1.3.4 Spezialisierungsoperatoren

Im geordneten Musterraum kann die Suche systematisch von den allgemeinen Mustern in Richtung zu den spezielleren Mustern erfolgen, oder umgekehrt, von den spezielleren hin zu den allgemeineren Mustern. Seltener sind Suchalgorithmen, die die Suchrichtung wechseln. Viele Suchalgorithmen durchlaufen den Suchraum von den allgemeineren Mustern ausgehend in Richtung zu den spezielleren Mustern. Die Suche ist oft — vor allem in der ILP — implementiert als die Anwendung eines Spezialisierungsoperators, der allgemeinere Muster zu von diesen subsumierten spezielleren Mustern verfeinert. Mit den allgemeinsten Mustern beginnend, erzeugt die wiederholte, systematische Anwendung des Spezialisierungsoperators den gesamten Suchraum. Sie wird möglichst so gesteuert, daß jedes zur Sprache gehörende Muster genau einmal generiert wird, daß aber keine subsumtionsäquivalenten syntaktischen Varianten von bereits erzeugten Mustern entstehen. Um zu verhindern, daß ein Muster durch verschiedene Sequenzen von elementaren Spezialisierungsoperationen mehrfach generiert wird, legen viele Spezialisierungsoperatoren eine Reihenfolge fest, in der elementare Spezialisierungsoperationen aufeinander folgen müssen.

Beispiel 9 (zu Beispiel 1) *Bei der Suche nach Assoziationen in Bestellungen oder Warenkörben besteht eine Spezialisierungsoperation darin, ein Muster um einen weiteren Artikel zu vergrößern. Der Buchversender möchte häufige Assoziationen der drei Bücher Harry Potter, Bd.5 ($HP5$), Robinson Crusoe (RC) und Harry Potter, Bd.4 ($HP4$) in den Bestellungen entdecken. Um Redundanzen auszuschließen, dürfen die möglichen Spezialisierungsoperationen „Anfügen von HP5“, „Anfügen von RC“ und „Anfügen von HP4“, nur in dieser Reihenfolge ausgeführt werden. Der Suchraum dieser Anwendung ist in Abbildung 1.1 dargestellt. Die durchgezogenen Linien symbolisieren erlaubte Spezialisierungsoperationen. Die gestrichelten Linien stehen für Spezialisierungsoperationen, die die vorgeschriebene Reihenfolge verletzen und daher nicht erlaubt sind.*

Die allgemeinsten Muster im Suchraum sind $\{HP5\}$, $\{RC\}$ und $\{HP4\}$. Das Muster $HP5$ kann spezialisiert werden zu $\{HP5, RC\}$ und $\{HP5, HP4\}$. Das Muster $\{RC\}$

Abbildung 1.1 Spezialisierungsoperationen im Suchraum zum Beispiel 9.



kann spezialisiert werden zu $\{RC, HP4\}$, aber nicht zu $\{HP5, RC\}$, weil diese Spezialisierungsoperation die vorgeschriebene Reihenfolge nicht einhält.

1.3.5 Sichere Beschränkung des Suchraums

Die meisten Akzeptanzkriterien sind so realisiert, daß eine den Zielen der jeweiligen Data Mining-Aufgabe gemäße Interessantheitsfunktion die Interessantheit von Mustern mit einem numerischen Interessantheitswert bewertet. Ein Muster wird als interessant akzeptiert, wenn sein Interessantheitswert einen bestimmten Schwellwert erreicht oder überschreitet. Eine Interessantheitsfunktion ist monoton zur extensionalen Subsumtion, wenn sie für ein Muster M nie höhere Interessantheitswerte ergibt als für ein allgemeineres Muster M', das das Muster M extensional subsumiert.

Beispiel 10 (zu Beispiel 1) *Der Buchversender sucht Assoziationen von Büchern, die in mindestens 0.05% aller Bestellungen vorkommen. Das Muster $\{HP4\}$ kommt in 0.1% aller Bestellungen vor und gilt deshalb als interessant. Das Muster $\{RC\}$ kommt nur in 0.01% aller Bestellungen vor und gilt deshalb als nicht interessant. Das Muster $\{RC, HP4\}$ ist sicher nicht interessant, weil es in höchstens 0.01% aller Bestellungen vorkommen kann.*

Wenn die verwendete Interessantheitsfunktion monoton und die Suche als Anwendung eines Spezialisierungsoperators realisiert ist, läßt sich der Suchraum dadurch sicher beschränken, daß zu uninteressante Muster von weiteren Anwendungen des Spezialisierungsoperators ausgeschlossen werden. Dadurch werden nicht-akzeptable Muster aus dem Suchraum eliminiert, ohne daß ihre Interessantheit durch eine Datenbankanfrage ausgewertet werden muß.

1.4 Thema der Arbeit

Diese Arbeit möchte einen Beitrag zur technischen Seite des Data Mining leisten, indem sie weitergehende Methoden zur sicheren Beschränkung des Suchraums beim Finden interessanter Teilgruppen im Rahmen der individuenzentrierten ILP ausarbeitet und ihre Wirksamkeit in experimentellen Anwendungen evaluiert. Im einzelnen werden folgende Methoden betrachtet.

1.4.1 Optimumschätzfunktionen

Viele praktisch relevante Interessantheitsfunktionen sind nicht monoton zur extensionalen Subsumtion, zum Beispiel solche, die auf statistischen Tests beruhen. Für diese bieten Optimumschätzfunktionen einen Ansatzpunkt, den Suchraum dennoch sicher zu beschränken.

Optimumschätzfunktionen berechnen zu einem gegebenen Muster den maximalen Interessantheitswert, den davon subsumierte Muster im günstigsten Fall erreichen können. Wenn dieses Optimum unter dem geforderten Schwellwert liegt, brauchen die subsumierten Muster bei der weiteren Suche nicht berücksichtigt zu werden. In dieser Arbeit werden Optimumschätzfunktionen zu gegebenen Interessantheitsfunktionen entwickelt und ihr Beitrag zur Suchraumbeschränkung untersucht.

1.4.2 Der Apriori-Algorithmus und die Teilmengenbedingung

Das oben beschriebene Vorgehen zur sicheren Beschränkung des Suchraums, das auf dem Abbruch der weiteren Spezialisierung eines nicht-akzeptablen Musters M beruht, hat folgende Schwäche: Es entfernt nur solche von M subsumierten Muster aus dem Suchraum, die der Spezialisierungsoperator als Spezialisierungen von M erzeugt. Von M subsumierte Muster können im Suchraum verbleiben, wenn sie durch Sequenzen von Spezialisierungsoperationen entstehen, die M nicht enthalten.

Beispiel 11 (zu Beispiel 1) *Gegeben sei ein Suchraum, der mit Sequenzen von Spezialisierungsoperationen wie in Abbildung 1.1 aufgebaut werden soll. Das Muster $\{RC\}$ sei nicht ausreichend häufig. Es wird daher nicht weiter spezialisiert, so daß seine Spezialisierung $\{RC, HP4\}$ nicht erzeugt werden kann.*

Das Muster $\{HP5, RC, HP4\}$, das ebenfalls von $\{RC\}$ subsumiert wird und folglich ebenfalls sicher nicht akzeptabel ist, verbleibt im Suchraum, wenn es über eine Spezialisierungssequenz erzeugt wird, die $\{RC\}$ nicht beinhaltet.

Der Apriori-Algorithmus von [AMS+96, MTV94], ist ein sehr bekannter und vielfach variierter Algorithmus zum Entdecken häufiger Assoziationen in Warenkorb-Datenbanken wie in Beispiel 1. Er nützt auf elegante Weise alle Subsumtionsbeziehungen zwischen Mustern zur Beschränkung des Suchraums.

Bei der Suche nach häufigen Assoziationen in Warenkorb-Datenbanken umfaßt die Mustersprache die möglichen Assoziationen von Artikeln, die in der Warenkorb-Datenbank vorkommen können. Die Assoziationen sind die Teilmengen einer festen Menge von Artikeln; alle Teilmengen der Artikelmenge gehören zur Mustersprache. Die Teilmengenbeziehung zwischen zwei Artikelmengen bietet ein syntaktisches Kriterium für extensionale Subsumtion zwischen Artikelmengen: wenn eine Artikelmenge M_1 eine Teilmenge der Artikelmenge M_2 ist, so ist M_1 allgemeiner als M_2 bezüglich der extensionalen Subsumtion. Daraus folgt, daß eine Artikelmenge sicher nicht häufig sein kann, wenn sie eine nichthäufige Teilmenge besitzt. Dieser Zusammenhang läßt sich umkehren zu der Bedingung, daß eine Artikelmenge nur dann häufig sein kann, wenn alle ihre Teilmengen häufig sind. Auf dieser Bedingung, die im folgenden als Teilmengenbedingung bezeichnet wird, basiert die Beschränkung des Suchraums im Apriori-Algorithmus. Der Algorithmus durchsucht die Mustersprache vom Allgemeinen zum Speziellen. Die Suche ist realisiert als die Anwendung eines Spezialisierungsoperators. Sie erfolgt ebenenweise; das bedeutet hier, daß Artikelmengen mit i Artikeln erst erzeugt und ausgewertet werden, wenn alle Artikelmengen mit $i - 1$ Artikeln bearbeitet sind. Der Algorithmus speichert alle häufigen Artikelmengen in einer Menge S. Bei der Generierung jeder neuen Artikelmenge überprüft der Algorithmus, ob sie die Teilmengenbedingung erfüllt, das heißt, ob alle ihre Teilmengen in S vorkommen. Nur dann wird die neue Artikelmenge ausgewertet. Wenn sie die Teilmengenbedingung nicht erfüllt, kann sie ohne Auswertung verworfen werden. Sie wird dann nicht in S gespeichert und nicht weiter spezialisiert.

Ein Schwerpunkt dieser Arbeit liegt darin, die ebenenweise Suche und die Beschränkung des Suchraums anhand der Teilmengenbedingung des Apriori-Algorithmus auf die komplexeren Mustersprachen der ILP zu übertragen. Die Muster einer ILP-Sprache sind meist Disjunktionen von Literalen, die Variablen enthalten können. Die Literaldisjunktionen, genannt Klauseln, werden als Mengen von Literalen dargestellt. Als Ordnung einer

ILP-Sprache ist die θ-Subsumtion gebräuchlich. Nach der θ-Subsumtion stehen zwei Klauseln in einem Subsumtionsverhältnis, wenn die eine Klausel durch Umbenennung ihrer Variablen in eine Teilmenge der anderen Klausel überführt werden kann. Weil verschiedene Variablenumbenennungen möglich sind, ist es im allgemeinen sehr aufwendig, für zwei gegebene Klauseln festzustellen, ob die eine Klausel die andere θ-subsumiert. Entsprechend aufwendig ist es, bei der Suche vom Allgemeinen zum Speziellen die θ-Subsumtionsbeziehungen zwischen Mustern zu berechnen, um den Suchraum beschränken zu können. Der dazu erforderliche Aufwand kann in manchen Fällen die Einsparungen aufwiegen, die durch die Beschränkung des Suchraums erzielt werden.

Die Adaption der Teilmengenbedingung für ILP-Sprachen ist deshalb schwierig, weil in einer ILP-Sprache nicht mehr alle Teilmengen einer Literalmenge sinnvoll zu untersuchende Muster darstellen. Ein Muster ist nicht einfach durch Anfügen eines beliebigen Literals aus einer vorgegebenen Literalmenge weiter zu spezialisieren, weil sich dadurch redundante oder in sich widersprüchliche Muster ergeben können, im allgemeinen so viele, daß der Suchraum „explodiert“. Um die ebenenweise Suche und die Teilmengenbedingung des Apriori-Algorithmus dennoch für die Suche nach interessanten Mustern im relationalen Data Mining übernehmen zu können, wird in dieser Arbeit das Apriori-Suchverfahren erweitert und ein dazu passender Formalismus zur Deklaration einer ILP-Hypothesensprache entwickelt. Der Formalismus ist auch für die Deklaration von SQL-Hypothesensprachen geeignet und ermöglicht somit Data Mining direkt auf relationalen SQL-Datenbanken. Ansätze und Formalismen zur Sprachdeklaration werden auch als deklarativer Sprachbias (*Bias*, englisch, bedeutet Vorurteil, Voreinstellung) bezeichnet, da sie es dem Anwender erlauben, die Hypothesensprache für eine gegebene Lern- und Data Mining-Anwendung individuell anzupassen und eine Obermenge der dabei erzielbaren Resultate vorzubestimmen. Während in aussagenlogischen Lern- und Data Mining-Anwendungen die Hypothesensprache häufig durch den eingesetzten Lernalgorithmus weitgehend festgelegt ist, kommt in der ILP, wegen der ausdrucksmächtigeren und umfangreicheren Hypothesensprachen, dem deklarativen Sprachbias große Bedeutung zu [NRA+96, Tau94].

1.4.3 Taxonomien

Assoziationsregeln in Warenkörben müssen sich nicht auf die Ebene individueller Artikel beschränken, sondern können auch Zusammenhänge auf einer allgemeineren Ebene beschreiben, wenn mehrere Artikel zu einem verallgemeinerten Artikel zusammengefaßt werden.

Beispiel 12 (zu Beispiel 1) *Um Assoziationsregeln finden zu können, die Zusammenhänge auf der Ebene von Autoren beschreiben, werden alle Bücher desselben Autors „Autor“ zu einem verallgemeinerten Artikel „Buch von Autor“ zusammengefaßt. Damit lassen sich Assoziationsregeln wie die folgende entdecken:*

> *Viele der Kunden, die ein Buch von J. K. Rowling bestellt haben, haben gleichzeitig auch ein Buch von Lemony Snicket, ein Buch von Philip Pullman, ein Buch von Amy Tan und ein Buch von Stephen King bestellt.*

Solche verallgemeinerten Assoziationsregeln werden aus häufigen Assoziationen von verallgemeinerten und einfachen Artikeln abgeleitet. Ein Spezialisierungsoperator kann eine Artikelmenge, die einen verallgemeinerten Artikel enthält, nicht nur durch Anfügen eines weiteren Artikels spezialisieren, sondern auch dadurch, daß er den verallgemeinerten Artikel durch einen von ihm subsumierten einfachen Artikel ersetzt. Die resultierende Artikelmenge ist von der ursprünglichen Artikelmenge extensional subsumiert.

Beispiel 13 *Der verallgemeinerte Artikel „Buch von Autor* J. K. Rowling*“ (Row) umfaßt die einfachen Artikel* Harry Potter and the Order of the Phoenix, Bd.5 *(HP5) und* Harry Potter and the Goblet of Fire, Bd.4 $(HP4)$.

Das Muster $\{Row\}$ kann spezialisiert werden zu $\{HP5\}$ und zu $\{HP4\}$. Das Muster $\{Row, RC\}$ (mit der Bedeutung „ein Buch von J. K. Rowling und Robinson Crusoe“) kann spezialisiert werden zu $\{HP5, RC\}$ und zu $\{HP4, RC\}$ (mit der Bedeutung „Harry Potter, Bd.5 und Robinson Crusoe“ beziehungsweise „Harry Potter, Bd.4 und Robinson Crusoe“).

Allgemeinerbeziehungen wie die zwischen einfachen und verallgemeinerten Artikeln in der Warenkorbanalyse können auch im relationalen Data Mining bestehen. Sie ergeben sich beispielsweise aus Taxonomien, die auf den Wertebereichen von Attributen definiert sind, oder lassen sich aus dem Vorwissen über eine Anwendung ableiten. Subsumtionsbeziehungen zwischen einfachen und verallgemeinerten Artikeln beziehungsweise ihre Entsprechungen in einer ILP-Hypothesensprache lassen sich zur Beschränkung des Suchraums nutzen. Diese Arbeit beschreibt eine Methode, die Beschränkung des Suchraums aufgrund solcher Subsumtionsbeziehungen zu integrieren mit der Suchraumbeschränkung durch die Teilmengenbedingung.

1.4.4 Ausschluß akzeptierter Muster

Um in Anwendungen, in denen mit sehr vielen akzeptierten Mustern zu rechnen ist, die Zahl akzeptierter Muster zu reduzieren, können Muster, die von akzeptierten Mustern subsumiert sind, aus dem Suchraum ausgeschlossen werden. Akzeptierte Muster werden dann nicht weiter spezialisiert. Dieses Vorgehen verkleinert auch den Suchraum; da es aber die Menge akzeptierter Muster verändert, leistet es keine sichere Suchraumbeschränkung. Im Zusammenhang dieser Arbeit ist es interessant, die Verkleinerung des Suchraums, die der Ausschluß akzeptierter Muster von der weiteren Spezialisierung bewirkt, mit der Wirkung der Methoden zur sicheren Suchraumbeschränkung zu vergleichen.

1.4.5 Experimentelle Untersuchungen

Um die Nützlichkeit der verschiedenen Methoden zur Beschränkung des Suchraums auch praktisch zu untersuchen, wurden Experimente durchgeführt. Diese bilden einen weiteren Schwerpunkt der Arbeit.

1.5 Überblick über die Arbeit

Kapitel 2 positioniert das Thema und die Beiträge dieser Arbeit in der KDD und der ILP. Es formalisiert die Suche nach interessanten Mustern im Rahmen der individuenzentrierten nicht-monotonen ILP und führt den Suchalgorithmus ein, der im Verlauf der Arbeit weiterentwickelt wird. Kapitel 3 stellt verschiedene gebräuchliche Interessantheitskriterien vor, für die zum Teil Optimumschätzfunktionen hergeleitet werden. Kapitel 4 entwickelt einen Suchalgorithmus und einen Sprachbias, die es ermöglichen, einen ILP-Suchraum mittels der Teilmengenbedingung zu beschränken. Anwendungsbeispiele demonstrieren die Verwendung und Wirkung des entwickelten Ansatzes wie auch seine Grenzen. Das Kapitel enthält außerdem eine ausführliche Diskussion verwandter Arbeiten. Kapitel 5 arbeitet aus, wie sich die Beschränkung des Suchraums anhand von Subsumtionsbeziehungen zwischen Artikeln (in der Warenkorbanalyse) beziehungsweise Literalen (in der ILP) in den bisher entwickelten Suchalgorithmus integrieren läßt. Die Subsumtionsbeziehungen können sich zum Beispiel aus Taxonomien oder durch Diskretisierung numerischer Attribute ergeben. Auch dieses Kapitel bietet Anwendungsbeispiele und diskutiert verwandte Ansätze. Kapitel 6 berichtet über die Experimente, in denen die Wirksamkeit der verschiedenen Methoden zur Suchraumbeschränkung evaluiert wird. Dazu werden verschiedene beispielhafte Data Mining-Aufgaben spezifiziert und mit einer prototypischen Implementation des entwickelten Algorithmus in mehreren Suchläufen bearbeitet, in denen die eingesetzten

Methoden zur Suchraumbeschränkung variieren. Um die Tauglichkeit und Angemessenheit des entwickelten Ansatzes praktisch zu testen, wurde er im Rahmen eines Discovery Challenge auf ein echtes Data Mining-Problem angewendet. Darüber berichtet Kapitel 7. Es enthält außerdem einen Vergleich mit anderen Beiträgen zum Discovery Challenge. Die Arbeit endet mit abschließenden Bemerkungen in Kapitel 8. Teile der Arbeit wurden bereits in [Web02, Web00, Web99b, Web99a, Web98b, Web98a, Web97] veröffentlicht.

Kapitel 2

Grundlagen

Im Mittelpunkt dieser Arbeit steht mit dem Finden interessanter Muster in Datenbanken eine Aufgabenstellung der Knowledge Discovery in Data Bases (KDD); ihren formalen Rahmen bildet die Induktive Logikprogrammierung (ILP). Dieses Kapitel gibt einen Überblick über KDD und ILP und führt Grundlagen aus Logik, Logikprogrammierung und Datenbanken ein. Schließlich formalisiert es die Suche nach interessanten Mustern im Rahmen der individuenzentrierten ILP und skizziert einen grundlegenden Suchalgorithmus.

2.1 Knowledge Discovery in Databases und Data Mining

Das Forschungsgebiet Knowledge Discovery in Databases und Data Mining ist gekennzeichnet durch seine Interdisziplinarität. Zu den beitragenden Gebieten gehören Maschinelles Lernen, Statistik, Datenbanktheorie, Mustererkennung, Visualisierung, Information Retrieval, Data Warehousing und Online Analytical Processing, verteiltes und paralleles Rechnen und weitere. Der folgende Abschnitt listet einige Punkte auf, die für die KDD typisch sind und sie von den benachbarten Gebieten, insbesondere dem Maschinellen Lernen, unterscheiden. Er orientiert sich an [Kod99].

2.1.1 Charakterisierung und Abgrenzung von KDD und Data Mining

Die weithin anerkannte Definition beschreibt Knowledge Discovery in Databases als den Prozeß, (statistisch) gültige, neue, möglicherweise nützliche und letztendlich verständliche Struktur in Daten zu identifizieren. In der Zielsetzung, neues Wissen zu entdecken, unterscheidet sich KDD von Maschinellem Lernen und Statistik. Während letztere traditionell Wissen suchen, das bereits Bekanntem nicht widerspricht, können in der KDD Entdeckungen, die dem bisher vorliegenden Wissen widersprechen, ganz besonders interessant sein [Kod99]. Ein typisches Problem, das in der KDD zu lösen ist, besteht darin, daß die Data Mining-Aktivitäten häufig bereits bekannte Muster und Regelmäßigkeiten in großer Zahl in den Daten wiederentdecken.

Charakteristisch für die KDD ist die explizite Orientierung darauf, anwendbares und umsetzbares Wissen zu entdecken. Das entdeckte Wissen soll Handlungen und Entscheidungen des KDD-Anwenders beeinflussen und damit schließlich Veränderungen in der realen Welt bewirken können [Fei98, Kod99]. Diese praxis-orientierte Sicht bringt die erweiterte Definition von KDD und Data Mining von Feist zum Ausdruck:

> „Data Mining ist der Prozeß, automatisch vorher unbekannte, statistisch korrekte, interessante und interpretierbare Zusammenhänge in großen Datenmengen zu identifizieren *und diese für wichtige Unternehmensentscheidungen zu verwenden*“. [Fei98]

Dieser Anspruch bringt mit sich, daß in der KDD die Interessantheit von Entdeckungen subjektiv und relativ zu den Zielen der jeweiligen Anwendung beurteilt werden muß. Die Forschung im Maschinellen Lernen hat sich bisher (mit wenigen Ausnahmen) darauf konzentriert, universelle Methoden zu entwickeln, um den Wissenserwerb für alle möglichen Anwendungen in eine sinnvolle Richtung zu lenken. Die KDD befaßt sich dagegen verstärkt mit der Entwicklung und Erforschung von Interessantheitsmaßen und Bewertungskriterien, die subjektive und individuelle Zielsetzungen und Voraussetzungen, wie etwa bereits vorhandenes Wissen, berücksichtigen [Mül99, ST96]. Die Bedeutung sogenannter objektiver, formaler Gütekriterien wie Allgemeingültigkeit, Präzision, Beweisbarkeit oder Vorhersagegenauigkeit tritt zurück gegenüber subjektiven Kriterien wie Interpretierbarkeit und Umsetzbarkeit. Andererseits verursacht die Umsetzung von KDD-Ergebnissen auch Kosten und birgt Risiken. Gerade wenn besonders überraschende und unerwartete Zusammenhänge entdeckt wurden, unterstützen erfüllte formale und objektive Gütekriterien die Bereitschaft der Anwender, die gewonnenen Erkenntnisse zu akzeptieren und umzusetzen.

Die Definition der KDD betont auch, daß der Erwerb von Wissen aus Daten ein komplexer Prozeß ist. Dieser erschöpft sich nicht in der Anwendung eines einzelnen Wissensextraktionsalgorithmus, sondern erfordert die Kombination und Integration vielfältiger Lösungsmethoden und Verfahren [Kod99]. Dementsprechend sind heute schon große KDD-Softwaresysteme verfügbar, die verschiedene Phasen eines KDD-Prozesses unterstützen und jeweils mehrere Verfahren und Algorithmen zur Lösung der anfallenden Teilaufgaben anbieten, beispielsweise Clementine von SPSS, EnterpriseMiner von SAS und andere. Die KDD-Systeme bieten die verfügbaren Verfahren auf einer graphischen Oberfläche an und erlauben, diese zu selektieren und so hintereinander zu schalten, daß die Ergebnisse der einen Systemkomponente die Eingabe der nächsten Komponente bilden. Dieser interaktive und explorative Umgang mit den Daten geht über die bloße Anwendung eigenständiger, üblicherweise im Stapelbetrieb laufender Modellierungsverfahren hinaus [Kod99].

2.1.2 Der Prozeß der Wissensentdeckung

In neuerer Zeit wurden verschiedene Prozeßmodelle entwickelt und vorgestellt, die die einzelnen Phasen, aus denen ein KDD-Prozeß in der praktischen Ausführung besteht, unabhängig von einer bestimmten Anwendung beschreiben. Damit bieten sie einen Überblick über die verschiedenen Teilaufgaben, die in einer Data Mining-Anwendung zu lösen sind. Die Prozeßmodelle unterscheiden sich in ihrem Detaillierungsgrad und in ihren Schwerpunkten und auch darin, welche Teilaufgaben sie als Bestandteile des Data Mining-Prozesses betrachten. Sie stimmen jedoch weitgehend darin überein, daß sich die Reihenfolge, in der die einzelnen Phasen und Teilaufgaben aufeinanderfolgen, nicht fest vorgeben läßt, sondern daß im Verlauf der meisten Anwendungen wiederholte Durchgänge durch einzelne Phasen oder durch Teilaufgaben innerhalb der Phasen nötig werden können, genauso auch Rück- und Vorsprünge zwischen Phasen und Teilaufgaben [Wel99].

Ein bekanntes Prozeßmodell ist das CRISP-DM (Cross Industry Standard Process Modell for Data Mining), das aus der Zusammenarbeit führender industrieller Data Mining-Anwender und -Entwickler hervorgegangen ist und in Arbeitsgruppen und Projekten weiterentwickelt wurde [CCK+00]. CRISP-DM nennt sechs Phasen eines typischen Data Mining–Projekts, innerhalb derer verschiedene generische Aufgaben auszuführen und bestimmte Zwischenergebnisse zu erzielen sind.

1. Verständnis für das Anwendungsgebiet enwickeln: Die Ziele und Erfolgskriterien des Projekt-Auftraggebers aus der Sicht der Anwendung werden in technische Ziele und Erfolgskriterien aus Data Mining-Sicht übersetzt. Ein entsprechender vorläufiger Projektplan wird erstellt.

2. Verständnis für die Daten entwickeln: Die für das Data Mining-Projekt vorgesehenen

Daten werden gesammelt und inspiziert, um ihren Umfang und Gehalt sowie ihre Qualität festzustellen.

3. Vorbereitung der Daten: Die Datenbasis wird aufbereitet und in die für die Modellierungsphase nötige Form gebracht.

4. Modellierung: Wissensextraktions- oder Modellbildungsalgorithmen werden auf die vorbereiteten Daten angewendet. Hier findet also das eigentliche Data Mining statt.

5. Bewertung der Ergebnisse: Die bisher erzielten werden mit den angestrebten Ergebnissen verglichen. Dann wird das weitere Vorgehen geplant. Dies kann in der Umsetzung der Ergebnisse bestehen, im erneuten Durchlauf durch einzelne oder mehrere Phasen des KDD-Prozesses oder auch im Start eines neues Data Mining-Projekts.

6. Umsetzung der Ergebnisse: Die praktische Anwendung der Projektresultate wird geplant, ebenso die Überwachung dieser Anwendung und die Pflege der Projektresultate.

2.1.3 Anwendungsziele des Data Mining

Auf der allgemeinsten Ebene unterscheiden die meisten Autoren zwei bedeutende Typen von Data Mining-Problemen [Han99, FPSS96]. Erstens die Klassifikation oder Vorhersage noch unklassifizierter oder zukünftiger Fälle anhand von vorklassifiziertem Datenmaterial — dies ist der vorherrschende Problemtyp —, und zweitens die Beschreibung eines Anwendungsgebiets. Darunter versteht man die Bestimmung möglichst aller wichtigen Zusammenhänge und gegenseitigen Abhängigkeiten zwischen den Variablen eines durch Daten abgebildeten Weltbereichs. Klösgen nennt als dritten, genauso bedeutenden Problemtyp das Finden sogenannter *Nuggets* (Goldklumpen) [Klö99]. Die Nuggets sind einzelne Hypothesen über Zusammenhänge oder Regelmäßigkeiten in Daten. Diese Hypothesen sind voneinander unabhängig und unvollständig und erlauben weder die Bestimmung aller unbekannten Fälle noch erfassen sie alle im Datenmaterial existierenden Zusammenhänge. Als Resultat einer Data Mining-Anwendung sind sie dann zufriedenstellend, wenn das vorhandene Datenmaterial keine weiterreichenden Schlüsse zuläßt, zum Beispiel weil relevante Attribute fehlen, oder wenn die zur Beschreibung der Hypothesen gewählte Sprache nicht ausreicht. Solche partiellen Ergebnisse können einen Ausgangspunkt für weitere und detailliertere Analysen bilden.

Diese grobe Einteilung wird weiter verfeinert zu einer Liste möglicher Ziele von Data Mining-Anwendungen. Bei der Auflistung und Definition der möglichen Zielsetzungen stimmen verschiedene Publikationen nicht immer überein. So nennen zum Beispiel das Crisp-Papier [CCK+00], ein Enzyklopädie-Beitrag von J. Han [Han99] und das Einführungskapitel [FPSS96] eines bekannten KDD-Buchs [FPSSU96] jeweils sechs speziellere Data Mining-Anwendungsziele, die sie aber unterschiedlich einteilen und voneinander abgrenzen.

Klassifikation [CCK+99, Han99, FPSS96]. Dies ist die wichtigste, am besten erforschte und am genauesten definierte Data Mining-Zielsetzung. Dabei ist eine Menge von Objekten gegeben, die durch bestimmte Merkmale beschrieben sind. Jedes Objekt ist genau einer Klasse aus einer diskreten, endlichen Menge von Klassen zugeordnet. Das Ziel ist, ein Klassifikationsmodell oder einen Klassifikator zu erstellen, der bisher unbekannten oder nicht klassifizierten Objekten die korrekte Klasse zuweist.

Vorhersage (Prediction [Han99, FPSS96], Regression [CCK+99]). Diese Zielsetzung ist sehr ähnlich zur Klassifikation. Der Unterschied besteht darin, daß nicht eine qualitative, diskrete Klassenzugehörigkeit, sondern ein Wert eines kontinuierlichen Zielattributs vorhergesagt werden soll.

Unterteilung (Clustering [Han99, FPSS96], Segmentation [CCK+99]). Darunter versteht man die manuelle, halb- oder vollautomatische Unterteilung einer Menge von Datenobjekten in verschiedene Klassen. Im Unterschied zur Klassifikation sind hier die Klassen nicht vorgegeben, sondern müssen durch Auswertung der Daten so bestimmt werden, daß die zum selben Segment gehörenden Datenobjekte zueinander ähnlich sind und sich von den Datenobjekten in den anderen Segmenten unterscheiden.

Aggregation [FPSS96], Datenbeschreibung und Datenzusammenfassung [CCK+99], Klassenbeschreibung [Han99]. Hier ist eine kompakte Beschreibung der Daten oder von Teilmengen der Daten auf niedriger Ebene verlangt. Die Beschreibung soll einen Überblick über die Struktur der Daten geben, etwa in Form von Häufigkeiten, Summen und Durchschnitten, Maximal- und Minimalwerten, Varianzen und Verteilungen einzelner Attributwerte.

Zeitreihenanalyse [Han99]. Die Zeitreihenanalyse forscht in Daten, die mit Zeit- oder Reihenfolgenmarkierungen versehen sind, nach häufigen sequentiellen Mustern, Periodizitäten, Trends und Abweichungen. Das Crisp-Manual subsumiert die Suche nach sequentiellen Mustern unter die Abhängigkeitsanalyse. [FPSS96] fassen die Suche nach Abweichungen in Zeitreihendaten unter das Entdecken von Abweichungen in allgemeinen Daten.

Abhängigkeitsanalyse, Dependency analysis [FPSS96, CCK+99],Association [Han99]. Bei der Abhängigkeitsananlyse geht es darum, ein Modell zu finden, das die wichtigsten Abhängigkeiten und Korrelationen zwischen Datenobjekten oder Ereignissen darstellt, und zwar sowohl strukturell als auch quantitativ. Die Abhängigkeiten lassen sich zwar zur Vorhersage unbekannter Merkmale verwenden, sollen aber meist vor allem zum besseren Verständnis des Anwendungsgebiets beitragen. Als prominente Spezialfälle dieser Data Mining-Zielsetzung nennt das Crisp-Manual das Finden von Assoziationen und Assoziationsregeln. Der Enzyklopädie-Beitrag von Han [Han99] führt das Finden von Assoziationen und Assoziationsregeln als eigenständige Zielsetzung auf.

Entdeckung von Abweichungen und Änderungen [CCK+99, FPSS96]. Das Ziel ist hier, Abweichungen von zeitlich früheren oder Standard-Werten sowie Änderungen von Standard-Werten zu erkennen. Das Entdecken von Ausreißern wird ebenfalls zu dieser Zielsetzung gezählt. Das Crisp-Modell [CCK+00] betrachtet das Entdecken von Abweichungen und Änderungen als zur Datenbeschreibung und -zusammenfassung gehörig.

Konzeptbeschreibung [CCK+99]. Das gewünschte Data Mining-Ergebnis ist hier eine verständliche Beschreibung von Konzepten oder Klassen von Datenobjekten. Konzeptbeschreibung ist verwandt zur Klassifikation und zur Segmentation. Während das Ergebnis der Segmentation eine bloße Auflistung der Segmente und ihrer Elemente sein kann, soll die Konzeptbeschreibung eine verständliche Beschreibung der gemeinsamen Eigenschaften eines Konzepts liefern. Konzeptbeschreibungen können zur Klassifikation neuer Objekte dienen, während andererseits von Klassifikationsalgorithmen erstellte Modelle als Konzeptbeschreibungen betrachtet werden können, sofern sie in verständlicher Form vorliegen. Der Unterschied besteht darin, daß Klassifikationsmodelle vollständig sein und für alle möglicherweise auftretenden Fälle eine Entscheidung bezüglich ihrer Klassenzugehörigkeit liefern müssen, während von Konzeptbeschreibungen diese Vollständigkeit nicht gefordert ist.

2.1.4 Ausgewählte Aufgabenstellungen

Während der vorangegangene Abschnitt 2.1.3 einen Überblick über mögliche Anwendungsziele des Data Mining gibt, beschreibt dieser Abschnitt einige formale Aufgaben-

Tafel 2.1 Konzeptlernen [Fla95]

Gegeben:	eine Sprache zur Beschreibung von Individuen L_e
	eine Sprache zur Beschreibung von Konzepten L_h
	ein Abdeckungs-Prädikat $match$, das Individuen und Konzepte abgleicht
	eine Menge $E^+ \cup E^-$ von positiven beziehungsweise negativen Beispielen für ein Zielkonzept
Gesucht:	vollständige und korrekte Beschreibungen h von Konzepten, die alle positiven Beispiele abdecken:
	$\forall e^+ \in E^+$: $match(h, e^+)$ (Vollständigkeit)
	und kein negatives Beispiel abdecken:
	$\forall e^- \in E^-$: $\neg match(h, e^-)$. (Korrektheit)

stellungen von Maschinellem Lernen und Data Mining, die für das Thema dieser Arbeit relevant sind.

Konzeptlernen

Das Konzeptlernen aus Beispielen ist eine sehr grundlegende Aufgabenstellung des Maschinellen Lernens. Eine formale Beschreibung zeigt Tafel 2.1. Im Zusammenhang mit Konzeptlernen aus Beispielen versteht man unter einem Konzept die Beschreibung einer bestimmten Teilmenge aus einer Population von Fällen oder Individuen. Ein Beispiel ist ein Individuum mit einer Markierung, die angibt, ob das Individuum zum Konzept gehört oder nicht. Ein Beispiel, das zum Konzept gehört, ist ein positives Beispiel; ein Beispiel, das nicht zum Konzept gehört, ist ein negatives Beispiel für das Konzept. Die Extension eines Konzepts ist die Menge aller zum Konzept gehörenden Individuen. Eine intensionale Beschreibung eines Konzepts gibt Eigenschaften an, in denen sich die zum Konzept gehörigen Individuen von anderen Individuen unterscheiden. Die Aufgabe des Konzeptlernens besteht darin, aus einer Menge positiver und einer Menge negativer Beispiele eine vollständige und korrekte intensionale Beschreibung des Konzepts zu induzieren. Da intensionale Konzeptbeschreibungen durch einen induktiven, also nicht unbedingt wahrheitserhaltenden Schluß zustandekommen, bezeichnet man sie als Hypothesen. Die Übereinstimmung zwischen Beispielen und Hypothesen ist formal definiert als ein Prädikat $match$, das Paare von Hypothesen und Individuen auf die Werte $wahr$ und $falsch$ abbildet. Das Prädikat ist $wahr$ für eine Hypothese und ein Beispiel, wenn die Hypothese das Beispiel abdeckt. Die genaue Definition des Prädikats $match$ hängt von der jeweiligen Anwendung und der Art der Hypothesen ab. Konzeptlernen (nicht zu verwechseln mit dem Anwendungsziel „Konzeptbeschreibung“ in Abschnitt 2.1.3) ist eine Klassifikationsaufgabe, bei der es nur die beiden Klassen „positiv“ und „negativ“ gibt.

Teilgruppenanalyse

Eine typische, allgemein gefaßte Aufgabenstellung des Data Mining ist die Teilgruppenanalyse. Die Teilgruppenanalyse sucht solche Teilgruppen in einer Population von Analyseobjekten (das sind Fälle, Ereignisse, Beispiele oder ähnliches), die im Vergleich zur Grundpopulation ein interessantes Verhalten aufweisen [Klö99]. Interessantheit kann dabei etwa bedeuten, daß das Verhalten der Teilgruppe in problemrelevanten Aspekten stark von der Gesamtpopulation abweicht. Zur Bewertung der Interessantheit von Teilgruppen dient eine Interessantheitsfunktion q, die jeder Teilgruppe einen numerischen Wert zuweist. Die Aufgabenstellung der Teilgruppenanalyse existiert in zwei Varianten. Die erste Variante gibt einen Schwellwert für die Interessantheit von Teilgruppen vor; das Ziel ist, in der Menge möglicher Teilgruppen L_h alle Teilgruppen zu bestimmen, deren Interessantheit den Schwellwert erreicht oder überschreitet. Die zweite Variante gibt einen Parameter k

Tafel 2.2 Teilgruppenanalyse nach [WMJ00].

Gegeben:	eine Population Pop von Individuen
	eine Sprache L_h zur Beschreibung von Teilgruppen von Pop
	eine Interessantheitsfunktion $d: (L_h, Pop) \rightarrow \mathbb{R}$
	eine natürliche Zahl k oder
	ein Schwellwert q_{min} für die Interessantheit
Gesucht:	die Menge $H \subseteq L_h$ der interessanten Teilgruppen mit
	$H = \{h \in L_h \mid q(h) \geq q_{min}\}$
oder	die Menge $H \subseteq L_h$ der k interessantesten Teilgruppen mit
	$H \subseteq L_h, \lvert H \rvert = k$ und es gibt keine $h \in H, h' \in L_h$, für die $q(h') > q(h)$

Tafel 2.3 Finden aller potentiell interessanten Sätze [MT97a, MT97b].

Gegeben:	eine Datenbank D
	eine Sprache L_h von Sätzen
	ein Akzeptanzprädikat *accept* $: (D, L_h) \rightarrow \{wahr, falsch\}$
Gesucht:	die Menge H der potentiell interessanten Sätze über D, für die gilt:
	$H(L_h, D, accept) = \{h \in L_h \mid accept(h, D) = wahr\}$

vor; das Ziel ist, die k interessantesten Teilgruppen in L_h zu identifizieren, also die k Teilgruppen mit dem höchsten Interessantheitswert. Tafel 2.2 gibt eine formale Beschreibung der Aufgabenstellung angelehnt an die Definition von [WMJ00].

Beispiel 14 *Eine mögliche Anwendung der Teilgruppenanalyse ist die Untersuchung von Kundendaten einer Bank, um Gruppen von Kunden zu identifizieren, die ein bestimmtes Angebot der Bank auffällig wenig nachfragen, mit dem Ziel, das Angebot für diese Kundengruppe zu verbessern. Eine mögliche interessante Teilgruppe wäre beispielsweise:*

> *Ältere Personen in einer bestimmtem Region benutzen Kreditkarten wesentlich seltener als der Durchschnitt aller Kunden.*

Die Teilgruppenanalyse ist eine deskriptive Aufgabenstellung. Sie versucht nicht, ein globales Modell des gesamten Anwendungsbereichs zu erzeugen, sondern entdeckt lokale Muster, die nur für einen bestimmten Ausschnitt der gesamten Datenmenge gelten. Teilgruppenanalyse ist besonders geeignet für Anwendungsbereiche, in denen vielfältige, inhomogene Informationen vorliegen, in denen die Beschreibungen verschiedener Teilgruppen der Population stark voneinander abweichen können, oder für Anwendungsbereiche, in denen große Unterschiede im Verhalten verschiedener Teilgruppen auftreten [Klö99]. Für die Teilgruppenanalyse war mit Explora schon sehr früh ein Data Mining-System verfügbar [HK91, Klö96].

Finden aller interessanten Sätze

Eine weitere allgemein formulierte Aufgabenstellung in KDD und Data Mining ist das Finden aller potentiell interessanten Sätze in Daten. Tafel 2.3 zeigt die Definition dieser Aufgabenstellung nach Mannila und Toivonen [MT97a, MT97b]. Eine konkrete Anwendung dieser Aufgabenstellung ist bestimmt durch die Daten, die Hypothesensprache und das Akzeptanzkriterium, das über die Interessantheit von Sätzen entscheidet. Sätze, die das gegebene Akzeptanzkriterium erfüllen, heißen akzeptable Sätze. Die Lösung der Aufgabe ist eine Teilmenge der Hypothesensprache, die genau die akzeptablen Sätze enthält. Da die Lösung alle akzeptablen Sätze enthalten muß, ist eine vollständige Suche im Hypothesenraum erforderlich. Darin unterscheidet sich das Finden interessanter Sätze zum Beispiel

vom Klassifikatorenlernen, bei dem meist eine nicht unbedingt optimale Lösung genügt, die mit einer heuristischen, also unvollständigen Suche gefunden werden kann.

Typisch für das Finden interessanter Sätze ist, daß es oft sehr große Lösungsmengen liefert, die überwiegend aus für den Anwender uninteressanten und nutzlosen Sätzen bestehen. Der Grund dafür ist, daß Interessantheit und Nützlichkeit subjektive, von der Anwendung und vom Anwender abhängende Kriterien sind, die nur unzureichend in objektive Kriterien gefaßt werden können. Als objektiv gelten Kriterien, die nur von der Syntax der Sätze und den Daten abhängen [ST96]. Das Finden aller potentiell interessanten Sätze kann daher meist nur eine Vorauswahl leisten, aus der der Anwender die tatsächlich nützlichen und interessanten Sätze manuell auswählen muß [MT97a, Klu97].

Abhängig von der Wahl des Akzeptanzkriteriums und der Hypothesensprache realisiert das Finden interessanter Sätze verschiedene Aufgabenstellungen. So ist das Assoziationenlernen ein Spezialfall des Findens aller interessanten Sätze [MT97a], ebenso das Finden von Integritätsbedingungen, von funktionalen und mehrwertigen Abhängigkeiten in Datenbanken oder von häufigen Episoden in Zeitreihendaten [MT97a].

Finden von Assoziationsregeln

Das Finden von Assoziationsregeln in Transaktionsdatenbanken wurde schon früh als eine genuine Data Mining-Aufgabe erkannt [AIS93] und hat seither großes Interesse in der KDD-Forschung gefunden. Es kann als Spezialfall des Findens interessanter Sätze wie auch als Spezialfall der Teilgruppenanalyse aufgefaßt werden [WMJ00].

Formal ist eine Transaktionsdatenbank eine Menge T von Merkmalsvektoren der Form $T_t = (a_{t,1}, \ldots, a_{t,n})$ über binären Attributen $I = \{I_1, \ldots, I_n\}$. Jeder Merkmalsvektor $T_t = (a_{t,1}, \ldots, a_{t,n})$ beschreibt eine Transaktion; die Merkmale $a_{t,j}$ für $j = 1, \ldots, n$ beschreiben das Vorhandensein oder die Abwesenheit der Artikel in der Transaktion mit $a_{t,j} = 1$, falls der j-te Artikel I_j in der Transaktion enthalten ist, und $a_{t,j} = 0$ sonst.

Eine Assoziation X von Artikeln ist eine Artikelmenge mit $X \subseteq I$. Die Assoziation X deckt eine Transaktion T_t ab genau dann, wenn jeder Artikel $I_j \in X$ in der Transaktion T_t vorkommt, das heißt,

$$match(T_t, X) = \textit{wahr} \quad \text{genau dann, wenn} \quad a_{t,j} = 1 \ \ \forall I_j \in X.$$

Die Häufigkeit $supp(X)$ einer Assoziation X ist der Anteil der von X abgedeckten Transaktionen in T:

$$\textit{supp}(X) = \frac{|\{T_t \in T \mid match(T_t, X) = wahr\}|}{|T|}$$

Eine Assoziationsregel ist ein Ausdruck der Form $X \Rightarrow I_j$ mit $X \subseteq I$ und $I_j \in I$. Die Häufigkeit einer Assoziationsregel $X \Rightarrow I_j$ ist gleich der Häufigkeit der Assoziation $X \cup \{I_j\}$, also $\textit{supp}(X \Rightarrow I_j) = \textit{supp}(X \cup \{I_j\})$. Die Konfidenz $\textit{conf}(X \Rightarrow I_j)$ einer Assoziationsregel $X \Rightarrow i_j$ ist definiert als der Anteil der Transaktionen, die die Assoziation $X \cup \{I_j\}$ abdeckt, an den Transaktionen, die die Assoziation X abdeckt, also

$$\textit{conf}(X \Rightarrow I_j) = \frac{\textit{supp}(X \cup \{I_j\})}{\textit{supp}(X)}.$$

Das Ziel ist, alle Assoziationsregeln zu finden, deren Häufigkeit und Konfidenz vorgegebene Schwellwerte $supp_{min}$ und $\textit{conf}_{min}$ erreichen oder überschreiten. Eine einfache Beschreibung des Grundproblems nach [MT97a, MT97b], Assoziationsregeln in einer Booleschen Datenbank zu finden, ist in Tafel 2.4 wiedergegeben.

2.2 Logikprogrammierung und Datenbanken

Die Aufgabenstellungen und Teilaufgaben, die in KDD und Data Mining auftreten, sind sehr vielfältig. Für verschiedene Anwendungsgebiete haben sich unterschiedliche Heran-

Tafel 2.4 Finden von Assoziationsregeln [MT97a, MT97b].

Gegeben:	eine Tabelle T mit n-Tupeln über den binären Attributen $I = \{I_1, \ldots, I_n\}$	
	ein Häufigkeitsschwellwert $supp_{min}$	
	ein Konfidenzschwellwert $conf_{min}$	
Gesucht:	die Menge aller Assoziationsregeln $X \Rightarrow I_j$ $(X \subseteq I, I_j \in I)$ mit	
	$supp(X \Rightarrow I_j) \geq supp_{min}$ und	(Häufigkeit)
	$conf(X \Rightarrow I_j) \geq conf_{min}$	(Konfidenz)

gehensweisen und formale Fassungen herausgebildet, wobei sich die Aufgabenstellungen zum Teil überschneiden. Eine einheitliche und theoretisch fundierte formale Grundlage zu ihrer Beschreibung bietet die Logik. Die Logik erklärt auch die Semantik relationaler Datenbanken.

Dieser Abschnitt führt Begriffe und Grundlagen aus Logik und Modelltheorie ein, die für die vorliegende Arbeit von Bedeutung sind. Weitergehende Einführungen und Überblicke bieten zum Beispiel [Llo87, NCdW97, CGT90, Sch92]. Die Definitionen und Sätze des folgenden Abschnitts stammen größtenteils aus [Llo87].

2.2.1 Grundbegriffe der Logik

Das nicht-logische anwendungsabhängige Vokabular einer Logiksprache besteht aus einer Menge von Prädikatsymbolen und einer Menge von Funktorsymbolen, die jeweils eine bestimmte Stelligkeit aufweisen. Ein n-stelliges Prädikat p wird auch geschrieben als p/n. Funktorsymbole mit der Stelligkeit 0 heißen Konstanten. Sprachen, in denen alle Funktoren Konstanten sind, heißen funktorenfrei. Das unveränderliche logische Vokabular der Logik 1. Ordnung umfaßt Variablen $V_1, V_2, \ldots$, die ein- und zweistelligen Junktoren $\neg, \wedge, \vee, \leftarrow, \leftrightarrow$, den Existenzquantor $\exists$ und den Allquantor $\forall$. Punktionszeichen sind das Komma und die Klammern (und). Nach den bekannten Regeln lassen sich daraus wohlgeformte Formeln bilden.

Definition 1 *Ein Term ist eine Variable V, eine Konstante c oder ein Funktorsymbol mit einem geklammerten Tupel von Termen $f(t_1, \ldots, t_m)$. Eine Formel ist induktiv definiert wie folgt:*

(a) wenn p ein n–stelliges Prädikat und $t_1, \ldots, t_n$ Terme sind, ist $p(t_1, \ldots, t_n)$ eine Formel;

(b) wenn F und G Formeln sind, dann sind auch $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ und $(F \leftrightarrow G)$ Formeln;

(c) wenn F eine Formel und x eine Variable ist, dann sind auch $(\exists x F)$ und $(\forall x F)$ Formeln.

Eine Formel, die gemäß (a) aufgebaut ist, wird auch als Atom oder atomare Formel bezeichnet. Ein Literal l ist ein Atom $p(t_1, \ldots, t_n)$ oder ein negiertes Atom $\neg p(t_1, \ldots, t_n)$. Ein positives Literal ist ein Atom, ein negatives Literal ist ein negiertes Atom.

Variablen V in prädikatenlogischen Formeln können durch den Existenzquantor $\exists$ oder den Allquantor $\forall$ gebunden sein. Der Geltungsbereich der Quantoren $\forall$ und $\exists$ in den Formeln $\forall V F$ beziehungsweise $\exists V F$ ist F. Das Vorkommen einer Variable in einer Formel ist gebunden, wenn die Variable unmittelbar hinter einem Quantor und im Geltungsbereich des Quantors vorkommt. Vorkommen von Variablen, die nicht durch einen Quantor gebunden sind, heißen freie Vorkommen der Variablen.

Formeln, in denen alle Variablen durch einen Quantor gebunden sind, heißen geschlossene Formeln. Wenn F eine Formel darstellt, so bezeichnet $\forall(F)$ den All-Abschluß von F,

in dem jedes freie Vorkommen einer Variablen V in F durch Voranstellen eines Allquantors $\forall V$ vor F gebunden wird. Analog ist $\exists(F)$ der Existenzabschluß von F.

Beispiel 15 *Für* $F = p(V_1, V_2) \wedge q(V_1)$ *heißt* $\forall(F) = \forall V_1 \forall V_2(p(V_1, V_2) \wedge q(V_1))$ *der All-Abschluß und* $\exists(F) = \exists V_1 \exists V_2(p(V_1, V_2) \wedge q(V_1))$ *der Existenz-Abschluß von* F.

Zur Schreibweise. Prädikat- und Funktorsymbole und damit auch Konstanten werden im folgenden als Kleinbuchstaben, Wörter mit kleinem Anfangsbuchstaben oder beliebige in Hochkommata eingeschlossene Zeichenketten geschrieben. Außer $V_1, V_2, \ldots$ bezeichnen auch andere Großbuchstaben oder Wörter mit großem Anfangsbuchstaben Variablen.

Beispiel 16 p, q, *pred sind meist Prädikatsymbole.* c, *const, 'Const.' sind Konstantenzeichen.* f *ist meist ein Funktorsymbol. Var, Id,* V_1, V_2, V_3 *bezeichnen Variablen.*

Substitutionen

Eine *Substitution* θ ist eine endliche Menge der Form $\{X_1/t_1, \ldots, X_n/t_n\}$. Dabei ist jedes X_i eine Variable und jedes t_i ein Term ungleich X_i. Die Variablen X_i sind untereinander verschieden. Eine Substitution, in der die t_i keine Variablen enthalten, ist eine Grundsubstitution.

Wenn F eine Formel und $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$ eine Substitution ist, bezeichnet $F\theta$ die Formel, die man erhält, wenn man gleichzeitig jedes freie (nicht durch einen Quantor gebundene) Vorkommen von X_i in F durch t_i ersetzt. $F\theta$ heißt dann eine Instanz von F. Wenn $F\theta$ keine Variablen enthält, ist $F\theta$ eine Grundinstanz von F. Ein Atom ohne Variablen heißt Grundatom.

Beispiel 17 *Es sind* $\theta_1 = \{X/a, Y/Z\}$ *und* $\theta_2 = \{X/a, Y/b\}$ *Substitutionen. Die Substitution* θ_2 *ist eine Grundsubstitution.*

Es sei $F = p(X, Y)$ *eine Formel. Dann sind* $F\theta_1 = p(a, Z)$ *und* $F\theta_2 = p(a, b)$ *Instanzen von* F *und* $F\theta_2$ *ist eine Grundinstanz von* F.

Klauseln

Eine besondere Rolle spielen in der Logikprogrammierung die Klauseln. Eine Klausel $C = a_1 \vee \ldots \vee a_k \vee \neg b_1 \vee \ldots \vee \neg b_n$ ist eine Disjunktion von positiven Literalen $a_1, \ldots, a_k$ und negativen Literalen $\neg b_1, \ldots, \neg b_n$. Alle Variablen in einer Klausel sind implizit allquantifiziert. Eine Klausel $C = a_1 \vee \ldots \vee a_k \vee \neg b_1 \vee \ldots \vee \neg b_n$ wird auch als Implikation $C = a_1 \vee \ldots \vee a_k \leftarrow b_1 \wedge \ldots \wedge b_n$ oder als Literalmenge $C = \{a_1, \ldots, a_k, \neg b_1, \ldots, \neg b_n\}$ geschrieben. Die Disjunktion $a_1 \vee \ldots \vee a_k$ heißt Klauselkopf, die Konjunktion $b_1 \wedge \ldots \wedge b_n$ heißt Klauselrumpf. Die leere Klausel $\square$ ist die Klausel mit leerem Rumpf und leerem Kopf. Sie bezeichnet einen Widerspruch, also eine unerfüllbare Formel.

Modelle

Eine Präinterpretation über einer Logiksprache L leistet das folgende:

(1) sie legt einen Wertebereich D fest (der auch als Universum bezeichnet wird) und definiert

(2) für alle Konstanten in L eine Abbildung auf D,

(3) für alle n-stelligen Funktorsymbole in L eine Abbildung von D^n auf D.

Eine Interpretation über einer Logiksprache L bestimmt zusätzlich noch

(4) für alle n-stelligen Prädikatsymbole in L eine Abbildung von D^n auf $\{$wahr, falsch$\}$.

Die Interpretation gibt also für alle Grundatome, die in L bildbar sind, einen Wahrheitswert an. Daraus lassen sich die Wahrheitswerte für alle geschlossenen Formeln in L ableiten.

Definition 2 *Ein Modell einer geschlossenen Formel F ist eine Interpretation $\mathcal{M}$, die die Formel F wahr macht. Man schreibt dann $\mathcal{M} \models F$.*

Eine Menge von geschlossenen Formeln S impliziert eine geschlossene Formel F, wenn alle Modelle von S auch Modelle von F sind. Man schreibt dann $S \models F$ und sagt: „F genügt S“ oder „F folgt aus S“ oder „S impliziert F“. Für $S \not\models F$ sagt man auch, F verletzt S.

Eine geschlossene Formel ist erfüllbar, wenn sie ein Modell hat. Eine geschlossene Formel F ist allgemeingültig, wenn alle ihre Interpretationen Modelle sind. Man schreibt $\models F$.

Eine Formel F ist unerfüllbar, wenn sie keine Modelle besitzt. Das Zeichen $\square$ bezeichnet eine unerfüllbare Formel. Man schreibt auch $F \models \square$, wenn F unerfüllbar ist.

Herbrand-Interpretationen und -Modelle

Herbrand-Interpretationen und -Modelle sind von besonderer Bedeutung, denn für Klauseln gilt: eine Klausel ist genau dann erfüllbar, wenn sie ein Herbrand-Modell hat. Zu einer Herbrand-Interpretation gehört eine Herbrand-Präinterpretation, die die Prädikat- und Funktorsymbole einer Logiksprache auf spezielle Werte abbildet, und zwar bildet sie jede Konstante und jeden Term auf eine gleichlautende Zeichenkette ab.

Beispiel 18 *Das Herbrand-Universum (der Wertebereich) einer Logiksprache mit dem einstelligen Funktorsymbol f und der Konstanten a ist:*

$$\{a, f(a), f(f(a)), f(f(f(a))), \ldots\}$$

Jeder Term wird sozusagen auf sich selbst abgebildet, also die Konstante a auf die Zeichenkette a, der Term $f(a)$ auf die Zeichenkette $f(a)$ und so weiter.

Als Herbrand-Basis bezeichnet man die Menge der Grundatome, die sich aus den Elementen des Herbrand-Universums und den Prädikatsymbolen der Logiksprache erzeugen lassen.

Beispiel 19 *Die Herbrand-Basis einer Logiksprache mit dem einstelligen Funktorsymbol f, der Konstanten a und dem einstelligen Prädikatsymbol p ist:*

$$\{p(a), p(f(a)), p(f(f(a))), p(f(f(f(a)))), \ldots\}$$

Da die Herbrand-Basis eindeutig festgelegt ist, läßt sich eine Zuweisung von Wahrheitswerten an die Grundatome definieren durch Nennen der Grundatome, denen $wahr$ zugewiesen ist. Eine Herbrand-Interpretation läßt sich daher mit der Menge derjenigen Grundatome aus der Herbrand-Basis identifizieren, denen sie den Wahrheitswert $wahr$ zuweist.

2.2.2 Logikprogrammierung

Die Repräsentation eines Anwendungsbereichs durch eine Logiktheorie besteht darin, die grundlegenden Eigenschaften und Zusammenhänge dieses Bereichs durch die entsprechenden prädikatenlogischen Formeln auszudrücken. Eine Theorie ist eine Menge von Formeln; die in der Theorie enthaltenen Formeln sind konjunktiv verknüpft. Um festzustellen, ob der Sachverhalt, den eine Formel F darstellt, in dem durch die Theorie Th repräsentierten Anwendungsbereich zutrifft, oder anders gesagt, ob F aus Th folgt, überprüft man die Formel $Th \rightarrow F$ auf ihre Allgemeingültigkeit. Dies geschieht üblicherweise durch Herleiten eines Widerspruchs aus der Theorie $Th \wedge \neg F$.

Die Prädikatenlogik 1. Ordnung ist unentscheidbar, das heißt, daß kein Algorithmus existiert, der für eine beliebige Formel F in endlicher Zeit entscheiden kann, ob F allgemeingültig ist oder nicht. Es gibt lediglich Semi-Entscheidungsverfahren, die in endlicher Zeit stoppen, wenn die gestellte Frage („Ist F erfüllbar?“ oder „Ist F unerfüllbar?“) mit „Ja“ beantwortet werden kann. Für allgemeine Prädikatenlogik 1. Ordnung gibt es also keine korrekte und vollständige Beweisprozedur. Es existieren jedoch korrekte und vollständige Beweisverfahren für Theorien und Formeln, die bestimmten Einschränkungen genügen.

Resolution

Das am häufigsten verwendete Verfahren, um die Unerfüllbarkeit von $Th \wedge \neg F$ nachzuweisen, ist die *Resolution* [Rob65]. Das Resolutionsverfahren ist anwendbar für Theorien $Th \wedge \neg F$, die als Klauseln repräsentiert sind. Ein Resolutionsschritt besteht darin,

(1) zwei Klauseln der Form

$$\begin{aligned} C_1 &= \{l_1, \ldots, l_n, \neg r_1\}, \\ C_2 &= \{k_1, \ldots, k_m, r_2\} \end{aligned}$$

auszuwählen,

(2) eine Substitution θ anzuwenden, für die $r_1\theta = r_2\theta$ wird, und

(3) $C_1\theta$ und $C_2\theta$ zur folgenden Klausel $C_{1,2}$ zu resolvieren:

$$\begin{aligned} C_{1,2} &= (C_1 \setminus \{\neg r_1\} \cup C_2 \setminus \{r_2\})\theta \\ &= \{l_1, \ldots, l_n, k_1, \ldots k_m\}\theta. \end{aligned}$$

Die Klauseln C_1 und C_2, die resolviert werden, stammen aus der Theorie $Th \wedge \neg F$ oder sind aus früheren Resolutionsschritten hervorgegangen. Die Klausel $C_{1,2}$ heißt *Resolvent* von C_1 und C_2.

Der Resolution liegt folgende Idee zugrunde: Wenn $Th \wedge \neg F$ erfüllbar ist, sind sowohl $C_1\theta$ als auch $C_2\theta$ erfüllbar. Wegen $\neg r_1\theta = r_2\theta$ kann aber nur entweder $\neg r_1\theta$ oder $r_2\theta$ gelten. Deshalb muß $(C_1 \setminus \{\neg r_1\}\theta)$ oder $(C_2 \setminus \{r_2\})\theta$ erfüllbar sein und damit auch die Disjunktion $(C_1 \setminus \{\neg r_1\}\theta) \cup (C_2 \setminus \{r_2\})\theta$.

Ein Resolutionsbeweis für $Th \wedge \neg F$ ist erfolgreich, wenn es gelingt, mit wiederholten Resolutionsschritten schließlich die unerfüllbare *leere Klausel* $\square$ herzuleiten. Dann ist die Unerfüllbarkeit von $Th \wedge \neg F$ und somit die Allgemeingültigkeit von $Th \rightarrow F$ gezeigt.

Definite Programme

Ein definites Programm ist eine endliche Menge definiter Programmklauseln. Eine definite Programmklausel ist eine Klausel $a \leftarrow l_1 \wedge \ldots \wedge l_n$, deren Kopf aus genau einem Atom a besteht; die l_i sind Atome. Eine definite Programmklausel mit leerem Rumpf $a \leftarrow$ ist ein Fakt. Ein Fakt, das keine Variablen enthält, heißt Grundfakt. Eine definite Anfrage oder definite Zielklausel ist eine Klausel der Form $\leftarrow a_1, \ldots, a_n$, also eine Klausel mit leerem Kopf. Eine definite Programmklausel mit einem oder keinem Kopfliteral heißt Hornklausel. Im Unterschied zu allgemeinen Theorien besitzen Horntheorien und damit auch definite Programme ein eindeutiges minimales Herbrandmodell. Dieses minimale Herbrandmodell enthält genau die Grundatome, die die Horntheorie impliziert.

SLD-Resolution

Die SLD-Resolution ist ein Beweisverfahren für definite Logikprogramme und Zielklauseln, das auf der Resolution aufbaut. Das S steht für Selektionsregel, das D für Definite Klauseln und L bedeutet Lineare Resolution. Die lineare Resolution und die Selektionsregel schreiben vor, welche Klauseln während eines Resolutionsbeweises im jeweils nächsten Resolutionschritt miteinander zu resolvieren sind. Damit ist der Ablauf eines SLD-Resolutionsbeweises determiniert.

Wenn eine Zielklausel $G = \leftarrow a_1, \ldots, a_n$ Variablen enthält, so ist eine Substitution θ für Variablen in G eine Antwort für $P \cup \{G\}$. Eine Antwort θ ist eine korrekte Antwort für $P \cup \{G\}$, wenn $\forall\,((a_1, \ldots, a_n)\,\theta)$ aus P folgt. Die Menge aller korrekten Antworten heißt die Antwortmenge von $P \cup \{G\}$.

Die SLD-Prozedur berechnet Antworten für $P \cup \{G\}$ aus den Substitutionen, die während des Resolutionsbeweises von $P \cup \{G\}$ ausgeführt wurden, um Literale in den resolvierten Klauseln zu unifizieren.

Eine Beweisprozedur für $P \cup \{G\}$ heißt korrekt, wenn sie nur korrekte Antworten berechnet, und vollständig, wenn sie alle korrekten Antworten berechnet.

Für definite Logikprogramme und definite Zielklauseln ist die SDL-Resolution eine korrekte Beweisprozedur. Sie ist vollständig in dem Sinn, daß sie für jede korrekte Antwort θ eine Antwort σ findet, so daß $\theta = \sigma\gamma$ für eine Substitution γ.

Normale Programme

Anders als in definiten Logikprogrammen sind in normalen Logikprogrammen negative Literale in Klauselrümpfen und damit auch in Anfragen erlaubt. Eine normale Programmklausel ist eine Klausel der Form $a \leftarrow l_1 \wedge \ldots \wedge l_n$, wobei a ein Atom ist und die l_i Literale sind. Ein normales Programm ist eine endliche Menge normaler Programmklauseln. Eine normale Anfrage- oder Zielklausel G ist eine Klausel mit leerem Kopf $\leftarrow l_1, \ldots, l_n$.

Semantik normaler Programme

Da in normalen Programmen negative Rumpfliterale erlaubt sind, kann auch negative Information dargestellt und in Beweisen verwendet werden, beispielweise mit einer Klausel $a \leftarrow \neg b$, die besagt, daß a gilt, wenn b falsch ist. Um a folgern zu können, muß $\neg b$ bewiesen werden. In normalen Programmen sind aber nur nichtnegierte Kopfliterale zugelassen; es kann also keine Klausel $\neg b \leftarrow l_1, \ldots, l_n$ geben, aus der $\neg b$ direkt gefolgert werden kann. Wenn b aus der Theorie herleitbar ist, ist damit gezeigt, daß $\neg b$ falsch ist. Wenn aber b aus der Theorie nicht herleitbar ist, ist der Wahrheitswert von $\neg b$ unbekannt.

Die in der Logikprogrammierung übliche Lösung für dieses Problem ist die Closed World Assumption (CWA, „Annahme der abgeschlossenen Welt“), die auf Reiter [Rei87] zurückgeht. Darunter versteht man die Annahme, daß eine Theorie den abgebildeten Weltbereich so vollständig beschreibt, daß alle darin wahren Aussagen aus der Theorie ableitbar sind. Wenn ein Grundatom b nicht aus einem Logikprogramm folgt, so ist nach der CWA $\neg b$ zu folgern.

Um die Semantik der Folgerungsbeziehung unter der CWA zu beschreiben, greift man auf die Vervollständigung $Comp(P)$ eines Logikprogramms P zurück [Cla87, Llo87], in der die Prädikatdefinitionen von P von Wenn-Aussagen zu Genau-dann-wenn-Aussagen ergänzt sind[1]. Analog zu definiten Logikprogrammen heißt für ein normales Logikprogramm P und eine normale Zielklausel $G = \leftarrow l_1, \ldots, l_n$ eine Substitution θ für die Variablen in $l_1, \ldots, l_n$ eine Antwort für $P \cup \{G\}$. Die Antwort θ heißt eine korrekte Antwort für $P \cup \{G\}$, wenn $\forall((l_1, \ldots, l_n)\theta)$ eine logische Folgerung von $Comp(P)$ ist, also wenn $Comp(P) \models \forall((l_1, \ldots, l_n)\theta)$.

[1] Außerdem enthält die Vervollständigung noch eine Gleichheitstheorie, die die Bedeutung des Gleichheitszeichens „=“ definiert.

Für definite Programme P gilt, daß aus der Vervollständigung $Comp(P)$ dieselbe nichtnegierte Information ableitbar ist wie aus P, also wenn $Comp(P) \models \forall(a_1 \wedge \ldots \wedge a_m)$, dann $P \models \forall(a_1 \wedge \ldots \wedge a_m)$. Für normale Programme gilt das nicht; jedes normale Programm ist widerspruchsfrei, aber die Vervollständigung *Comp(P)* eines normalen Programms P kann widersprüchlich sein.

Die CWA ist eine natürliche Annahme, wenn die gesamte Information in Form von Grundfakten vorliegt, denn dann sind die nicht-gültigen Fakten leicht zu erkennen. Für komplexere Logikprogramme ist die CWA problematisch: wegen der Unentscheidbarkeit der Prädikatenlogik gibt es keinen Algorithmus, der für ein beliebiges a in endlicher Zeit entscheiden kann, ob a eine logische Konsequenz von P ist oder nicht. Um die CWA in der Logikprogrammierung dennoch praktisch anwenden zu können, beschränkt man ihre Anwendung auf solche a, für die in endlicher Zeit beweisbar ist, daß a nicht logisch aus P folgt. Diese Inferenzregel (nimm $\neg a$ an, wenn ein versuchter Beweis von a in endlicher Zeit versagt), heißt Negation durch Versagen des Beweises (*negation by failure*). Die Menge der Atome a, für die $\neg a$ per Negation durch Beweisversagen folgt, ist also eine Teilmenge der Menge, für die $\neg a$ aus der CWA folgt.

SLDNF und erlaubtes Anfragestellen

SLDNF ist die SLD-Beweisprozedur erweitert um Negation durch Beweisversagen. Die SLDNF-Resolution ist korrekt für normale Logikprogramme P und normale Zielklauseln G, das heißt, jede berechnete Substitution θ für die Variablen von G folgt logisch aus $Comp(P) \cup \{G\}$.Die SLDNF-Resolution ist jedoch nur unter Einschränkungen vollständig für normale Logikprogramme, nämlich für hierarchische normale Logikprogramme und erlaubtes Anfragestellen $P \cup \{G\}$. Das Problem der SLD-Resolution im Zusammenhang mit Negation durch Beweisversagen liegt darin, daß SLD allgemeinste Antworten berechnet, negative Teilziele aber nur auswerten soll, wenn sie vollständig instanziiert sind. Die Einschränkung auf hierarchische normale Logikprogramme und erlaubtes Anfragestellen stellt sicher, daß Teilziele immer grund sind, wenn sie ausgewertet werden. Es folgen die Definitionen für hierarchische normale Logikprogramme und erlaubtes Anfragestellen.

Definition 3 *Ein normales Logikprogramm P ist hierarchisch, wenn die darin vorkommenden Prädikatsymbole so in eine Ordnung $<\cdot$ gebracht werden können, daß für jede Programmklausel $a \leftarrow l_1, \ldots, l_n$ das Prädikatsymbol des Kopfliterals a bezüglich $<\cdot$ kleiner ist als die Prädikatsymbole aller Rumpfliterale $l_1, \ldots, l_n$.*

Für jedes definite oder hierarchische normale Logikprogramm P gilt, daß die Vervollständigung $Comp(P)$ ein minimales normales Herbrand-Modell besitzt und somit konsistent ist. Minimal bedeutet, daß es kein echt kleineres normales Herbrand-Modell gibt; normal bedeutet, daß das Modell die Identitätsrelation dem Prädikatsymbol $=$ zuordnet.

Definition 4 (Bereichsbeschränktheit) *Eine normale Programmklausel C ist erlaubt und bereichsbeschränkt, wenn jede Variable der Klausel in einem positiven Rumpfliteral von C vorkommt. Eine normale Zielklausel G ist bereichsbeschränkt, wenn jede Variable in G in einem positiven Rumpfliteral von G vorkommt.*

Daraus folgt, daß ein Fakt genau dann bereichsbeschränkt ist, wenn es ein Grundfakt ist.

Definition 5 (Erlaubtes Anfragestellen,[Llo87]) *Es sei P ein normales Logikprogramm und G eine normale Zielklausel. $P \cup \{G\}$ ist erlaubt, wenn*

- *alle Programmklauseln in P zulässig sind,*
- *jede Programmklausel mit einem Kopfliteral a, dessen Prädikatsymbol in einem positiven Rumpfliteral der Zielklausel G oder einer anderen Programmklausel in P vorkommt, bereichsbeschränkt ist, und wenn*

- *G bereichsbeschränkt ist.*

Eine normale Programmklausel $C = a \leftarrow l_1, \ldots, l_n$ ist zulässig, wenn jede Variable in der Klausel im Kopfliteral a oder in einem positiven Rumpfliteral von C vorkommt.

Mit einer sicheren Selektionsregel ist die SLDNF-Resolution für hierarchische normale Programme vollständig, das heißt, sie berechnet für erlaubte $P \cup \{G\}$ alle korrekten Grundantworten. Eine Selektionsregel ist sicher, wenn sie in einer SLDNF-Ableitung für den nächsten Resolutionsschritt nur solche Zielklauseln auswählt, in denen alle negativen Literale grund sind.

Prolog

Prolog ist eine verbreitete Programmiersprache, die die Prinzipien der Logikprogrammierung für allgemeine praktische Programmierarbeiten anwendbar macht. Ein Prolog–Programm besteht im wesentlichen aus einer Menge definiter Programmklauseln und wird durch Stellen einer Anfrage ausgeführt.

Prolog-Programme lassen sich durch einen Prolog-Interpreter auswerten. Der Prolog-Interpreter ist eine Beweisprozedur, die in den meisten gängigen Prolog-Implementationen auf der SLDNF-Resolution beruht. Aus Gründen der Effizienz sind die Prolog-Interpreter meist so implementiert, daß die Auswahl der Klauseln, die resolviert werden, von der Reihenfolge der Klauseln im Programm und von der Reihenfolge der Literale in den Klauseln abhängt. Daher kann es zum Beispiel vorkommen, daß Widerspruchsbeweise in rekursiven definiten Logikprogrammen nicht terminieren oder falsche Ergebnisse liefern. SLDNF-Widerspruchsbeweise mit hierarchischen normalen Programmen und erlaubten Zielklauseln terminieren zwar; da die Selektionsregel jedoch nicht sicher ist, ist die Vollständigkeit der Ergebnisse nicht garantiert. Bei der Erstellung von Prolog-Programmen muß daher der Programmierer auf eine geeignete Anordnung der Literale und Klauseln achten.

Die Notation von Klauseln in Prolog weicht von der üblichen logischen Notation ab. In Prolog-Notation wird eine logische Klausel $a \leftarrow l_1 \wedge \ldots \wedge l_n$ als $a :- l_1, \ldots, l_n$ geschrieben. Konstanten und Prädikatsymbole müssen mit Kleinbuchstaben beginnen oder in Hochkommata eingeschlossen sein, zum Beispiel ”Const”. Variablen beginnen mit Großbuchstaben. Der Unterstrich „_“ stellt eine sogenannte anonyme Variable dar. Mehrere Unterstriche in einer Klausel stehen für mehrere, verschiedene Variablen. Anonyme Variablen verwendet man, wenn der Wert der Variablen im Programm nicht benötigt wird.

2.2.3 Datenbanken

Deduktive Datenbanken

Logikprogamme und Datenbanken stehen in einem engen Zusammenhang. Deduktive Datenbanken (nach [Llo87, LD94]) unterscheiden sich in zwei Aspekten von Logikprogrammen. Erstens sind deduktive Datenbanken typisiert. Ein Typ ist ein Wertebereich, also eine Menge von Werten. Der Wertevorrat der Interpretation einer deduktiven Datenbank kann aus mehreren Typen bestehen. Jeder Stelle eines Prädikats oder Funktors ist ein Typ zugeordnet. Die Stelle kann nur mit Werten des zugehörigen Typs belegt werden. Da sich typisierte Formeln erster Ordnung mit Standard-Transformationen in entsprechende typenfreie Formeln erster Ordnung überführen lassen und die Folgerungsbeziehungen in der typisierten Datenbank mit denen der transformierten Datenbank übereinstimmen, ist dieser Unterschied nicht gravierend.

Der zweite Unterschied zwischen Logikprogrammen und Datenbanken liegt im Bereich der Semantik. Anders als in der Logikprogrammierung gehört zur Semantik von deduktiven Datenbanken neben der CWA auch das Axiom der abgeschlossenen Welt (*domain closure axiom*) nach Reiter [Rei84]. Das *domain closure axiom* besagt, daß es in der durch die Datenbank repräsentierten Welt außer den explizit genannten keine weiteren Individuen

gibt. Mit anderen Worten heißt das, daß der Wertebereich jeder Interpretation endlich und in der Theorie bekannt ist. Die Vervollständigung einer deduktiven Datenbank enthält deshalb neben den Axiomen der *closed world assumption* zusätzlich noch *domain closure*–Axiome, die alle möglichen Objekte im Wertebereich, also alle möglichen Grundinstanziierungen von Variablen in der Theorie, durch explizites Auflisten bekannt machen. Im funktorfreien Fall kann das *domain closure axiom* zum Beispiel so aussehen: $\forall V(V = c_1 \vee V = c_2 \vee \ldots \vee V = c_{max})$. Die Semantik deduktiver Datenbanken beschreibt die folgende Definition:

Definition 6 *Es sei D eine Datenbank und $Q = \leftarrow W$ eine Anfrage und θ eine Antwort für $D \cup \{Q\}$. Dann ist θ eine korrekte Antwort für $Comp(D) \cup \{Q\}$, wenn $Comp(D) \models \forall(W\theta)$.*

Für Datenbanken gelten sinngemäß die Definitionen für Logikprogramme, mit dem Unterschied der Typisierung:

Definition 7 *Eine normale Datenbank-Klausel ist eine typisierte Klausel erster Ordnung der Form $a \leftarrow l_1 \wedge \ldots \wedge l_n$. Die l_i sind Literale, a ist ein Atom. Eine normale Datenbank ist eine endliche Menge normaler Datenbank-Klauseln. Eine normale Anfrage ist eine Anfrage der Form $G = \leftarrow l_1 \wedge \ldots \wedge l_n$. Die l_i sind Literale.*

Eine definite Datenbank-Klausel ist eine normale Datenbank-Klausel der Form $a \leftarrow l_1 \wedge \ldots \wedge l_n$. Die l_i und a sind Atome. Eine definite Datenbank ist eine endliche Menge von definiten Datenbank-Klauseln. Eine definite Anfrage ist eine Anfrage der Form $G = \leftarrow a_1 \wedge \ldots \wedge a_n$. Die a_i sind Atome.

Es sei D eine Datenbank und $Q = \leftarrow W$ eine Anfrage. W enthalte die freien Variablen $V_1, \ldots, V_n$. Eine Antwort für $D \cup \{Q\}$ ist eine Substitution θ für einige oder alle Variablen $V_1, \ldots, V_n$.

Es sei D eine Datenbank, $Q = \leftarrow W$ eine Anfrage und θ eine Antwort für $D \cup \{Q\}$. Dann heißt θ eine korrekte Antwort für $Comp(D) \cup \{Q\}$, wenn $\forall(W\theta)$ eine logische Konsequenz von $Comp(D)$ ist, also $Comp(D) \models \forall(W\theta)$.

Für deduktive Datenbanken ist die Anfrageauswertung durch SLDNF korrekt.

Eine deduktive hierarchische Datenbank ist eine deduktive Datenbank mit nicht–rekursiven Prädikatdefinitionen und nicht-rekursiven Typen. Für definite Datenbanken D_d und definite Anfragen $\leftarrow W_d$ und für hierarchische Datenbanken D_h mit hierarchischen Typen und normalen Anfragen $\leftarrow W_h$ ist die SLDNF-Beweisprozedur auch vollständig, das heißt, sie berechnet alle Grundsubstitutionen θ für die Variablen in W_x, für die gilt $Comp(D_x) \cup \{W_x\} \models W_x\theta$ für $x = h, d$.

Relationale Datenbanken

Relationale Datenbanken sind deduktive Datenbanken, die nur aus Grundfakten bestehen. Aufgrund ihrer praktischen Bedeutung hat sich für relationale Datenbanken eine eigenständige Terminologie herausgebildet. Entsprechungen zwischen Begriffen aus der Logikprogrammierung und dem Bereich relationaler Datenbanken sind in Tabelle 2.5 zusammengestellt [Dže96a].

Definition 8 *Eine relationale Datenbank besteht aus einer Menge von Relationen. Eine Relation R ist eine Menge von Tupeln und zwar eine Teilmenge eines Kartesischen Produkts $D_1 \times \ldots \times D_n$ über n Wertebereichen oder Typen $D_1, \ldots, D_n$. Die Stelligkeit von R ist n. Die Wertebereiche D_i können unendlich sein. Relationen sind üblicherweise endlich.*

Ein Relationenschema ist ein n-Tupel $(D_1, \ldots, D_n)$ aus Bezeichnungen von Wertebereichen. Die Bezeichnungen von Wertebereichen D_i im Relationenschema $(D_1, \ldots, D_n)$ heißen Attribute.

Ein Schema beschreibt alle möglichen Tupel, während eine Relation die tatsächlichen Tupel enthält. Eine Relation wird oft auch als Tabelle betrachtet, bei der jede Zeile aus einem Tupel besteht. Die Spalten der Tabelle entsprechen den Attributen.

Tafel 2.5 Entsprechungen zwischen der Terminologie von Logikprogrammierung und Relationalen Datenbanken.

Datenbanken	Logikprogrammierung
Relationenname p	Prädikatsymbol p
Attribut der Relation	Argument des Prädikats
Tupel $\in p\ \langle a_1, \ldots, a_n \rangle$	Grundfakt $p(a_1, \ldots, a_n)$
extensionale Definition von p — als Menge von Tupeln	extensionale Definition von p — als Menge von Grundfakten
intensionale Definition von p — durch eine Sicht	intensionale Definition von p — durch eine Menge von Regeln (Klauseln)

Relationale Datenbanken und deduktive hierarchische Datenbanken sind gleich ausdrucksstark. Da deduktive hierarchische Datenbanken intensionale Prädikatdefinitionen erlauben, lassen sie sich im allgemeinen kompakter repäsentieren, werden aber immer noch als relationale Datenrepräsentation betrachtet.

Die Theorie der relationalen Datenbanken liegt den weitverbreiteten relationalen Datenbank–Mamagementsystemen (RDBMS) zugrunde. Für die praktische Anwendung der relationalen Datenbanken und RDBMS sind auch Aspekte von großer Bedeutung, die in der Logikprogrammierung keine Beachtung finden. Dazu gehören Schlüssel und Normalisierung.

Ein Schlüssel ist eine minimale Teilmenge der Attribute eines Relationenschemas, in denen sich alle Tupel einer Relation über dem Relationenschema unterscheiden. Jede Relation muß einen Schlüssel besitzen, da eine Relation eine Menge ist und folglich nur unterschiedliche Tupel enthält. Relationen, in denen sämtliche Nicht-Schlüsselattribute funktional vom Schlüssel abhängen und keine weiteren funktionalen Abhängigkeiten bestehen, nennt man normalisiert.

Wenn der Schlüssel einer Relation r in einer anderen Relation r' vorkommt, so ist der Schlüssel von r' ein Fremdschlüssel für r [Sch87]. Fremdschlüssel stellen Beziehungen zwischen Relationen dar. Eine $1:1$–Beziehung liegt vor, wenn es zu jeder Wertekombination der Schlüsselattribute von r genau eine Wertekombination des Fremdschlüssels gibt. Eine $1:n$–Beziehung liegt vor, wenn es zu jeder Wertekombination der Schlüsselattribute von r mehrere Wertekombinationen des Fremdschlüssels geben kann.

Beispiel 20 *Gegeben sei eine Datenbank mit den zwei Relationen* $silbe(\underline{SId}, Len, Acc)$ *und* $phonem(\underline{PId}, SId, Typ)$*. Der Schlüssel* SId *der Relation* $silbe$ *kommt auch als abhängiges Attribut in der Relation* $phonem$ *vor. Daher ist der Schlüssel* PId *der Relation* $phonem$ *ein Fremdschlüssel für die Relation silbe.*

Da eine Silbe mehrere Phoneme enthalten kann, referenziert eine SId *mehrere* PId*. Die Beziehung zwischen* $silbe$ *und* $phonem$ *ist daher eine* $1:n$*–Beziehung.*

2.3 Induktive Logikprogrammierung

Die Induktive Logikprogrammierung (ILP) ist im Schnittpunkt von Logikprogrammierung und Maschinellem Lernen angesiedelt [MDR94]. Aus dem Maschinellen Lernen übernimmt sie ihre zentrale Aufgabenstellung: die Entwicklung von Verfahren und Werkzeugen zum Induktiven Lernen. Induktives Lernen generalisiert von vorgegebenen einzelnen Instanzen oder Beobachtungen (den sogenannten Beispielen), um allgemeine Regeln oder Muster (die sogenannten Hypothesen) zu formulieren, die auch für neue, noch unbekannte Fälle zutreffen. Aus der Logikprogrammierung stammen die Repräsentationsformalismen der ILP, also definite oder normale Logikprogramme und Datenbanken, sowie die entsprechenden Deduktionsverfahren. Die Nähe zur Logikprogrammierung führte auch zu einer

Tafel 2.6 Die normale Fassung der ILP.

Gegeben:	eine Menge positiver Beispiele E^+	
	eine (möglicherweise leere) Menge negativer Beispiele E^-	
	Hintergrundwissen B mit $B \not\models E^+$	
	eine Hypothesensprache L_h	
	ein Präferenzkriterium für Hypothesen aus L_h	
Gesucht:	eine Hypothese H, für die gilt:	
	$\forall e^+ \in E^+ \colon B \cup H \models e^+$	(Vollständigkeit)
	$\forall e^- \in E^- \colon B \cup H \not\models e^-$	(Korrektheit)
	H ist optimal bezüglich eines Präferenzkriteriums	

starken theoretischen Orientierung der ILP, so daß theoretische und formale Eigenschaften der entwickelten Verfahren wie Konvergenzverhalten und Komplexität oder auch die Erforschung der Lernbarkeit großes Interesse finden [MDR94]. Neuerdings beschäftigt sich die Forschung vermehrt mit der praktischen Anwendung der ILP [Rae00, BMK98, Dže01]. Die Hauptthemen der ILP sind damit die Untersuchung des theoretischen Rahmens der Induktion, die Entwicklung von Algorithmen zum Induktiven Lernen in prädikatenlogischen Repräsentationsformalismen und die praktische Anwendung der ILP–Verfahren in relationalen Lernproblemen [LWZ+96].

2.3.1 Die normale Fassung der ILP

Die ILP als abgegrenzte Forschungsrichtung begann sich in den späten 80er Jahren zu etablieren [Sam93], zum Beispiel mit den Arbeiten von Muggleton und Buntine, R. Wirth und Quinlan [MB88, Wir89, Qui90]. Der Forschungsgegenstand der ILP ist die Induktion logischer Theorien aus Beispielen. Als formale Beschreibung der ILP–Aufgabenstellung bildete sich die sogenannte normale Fassung der ILP heraus, die in Tafel 2.6 wiedergegeben ist. Gegeben sind eine Menge E^+ von positiven und eine möglicherweise leere Menge E^- von negativen Beispielen sowie eine Logiktheorie B, die bereits vorhandenes Hintergrundwissen darstellt. Als Ergebnis der Induktion ist eine vollständige und korrekte Theorie H gefordert, die zusammen mit dem Hintergrundwissen B die Beispiele erklären kann, in der Form, daß die positiven Beispiele aus induzierter Hypothese und Hintergrundwissen logisch folgen (Vollständigkeit), die negativen Beispiele aber nicht ableitbar sind (Korrektheit). Zusammen mit dem Hintergrundwissen kann die induzierte Hypothese die Beispiele ersetzen. Für die normale ILP–Fassung wird auch die Bezeichnung erklärende ILP (explanatory ILP) verwendet.

In der ILP sind Hintergrundwissen und Hypothese als Klauselmengen repräsentiert, vorwiegend in Form von Horntheorien. Ein Beispiel ist hier oft ein Grundfakt, seltener eine Klausel. Einige Systeme verlangen, daß auch das Hintergrundwissen in Form von Grundfakten vorliegt. Die Hypothesensprache L_h ist die Klauselmenge, aus der die Hypothese gewählt werden kann. Von der Hypothesensprache hängt ab, welche Hypothesen ein System lernen kann. Im allgemeinen gibt es mehrere Hypothesen, die die Forderungen nach Vollständigkeit und Korrektheit erfüllen. Das Präferenzkriterium gibt an, welche dieser Hypothesen als Ergebnis des Induktionsprozesses bevorzugt wird.

2.3.2 Die nichtmonotone Fassung der ILP

In neuerer Zeit hat mit der nichtmonotonen ILP eine weitere Fassung der ILP an Bedeutung gewonnen. Die nichtmonotone ILP geht auf die Arbeit von N. Helft [Hel89] zurück. Im Gegensatz zur normalen ILP, deren Ziel die Induktion von Hypothesen ist, aus denen (unter Verwendung des Hintergrundwissens) die positiven Beispiele gefolgert werden

Tafel 2.7 Die nichtmonotone Fassung der ILP.

Gegeben: eine Menge positiver Beispiele E^+
eine (möglicherweise leere) Menge negativer Beispiele E^-
Hintergrundwissen B
eine Hypothesensprache L_h

Gesucht: eine Hypothese $H \subset L_h$, für die gilt:
$\forall e^+ \in E^+$: $M(e^+ \cup B) \models H$
$\forall e^- \in E^-$: $M(e^- \cup B) \not\models H$

können, sucht die nichtmonotone ILP Hypothesen, die aus den Beispielen und dem Hintergrundwissen folgen. Tafel 2.7 zeigt eine neuere allgemeine Version der nichtmonotonen ILP–Fassung [WD95, NCdW97, Dže95, RD94]. Die nichtmonotone ILP betrachtet jedes Beispiel e zusammen mit dem allen Beispielen gemeinsamen Hintergrundwissen B als eine logische Theorie oder Datenbank $e \cup B$. Sie induziert Hypothesen, die aus jedem der mit dem Hintergrundwissen vervollständigten positiven Beispiele logisch folgen, aber nicht aus einem vervollständigten negativen Beispiel. Formal wird dabei anstelle jedes komplettierten Beispiels $e \cup B$ das minimale Herbrand-Modell $M^+(e \cup B)$ betrachtet, also die Menge der Grundfakten, die das Beispiel e zusammen mit dem Hintergrundwissen B impliziert. Die Gleichsetzung des komplettierten Beispiels $e \cup B$ mit seinem Modell $M^+(e \cup B)$ bringt mit sich, daß alle Grundfakten, die im komplettierten Beispiel nicht explizit aufgelistet oder daraus ableitbar sind, als falsch gelten. Damit wird eine *closed world*–Annahme realisiert, also eine Form von nichtmonotoner Infererenz, die dieser ILP–Fassung ihren Namen gibt [WD95]. (Eine Inferenzregel heißt nicht-monoton, wenn die Erweiterung einer Theorie dazu führen kann, daß die Menge ihrer logischen Konsequenzen abnimmt [Llo87].) Theoretische Unterschiede, Zusammenhänge und Übergänge zwischen den beiden ILP–Fassungen und weitere formale Fassungen diskutieren zum Beispiel [Fla95, WD95, Rae97].

Klauselentdeckung

Die nichtmonotone ILP gibt es in verschiedenen Spielarten. In der frühen Fassung der *Clausal Discovery* nach [DRB93a, DRB93b], die in dem System *Claudien* implementiert wurde und die die nichtmonotone Fassung in der ILP populär machte, gibt es keine negativen Beispiele und alle vorhandenen Daten sind in einer einzigen Theorie oder Datenbank E organisiert, die damit das einzige, und zwar positives Beispiel dieser Aufgabenstellung bildet. Gesucht ist eine Menge H von Klauseln, die die Datenbank E vollständig beschreibt, ohne überflüssige Klauseln zu enthalten Die Aufgabenstellung ist in Tafel 2.8 formalisiert. Diese Anwendung der nichtmonotonen ILP ist auf die Beschreibung der Daten ausgerichtet.

Verschiedene Semantiken sind möglich. Wenn E aus definiten Klauseln besteht, bezeichnet $M(E)$ das minimale Herbrand-Modell von E; wenn E auch negative Rumpfliterale enthält, kann es als Datenbank interpretiert werden, und $M(E)$ ist dann die Vervollständigung $Comp(E)$ [DRB93b]. Die Ur-Version des *clausal discovery*-Systems Claudien identifiziert die in der Datenbank E gültigen Klauseln h, indem es mithilfe einer geeigneten Beweisprozedur, etwa durch SLDNF-Resolution, überprüft, ob die Anfrage $body(h) \wedge \neg head(h)$ an die Datenbank E gelingt. Wenn sie gelingt, ist die Klausel h in E nicht gültig. Dieses Vorgehen beruht auf folgendem Lemma, das aus der Definition 2 der Gültigkeit einer Klausel h in einer Datenbank E folgt.

Lemma 1 *[DRB93a, DRB93b] Eine Klausel h verletzt eine Datenbank D genau dann, wenn es eine Substitution θ gibt, so daß $M(D) \models body(h)\theta$ und $M(D) \not\models head(h)\theta$.*

Tafel 2.8 Clausal discovery.

Gegeben:	eine Theorie oder Datenbank E	
	eine Hypothesensprache L_h	
Gesucht:	eine Hypothese $H \subset L_h$, für die gilt:	
	$M(E) \models H$	(Gültigkeit)
	$\forall H' \subset L_h$: wenn $M(E) \models H'$, dann $H \models H'$	(Vollständigkeit)
	$\neg\exists h \in H : H \setminus \{h\} \models h$	(Minimalität)

Lernen aus Interpretationen

Beim Lernen aus Interpretationen [RD94, Rae97] gibt es positive und negative Beispiele wie in Tafel 2.7. Jedes Beispiel entspricht dabei einer Menge von Grundfakten, die direkt als Faktenmenge repräsentiert ist oder als Logikprogramm, aus dem die Grundfakten ableitbar sind. Das Lernen aus Interpretationen wurde im Rahmen der nichtmonotonen ILP entwickelt mit einer beschreibenden Aufgabenstellung vor Augen, erwies sich dann aber auch als nützlich für das Konzeptlernen und prädiktive Aufgabenstellungen. Zum Beispiel lernt das System ICL [DRVL95] eine minimale Mengen von Klauseln, die in allen positiven Beispielen wahr sind, von denen jede aber mindestens ein negatives Beispiel verletzt. Diese minimale Klauselmenge beschreibt die Daten, kann aber auch dazu dienen, die Klassenzugehörigkeit (positiv oder negativ) neuer Fälle vorherzusagen.

Die Anwendung der nichtmonotonen ILP zur Prädiktion verdeutlicht, wo in der nichtmonotonen ILP, bei der die Hypothesen ja deduktiv aus dem Gegebenen folgen, der induktive Sprung stattfindet. Die Hypothesen folgen zwar deduktiv aus jedem der gegebenen positiven Beispiele und dem Hintergrundwissen, wenn sie aber (explizit bei der Prädiktion oder implizit bei der Deskription) auf bisher unbekannte Fälle übertragen werden, bedeutet dies einen induktiven Schluß [Fla95].

2.3.3 Individuenzentrierte induktive Logikprogrammierung

Die meisten Algorithmen und Verfahren, die im maschinellen Lernen und Data Mining entwickelt wurden, realisieren sogenanntes Attribut–Wert–Lernen. Die Daten bestehen dabei aus einer Menge von Attribut–Wert–Vektoren, von denen jeder genau ein Beispiel beschreibt. Alle Attribut–Wert–Vektoren haben dieselbe Länge und betreffen dieselben Attribute der Beispiele, so daß sie sich zu einer einzigen Tabelle anordnen lassen. Die Hypothesen, die beim Attribut–Wert–Lernen induziert werden, beschreiben ein Beispiel ausschließlich mittels der Merkmale, die in dem ihm entsprechenden Attribut–Wert–Vektor aufgeführt sind. Beziehungen zwischen mehreren Beispielen werden nicht berücksichtigt. Die Ausdrucksstärke des Attribut–Wert–Lernens entspricht der Aussagenlogik; daher wird es auch als aussagenlogisches Lernen bezeichnet.

Von der Etablierung der Induktiven Logikprogrammierung als Forschungsrichtung an war es eine offene und vieldiskutierte Frage, ob es überhaupt nötig und sinnvoll sei, Lernverfahren im prädikatenlogischen Rahmen zu entwickeln und anzuwenden, zumal aussagenlogisches Lernen im allgemeinen effizienter ist. Auch waren anfänglich aussagenlogische Lernverfahren weiter entwickelt und für praktische Anwendungsprobleme wie beispielsweise verrauschte Daten besser angepaßt als die frühen ILP–Systeme. Als Alternative zu prädikatenlogischem Lernen schlug man daher vor, die einschlägigen Anwendungen auf aussagenlogische Probleme abzubilden und die dafür vorhandenen aussagenlogischen Lernverfahren einzusetzen. Dieses Vorgehen, also die Transformation von prädikatenlogischen Daten in aussagenlogische Daten, bezeichnet man als Propositionalisierung. Inzwischen versteht man die Beziehungen zwischen ILP und aussagenlogischem Lernen besser. Insbesondere beginnt man, sogenannte individuenzentrierte Datenrepräsentationen und Lernprobleme als Bindeglied zwischen Induktiver Logikprogrammierung und aussagenlo-

gischem Lernen zu verstehen [LF01, LF00, FL01, FL99, RD94, Dže95, Rae97, FL99]. Eine einheitliche, präzise Definition für individuenzentrierte Datenrepräsentationen und Lernprobleme hat sich jedoch noch nicht herausgebildet.

Individuenzentrierte Lernprobleme

Ein Lernproblem ist bestimmt durch Daten und Hypothesensprache, die aussagen- oder prädikatenlogisch sein können. Ein Lernproblem kann als individuenzentriert verstanden werden, wenn die Daten eine klar erkennbare Population von Individuen repräsentieren und die induzierten Hypothesen über die Individuen generalisieren, also Teilmengen der Individuenpopulation beschreiben. Attribut–Wert–Lernen ist offensichtlich individuenzentriert, viele typische ILP–Anwendungen, etwa das Lernen rekursiver Prädikatdefinitionen, das Lernen von Familienbeziehungen und Programmsynthese-Aufgaben wie die Induktion von Listenprädikaten, jedoch nicht. Zum Beispiel ist es beim Lernen von Listenprädikaten wie reverse/2 gleichermaßen problematisch, einzelne Listen oder Paare von Listen als die im Mittelpunkt des Lernproblems stehende Individuenpopulation zu betrachten. Die einzelnen Listen stehen nicht direkt im Zentrum des Lernproblems, da ja Beziehungen zwischen den Listen gelernt werden sollen. Die Listenpaare lassen nicht uneingeschränkt als zentralen Untersuchungsgegenstand betrachten, da diese Sichtweise ignoriert, daß die Eingabeliste die Ausgabeliste bestimmt [LF00].

Die Induktive Logikprogammierung erlaubt jedoch auch die Formulierung und Bearbeitung individuenzentrierter Lernprobleme. Die Daten können dazu in verschiedenen Formaten vorliegen. Datenrepräsentationen, die besonders auf individuenzentrierte Lernprobleme festgelegt sind, werden auch als individuenzentrierte Datenrepräsentationen bezeichnet [Fla99b]. Dabei ist die im Mittelpunkt stehende Individuenpopulation schon durch das Format der Daten vorgegeben. Die Bedeutung der individuenzentrierten Datenrepräsentationen und Lernprobleme für die ILP liegt darin, daß sie eine natürliche Erweiterung aussagenlogischer Repräsentationen und aussagenlogischen Lernens hin zur Prädikatenlogik erster Ordnung bilden.

Datenrepräsentationen für individuenzentrierte Lernprobleme

Attribut–Wert–Repräsentationen. In Attribut–Wert–Repräsentationen liegen die Daten in Form einer Tabelle vor. Jede Zeile der Tabelle beschreibt ein Individuum i anhand einer festen Anzahl von Attributen A_j. Ein Attribut A_j kann einen von mehreren vorgegeben Werten $\{a_{j,1}, \ldots, a_{j,n_j}\}$ annehmen. Die Attribute A_j entsprechen den Spalten der Tabelle.

Beispiel 21 *Gegeben sei eine phonetische Datenbank mit Daten zu Silben. Eine Attribut–Wert–Repräsentation der Daten zeigt Tafel 2.9. Jede Silbe ist beschrieben mit den Attributen Sid, Acc, Bnd, Swb, Swe, Exp_len, Mes_len und Lws. Das Schlüsselattribut Sid ist eindeutiger Identifikator für eine Silbe, Acc ist die Art des Satzakzents, den die Silbe trägt und bnd die Phrasengrenze. Das Attribut Swb gibt den Abstand einer Silbe zum Wortanfang als Anzahl der dazwischenliegenden Silben plus 1 an; Swe ist der Abstand der Silbe zum Wortende. Das numerische Attribut Exp_len ist die erwartete, Mes_len die gemessene Länge der Silbe. Das Attribut Lws hat den Wert yes, wenn im Lexikon ein Wortakzent für die Silbe verzeichnet ist, no sonst.*

Aus Sicht der Logik lassen sich Tupel oder Merkmalsvektoren, die die Individuen beschreiben, als Konjunktionen von Attribut–Wert–Literalen der Form $A_j = a_{j,k}$ betrachten. Dabei ist $a_{j,k}$ ein Wert aus dem Wertebereich des Attributs A_j. Da die Attribut–Wert–Literale Aussagenvariablen entsprechen, bezeichnet man das Lernen in Attribut–Wert–Repräsentationen auch als aussagenlogisches Lernen [Fla99b]. Die Hypothesen sind beim

Tafel 2.9 Eine Attribut–Wert–Repräsentation für Silben.

Sid	Acc	Bnd	Swb	Swe	Exp_len	Mes_len	Lws
$s1$	0	0	1	6	1.61	1.2	no
$s2$	0	0	2	5	1.36	0.8	no
$s3$	$'l * h'$	0	3	4	2.1	2.0	yes
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Attribut–Wert–Lernen ebenfalls Konjunktionen von Attribut–Wert–Literalen. Beim Attribut–Wert–Lernen gibt die Form der Daten die betrachteten Individuen vor: jeder Attribut-Wert–Vektor in den Daten entspricht genau einem Individuum. Daher ist es nicht nötig, die Individuen explizit zu referenzieren.

Attribut–Wert–Lernen läßt sich auch als ILP–Problem formulieren. Eine Attribut–Wert-Tabelle wird dann als Menge von Grundfakten der Form $individuum(o_i, a_{1,i}, \ldots, a_{n,i})$ dargestellt. Jede Grundfakt beschreibt ein Individuum und entspricht einer Zeile der Tabelle. In jedem Grundfakt ist o_i ein Identifikator des repräsentierten Individuums; $a_{j,i}$ ist der Wert des Attributs A_j, den das Individuum o_i aufweist. Die Grundfakten bilden zusammen die Definition des Prädikats $individuum/n+1$. Hypothesen eines solchen ILP–Problems haben die Form $individuum(O, A_1, A_2, \ldots, A_n)$, wobei die Variablen A_j mit festen Werten instanziiert sein können, zum Beispiel $A_2 = rot \wedge A_3 = groß$.

Multiple instance–**Repräsentationen.** In einer *multiple instance*–Repräsentation liegen die Daten ebenfalls in Form einer Tabelle von Merkmalsvektoren vor. Im Unterschied zum üblichen Attribut–Wert–Lernen besteht eine Individuumsbeschreibung aber nicht aus nur einem Merkmalsvektor, sondern aus einer Menge von Merkmalsvektoren, deren Anzahl von Objekt zu Objekt schwanken kann. Für eine *multiple instance*–Repräsentation gilt also nicht, daß genau ein Merkmalsvektor ein Individuum repräsentiert.

In dem von Dietterich et al. formulierten *multiple instance*–Problem ist die Klassenzugehörigkeit von *multiple instance*–repräsentierten Beispielen vorherzusagen, unter der Annahme, daß für jedes Beispiel ein einziger seiner Merkmalsvektoren genügt, um ihm die richtigen Klasse zuzuweisen [DL97]. Während beim Attribut–Wert-Lernen also lediglich die relevanten Merkmale innerhalb eines Merkmalsvektors zu bestimmen sind, ist es bei einem *multiple instance*–Problem Teil der Lernaufgabe, auch den jeweils richtigen Merkmalsvektor auszuwählen.

Beispiel 22 *Gegeben sei eine phonetische Datenbank mit Daten zu Silben. Jede Silbe ist beschrieben durch Attribute der Phoneme, aus denen sie besteht. Das Attribut Pid ist eindeutiger Identifikator für Phoneme, Sid ist der Identifikator der Silbe, zu der das Phonem gehört, Onc weist das Phonem als $Onset$, $Nukleus$ oder $Coda$ aus. Das Attribut $Phonem$ gibt die Art des Phonems an, zum Beispiel $langes\_a$, $kurzes\_a$, l usw. Das Attribut Typ gibt an, ob das Phonem zu den kurzen Vokalen, langen Vokalen, Reibelauten, Gleitlauten, Nasalen etc. gehört. Es soll über Silben gelernt werden. Jedes betrachtete Individuum ist also durch mehrere Zeilen der in Tafel 2.10 gezeigten Attribut–Wert–Tabelle beschrieben.*

Dietterich et al. haben das *multiple instance*-Problem als wichtige und echte Erweiterung des klassischen Attribut–Wert–Lernens erkannt, das in vielen Anwendungsfeldern auftritt und mit den gebräuchlichen Algorithmen zum Attribut–Wert–Lernen nicht optimal bearbeitet werden kann [DL97]. Es unterscheidet sich vom Attribut–Wert-Lernen dadurch, daß die Länge der Merkmalsbeschreibungen zwischen den Individuen variiert und die Beziehung zwischen einem Individuum und seinen Merkmalsvektoren nicht determiniert ist insofern, als nicht von vornherein festgelegt ist, aus welchem der Merkmalsvektoren eines Individuum seine Klassenzugehörigkeit vorhergesagt wird.

Tafel 2.10 Eine *multiple instance*–Repräsentation für Silben.

Pid	Sid	Onc	$Phonem$	Typ
$s1p1$	$s1$	$onset$	nc_m	$nasal$
$s1p2$	$s1$	$nucleus$	lv_icolon	$long$
$s2p1$	$s2$	$onset$	lc_l	$liquid$
$s2p2$	$s2$	$nucleus$	lv_icolon	$long$
$s3p1$	$s3$	$onset$	pc_t	$plosiv$
$s3p2$	$s3$	$nucleus$	lv_Ecolon	$long$
$s3p3$	$s3$	$coda$	lc_R	$liquid$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Tafel 2.11 Eine termbasierte Repräsentation für Silben.

$$
\begin{aligned}
s1 =&[acc(0), bnd(0), swe(6), exp\_len(1.61), mes\_len(1.2), lws(no),\\
&[(onset, nc\_m, nasal),\\
&(nucleus, lv\_icolon, long)]]\\
s2 =&[acc(0), bnd(0), swb(2), swe(5), exp\_len(1.36), mes\_len(0.8), lws(no),\\
&[(onset, lc\_l, liquid),\\
&(nucleus, lv\_icolon, long)]]\\
s3 =&[acc('l{*}h'), bnd(0), swb(3), swe(4), exp\_len(2.1), mes\_len(2), lws(yes),\\
&[(onset, pc\_t, plosiv),\\
&(nucleus, lv\_Ecolon, long),\\
&(coda, lc\_R, liquid)]]\\
&\vdots
\end{aligned}
$$

Termbasierte Repräsentationen. In den termbasierten Datenrepräsentationen, die Lachiche und Flach [LF00] vorschlagen, ist jedes Individuum durch einen Term beschrieben. Terme sind rekursiv definiert. Ein Term ist eine Konstante oder ein Tupel oder eine Menge oder Multi–Menge von Termen. (In einer Multi–Menge kann ein Objekt mehrfach vorkommen.) Die Mengen und Multi–Mengen ermöglichen eine variable Länge der Individuumsbeschreibungen und führen Nichtdeterminiertheit in die termbasierte Repräsentation ein. Eine termbasierte Individuumsbeschreibung, die nur Konstanten und keine Mengen oder Multi–Mengen enthält, ist gleichwertig zu einem Attribut–Wert–Vektor. Eine termbasierte Repräsentation, in der jede Individuumsbeschreibung aus einer Menge von Konstantentupeln besteht, entspricht einer *multiple instance*–Repräsentation.

Beispiel 23 *In Tafel 2.11 ist eine termbasierte Repräsentation für Silben zu sehen. Die Terme, die die Silben beschreiben, enthalten Listen variabler Länge mit Merkmalstupeln, die Merkmale der zur Silbe gehörigen Phoneme beschreiben.*

Interpretationen-Repräsentationen. In Interpretationen-Repräsentationen, die für das im Abschnitt 2.3.2 beschriebene Lernen aus Interpretationen verwendet werden, ist jedes Individuum beschrieben durch eine Menge von Grundfakten oder durch eine Theorie, deren (Herbrand–)Modell als Beschreibung des Individuums angenommen wird. Die Ausdruckskraft jedes Grundfakts entspricht der einer Aussagenvariablen. Im Unterschied zu einer

Tafel 2.12 Eine Interpretationen-Repräsentation für Silben.

$$
\begin{aligned}
s1 =\{&acc(0), bnd(0), swe(6), exp\_len(1.61), mes\_len(1.2), lws(no),\\
&phon(s1p1, onset, nc\_m, nasal), phon(s1p2, nucleus, lv\_icolon, long),\\
&follows(s1p1, s1p2)\}\\
s2 =\{&acc(0), bnd(0), swb(2), swe(5), exp\_len(1.36), mes\_len(0.8), lws(no),\\
&phon(s2p1, onset, lc\_l, liquid), phon(s2p2, nucleus, lv\_icolon, long),\\
&follows(s2p1, s2p2)\}\\
s3 =[&acc('l*h'), bnd(0), swb(3), swe(4), exp\_len(2.1), mes\_len(2), lws(yes),\\
&phon(s3p1, onset, pc\_t, plosiv), phon(s3p2, nucleus, lv\_Ecolon, long),\\
&phon(s3p3, coda, lc\_R, liquid),\\
&follows(s3p1, s3p2), follows(s3p2, s3p3)\}\\
&\vdots
\end{aligned}
$$

aussagenlogischen Repräsentation erlaubt eine Interpretationenrepäsentation aber Individuenbeschreibungen unterschiedlicher Länge und ist nicht-deterministisch, weil jede Individuumsbeschreibung mehrere Fakten über demselben Prädikat enthalten kann, von denen eines oder mehrere nicht-deterministisch zur Beschreibung eines Individuums ausgewählt werden können. Die Beschreibungen der verschiedenen Individuen sind voneinander und vom allen Individuen gemeinsamen Hintergrundwissen B getrennt; jedes Individuum wird als eigenständige, von den anderen Individuen unabhängige Datenbank behandelt.

Beispiel 24 *Tafel 2.12 zeigt eine Interpretationen-Repräsentation für die Silben, die auch Informationen über die Phoneme jeder Silbe enthält. Auch die Vorgänger-/Nachfolger-Beziehung follows/2 zwischen Phonemen läßt sich hier einfach repräsentieren.*

Relationale Datenrepräsentationen. In relationalen Datenrepräsentationen liegen die Daten wie in Abschnitt 2.2.3 beschrieben in Form mehrerer Relationen vor. Eine Relation ist eine Menge von Grundfakten über einem Prädikat. Eine relationale Datenbank besteht aus mehreren Tabellen, die zueinander in Beziehung stehen können. Auch Datenrepräsentationen, die zusätzlich zu den Relationen intensional definiertes, nicht-rekursives Hintergrundwissen enthalten, bezeichnet man als relational, da nicht-rekursives Hintergrundwissen die Ausdrucksstärke nicht wesentlich erhöht.

Beispiel 25 *Tafel 2.13 zeigt ein Beispiel für eine relationale Repräsentation der phonetischen Daten. Die Daten sind in zwei Relationen organisiert. Die Relation $silbe$ beschreibt Silben anhand der in Beispiel 21 genannten Attribute. Die Relation $phon$ beschreibt Phoneme mittels der Attribute aus der* multiple instance*–Repräsentation in Beispiel 22. Als weitere Attribute sind für jede Silbe die in der sprachlichen Äußerung vorangehende und die nachfolgende Silbe aufgeführt, ebenso für jedes Phonem sein Vorgänger und sein Nachfolger.*

In dieser Datenbank gibt es eine nicht-determinierte Beziehung von Silben zu Phonemen, denn ein Silbenidentifikator kann beliebig oft in der Tabelle $phon$ vorkommen. Außerdem bestehen rekursive Beziehungen innerhalb der Relation $silbe$, in denen eine Silbe auf ihre Vorgänger- und ihre Nachfolger-Silbe verweist. Entsprechende rekursive Beziehungen bestehen in der $phon$-Tabelle.

Tafel 2.13 Eine relationale Repräsentation für Silben.

$$silbe(s1, 0, 0, -, s2, 1, 6, 1.61, 1.2, no).$$
$$silbe(s2, 0, 0, s1, s3, 2, 5, 1.36, 0.8, no).$$
$$silbe(s3, 'l*h', 0, s2, s4, 4, 2.1, 2.0, yes).$$
$$\vdots$$

$$phon(s1p1, s1, -, s1p2, onset, nc\_m, nasal).$$
$$phon(s1p2, s1, s1p1, s2p1, nucleus, lv\_icolon, long).$$
$$phon(s2p1, s2, s1p2, s2p2, onset, lc\_l, liquid).$$
$$phon(s2p2, s2, s2p1, s3p1, nucleus, lv\_icolon, long).$$
$$phon(s3p1, s3, s2p2, s3p2, onset, pc\_t, plosiv).$$
$$phon(s3p2, s3, s3p1, s3p3, nucleus, lv\_Ecolon, long).$$
$$phon(s3p3, s3, s3p2, s4p1, coda, lc\_R, liquid).$$
$$\vdots$$

Diskussion. Die Datenrepräsentationen unterscheiden sich in ihrer Ausdruckskraft, also darin, welche Arten von Informationen sich leicht verständlich und kompakt darstellen lassen, und auch hinsichtlich der Effizienz, mit der ein Lernverfahren auf den repräsentierten Daten arbeitet. Die vorangangene Übersicht über individuenzentrierte Datenrepräsentationen, die sich an [LF00] orientiert, ist geordnet nach zunehmender Ausdrucksstärke und Flexibilität. Mit Ausnahme der Attribut–Wert–Repräsentation lassen alle beschriebenen Repräsentationsmöglichkeiten Nichtdeterminiertheit zu. Damit übersteigt ihre Ausdrucksfähigkeit die der aussagenlogischen Attribut–Wert–Repräsentationen wesentlich; sie lassen sich also der ILP zuordnen.

In Attribut–Wert–Repräsentationen, *multiple instance*–Repräsentationen, termbasierten und Interpretationen–Repräsentationen ist die Individuenpopulation, über die gelernt wird, durch das Format der Daten vorgegeben. Sie können insofern als individuenzentrierte Datenrepräsentationen bezeichnet werden. Relationale Datenrepräsentationen sind dagegen nicht im engeren Sinne individuenzentriert, da sie mehrere Individuenpopulationen beschreiben können. Die Individuenpopulation, über die gelernt werden soll, wird nicht durch das Format der Daten festgelegt, sondern muß explizit angegeben werden, indem eine Hypothesensprache gewählt wird, in der alle Hypothesen die im Mittelpunkt der Lernaufgabe stehenden Individuen beschreiben.

Diese Flexibilität und Anpaßbarkeit ist ein Vorteil der relationalen Datenrepräsentation, da sich die Daten durch entsprechende Deklarationen auf verschiedene Individuenpopulationen ausrichten lassen, ohne daß dazu, wie etwa bei termbasierten oder Interpretationen–Repräsentationen, die Daten transformiert werden müssen. Andererseits können termbasierte und Interpretationen–Repräsentationen den relationalen Repräsentationen hinsichtlich der Effizienz des Lernens überlegen sein, weil sie die zu einem Individuum gehörende Information kapseln und nicht erst während des Lernens zusammenführen müssen, etwa durch teure Verbundberechnungen.

Propositionalisierung

Alle individuenzentrierten Lernprobleme lassen sich prinzipiell propositionalisieren. Die Ausdrucksfähigkeit eines individuenzentrierten Lernproblems geht über die Aussagenlogik

Tafel 2.14 Eine Attribut–Wert–Repräsentation für Silben mit Merkmalen der Phoneme.

Sid	*P1_onc*	*P1_typ*	*P2_onc*	*P2_typ*	*P3_onc*	*P3_typ*	...
s1	*onset*	*nasal*	*nucl.*	*long*	—	—	...
s2	*onset*	*liquid*	*nucl.*	*long*	—	—	...
s3	*onset*	*plosiv*	*nucl.*	*long*	*coda*	*liquid*	...
⋮							

hinaus und ist damit echt prädikatenlogisch, wenn Daten und Hypothesensprache nichtdeterminierte oder rekursive Verbindungen zwischen Dateneinheiten enthalten [Fla99a, DR98]. Solche Lernprobleme lassen sich meist nicht mehr unmittelbar propositionalisieren [DR98, FL99]. Die Schwierigkeiten, die bei der Propositionalisierung auftreten können, werden am Beispiel zweier wenig ausgefeilter Propositionalisierungsmethoden verdeutlicht.

Universelle Relation. Eine Möglichkeit, mehr-relational repräsentierte Daten zu propositionalisieren, ist die Berechnung einer Universellen Relation (UR) [Ull82]. Darunter versteht man eine einzige Verbundrelation, die alle Relationen einer Datenbank beinhaltet. Das *multiple instance*-Problem zeigt, daß es nicht genügt, alle relevanten Daten zu einer einzigen Tabelle zusammenzufassen, um aussagenlogische Lernverfahren erfolgreich darauf anwenden zu können, sondern daß eine Tabelle gefordert ist, in der jedes Individuum genau einer Zeile entspricht. Um eine UR für ein individuenzentriertes Lernproblem zu berechnen, müssen — vereinfacht gesagt — die Tupel aus anderen Relationen, die zu einem Individuum i oder zu einem mit i verbundenen Tupel in Beziehung stehen, als zusätzliche Merkmale dieses Individuums i in die ihm entsprechende Zeile der UR eingetragen werden. Nichtdeterminierte und rekursive Verbindungen in den Daten führen dabei zu Problemen. Erstens ergeben sie unterschiedlich viele Merkmale für die verschiedenen Individuen. Da das Tabellen-Format eine einheitliche Länge für alle Merkmalsvektoren erzwingt, bestimmt die maximale Anzahl von Merkmalen eines Individuums die Anzahl der Attribute der UR. Diese maximale Anzahl von Merkmalen ist aber nicht für alle Individuen gegeben, so daß die UR viele Lücken enthält. Zweitens können Tupel zu verschiedenen Individuen in Beziehung stehen und müssen entsprechend oft in die UR aufgenommen werden. Daten mit nicht–determinierten oder rekursiven Verbindungen ergeben damit im allgemeinen eine UR mit so vielen redundanten oder unbekannten Werten, daß ihre Berechnung nicht mehr vernünftig erscheint. Dies wurde schon in Beispiel 4 angedeutet.

Ein weiteres bekanntes Problem, das bei der Repräsentation der Daten in einer UR auftritt, ist, daß sich viele mögliche Regelmäßigkeiten und Zusammenhänge in den Daten nur sehr umständlich repräsentieren lassen.

Beispiel 26 *Tafel 2.14 zeigt ein gekürztes Beispiel für eine Attribut–Wert–Repräsentation der Silben, die die Merkmale der Phoneme beinhaltet. Die Attribut–Wert–Tabelle muß Attribute für die maximale Anzahl von Phonemen anlegen, die in einer Silbe vorkommen können. Für Silben mit weniger Phonemen enthält die Tabelle den Wert „—“ (unbekannt). Das Konzept „Silben, die einen Nasal enthalten“ muß mit dieser Datenrepräsentation so dargestellt werden:*

$$P1\_typ = nasal \vee P2\_typ = nasal \vee P3\_typ = nasal$$

Tafel 2.15 zeigt ein gekürztes Beispiel für eine Attribut–Wert–Repräsentation der Phoneme, die die Merkmale der Silben beinhaltet. Die Tabelle wiederholt die Merkmale einer Silbe für jedes ihrer Phoneme.

Tafel 2.15 Eine Attribut–Wert–Repräsentation für Phoneme mit Merkmalen der Silben.

Pid	Sid	Acc	Bnd	Swb	Swe	Exp_len	Mes_len	Lws
s1p1	$s1$	0	0	1	6	1.61	1.2	no
s1p2	$s1$	0	0	1	6	1.61	1.2	no
s2p1	$s2$	0	0	2	5	1.36	0.8	no
s2p2	$s2$	0	0	2	5	1.36	0.8	no
s3p1	$s3$	$'l*h'$	0	3	4	2.1	2.0	yes
s3p2	$s3$	$'l*h'$	0	3	4	2.1	2.0	yes
s3p3	$s3$	$'l*h'$	0	3	4	2.1	2.0	yes
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Propositionalisierung auf Ebene der Hypothesen. Eine weitere Möglichkeit [DR98], prädikatenlogisch repräsentierte Daten mit einer zentralen Individuenpopulation in Attribut–Wert–Form zu bringen, besteht darin, eine Attribut–Wert–Tabelle zu generieren, die für jede Hypothese der prädikatenlogischen Hypothesensprache (die dazu natürlich endlich sein muß) ein binäres Attribut enthält. Jede Zeile dieser Tabelle entspricht einem Individuum. In ein Feld der Tabelle wird *true* eingetragen, wenn die der jeweiligen Spalte entsprechende Hypothese das der Zeile entsprechende Individuum abdeckt, und sonst *false*. Da ILP–Hypothesenräume typischerweise wegen der Ausdrucksfähigkeit der Hypothesensprache sehr groß sind, ergibt diese Transformation eine sehr große Tabelle. Die Berechnung der Attribut–Wert–Tabelle verursacht großen Aufwand, während das eigentliche Lernen zum Erzeugen-und-Testen von Hypothesen (also Attributen) degeneriert [DR98].

Die Entwicklung besserer, praktisch nützlicher Propositionalisierungsverfahren ist ein Teilgebiet der ILP. Einen Überblick über aktuelle Ansätze gibt [KLF01]. Aus praktischer Sicht ist es in manchen Fällen dennoch vorteilhaft, die ursprüngliche prädikatenlogische Datenrepräsentation beizubehalten und ILP–Lernverfahren einzusetzen. Dabei erlaubt die Beschränkung auf individuenzentrierte Repräsentationen und Lernprobleme, ILP–Systeme zu entwickeln, die fortgeschrittene Algorithmen und Techniken vom aussagenlogischen Attribut–Wert–Lernen auf das prädikatenlogische Lernen übertragen. Beispiele dafür sind die Induktion von logischen Entscheidungsbäumen [BR98], Bayes'sche Klassifikation auf relationalen Daten [FL99], das Entdecken von ungewöhnlichen Teilgruppen [Wro97] oder Assoziationsregeln [Deh98] in multi-relationalen Datenbanken und auch die vorliegende Arbeit.

2.3.4 Nutzen der ILP für KDD und Data Mining

Die ILP bietet viele Vorteile für KDD und Data Mining [Dže96b, Fay97]. Der vordergründigste ist, daß die ILP–Verfahren direkt auf die weitverbreiteten (multi)-relationalen Datenbanken anwendbar sind und keine Transformation in Attribut–Wert–Format erfordern. ILP–Verfahren repräsentieren das entdeckte Wissen in symbolischer Form, typischerweise unter Verwendung des Vokabulars, mit dem auch die Daten beschrieben sind. Diese Übereinstimmung erleichtert es, die Lernergebnisse in der Terminologie der jeweiligen KDD-Anwendung auszudrücken, und trägt so zur angestrebten Verständlichkeit bei. Zwischen der Datenzugriffssprache SQL, die von vielen RDBMS angeboten wird, und den prädikatenlogischen Hypothesensprachen der ILP besteht eine wohlbekannte und fundierte Entsprechung, die den Übergang vom Datenspeicher zum Lernverfahren für ILP–Ansätze sehr vereinfacht. Die Kopplung zwischen Datenbank und Lernverfahren kann so weit gehen, daß das Lernverfahren direkt auf der Datenbank operiert. Auch die traditionelle Fähigkeit der ILP–Verfahren, Hintergrundwissen in den Lern- und Entdeckungsprozeß einzube-

ziehen, ist für die KDD von besonderer Bedeutung.

Da ILP–Hypothesenräume durch die Ausdruckskraft der Prädikatenlogik sehr groß werden können, befaßt sich die ILP–Forschung auch explizit damit, Mechanismen zu entwickeln, mit denen sich die Hypothesensprache eines Lernsystems gezielt auf die Anforderungen einer Anwendung zuschneiden läßt. Eine solche Möglichkeit, die Hypothesensprache eines Lern- oder Entdeckungssystem für jede Anwendung deklarativ und veränderbar festzulegen, bezeichnet man als deklarativen Sprachbias[2] [WMJ00]. Für die KDD ist ein deklarativer Sprachbias besonders angemessen, da er erlaubt, die Suche auf interessantes und potentiell nützliches Wissen zu lenken, und damit das Bestreben der KDD unterstützt, den KDD–Prozeß auf die spezifischen Ziele einer Anwendung auszurichten. In den Voreinstellungen der Hypothesensprache drückt sich ebenfalls eine Art von Hintergrundwissen aus. Es kann dazu genutzt werden, den Wissensentdeckungsprozeß effizienter und erfolgreicher zu machen und den Anteil nutzloser und uninteressanter Entdeckungen unter den Resultaten einer Data Mining-Anwendung zu verringern, die diese sonst typischerweise in großer Zahl liefert. Schließlich ermöglichen die Ausdrucksfähigkeit und Flexibilität der in der ILP verwendeten Hyothesensprachen, durch geeignete Sprachdeklarationen verschiedene spezifische Data Mining–Aufgaben zu realisieren und sie innerhalb eines einzigen ILP–basierten Data Mining-Systems zu bearbeiten. Dies kommt dem explorativen Data Mining entgegen [DT99].

2.4 Teilgruppenanalyse im Rahmen der ILP

Die Teilgruppenanalyse ist eine allgemeine Aufgabenstellung von Data Mining und KDD. Sie ist formal beschrieben in Tafel 2.2. Gegeben sind eine Individuenpopulation Pop, eine Hypothesensprache L_h, eine Interessantheitsfunktion d und ein Akzeptanzkriterium acc. Die Hypothesen beschreiben Teilgruppen der Population Pop. Die Interessantheit der Hypothesen wird bewertet mit einer Funktion d. Das Akzeptanzkriterium legt das Ziel der Teilgruppenanalyse fest: die k interessantesten Hypothesen aus L_h zu finden, oder, alle Hypothesen zu bestimmen, deren Interessantheit einen bestimmten Schwellwert erreicht. Dieser Abschnitt formalisiert die Teilgruppenanalyse im Rahmen der individuenzentrierten nicht-monotonen ILP und beschreibt einen grundlegenden Algorithmus zur Suche nach interessanten Teilgruppen.

2.4.1 Definition von Individuen und Teilgruppen

Die Teilgruppenanalyse wird hier auf eine Individuenpopulation angewendet, die in einer relationalen Datenbank D repräsentiert ist. Wie in Abschnitt 2.3.3 erläutert, ist in einer relationalen Datenrepräsentation die Individuenpopulation nicht durch das Datenformat vorgegeben, sondern muß für die jeweilige Aufgabe geeignet festgelegt werden.

Grundpopulation

Die Individuenpopulation Pop wird hier definiert als die Menge der Grundinstanzen $Pop = (V_1, \ldots, V_n)\sigma$ einer geordneten Menge von Variablen $(V_1, \ldots, V_n)$, die in einem Literal pop vorkommen, so daß $pop\sigma$ in der Datenbank D gültig ist. Das Literal pop heißt Populationsliteral, die Variablen $V_1, \ldots, V_n$ heißen Individuenvariablen. Alle Nicht-Individuenvariablen in pop sind existenzquantifiziert.

Definition 9 *Gegeben seien eine Datenbank D, ein Literal pop sowie eine Menge von Variablen $(V_1, \ldots, V_n)$, die in pop vorkommen. Die Population Pop ist die Menge aller*

[2] „Bias“ bedeutet hier etwa Voreinstellung. Bisher hat sich noch kein gängiger deutscher Begriff für Bias und Sprachbias herausgebildet.

Grundinstanzen $(V_1, \ldots, V_n)\sigma$ der Individuenvariablen $\{V_1, \ldots, V_n\}$, für die $pop\sigma$ in der Datenbank D wahr ist:

$$Pop = \{(V_1 \ldots, V_n)\sigma \mid D \models \exists(pop\sigma) \text{ und } (V_1 \ldots, V_n)\sigma \text{ ist grund}\}$$

Beispiel 27 *Gegeben ist eine phonetische Datenbank $Phon$, die Silben, Phoneme und Wörter in sprachlichen Äußerungen beschreibt. Die Datenbank besteht aus drei Relationen syl, $word$ und $phon$.*

Das Relationenschema der Relation syl ist $syl(Sid, Acc, Len)$. Das Schlüsselattribut Sid ist ein eindeutiger Identifikator für Silben, Acc gibt an, ob die Silbe einen Satzakzent trägt und gegebenenfalls die Art des Satzakzents, Len beschreibt die gemessene Länge der Silbe. Die Population $Sids$ der Silben in der Datenbank wird definiert als die Menge aller Grundinstanzen $(Sid)\sigma$ der Variable Sid im Literal $syl(Sid, Acc, Len)$, so daß $Phon \models syl(Sid, Acc, Len)\sigma$, also,

$$Sids = \{(Sid)\sigma | Phon \models syl(Sid, Acc, Len)\sigma \text{ und } Sid\sigma \text{ ist grund}\}$$

Das Populationsliteral ist hier $syl(Sid, Acc, Len)$. Die einzige Individuenvariable ist Sid.

Teilgruppen und Hypothesen

Eine Teilgruppe ist eine Teilmenge der Individuenpopulation Pop. Eine Teilgruppe wird beschrieben durch eine Konjunktion aus dem Literal pop, das die Individuenpopulation definiert, und einem weiteren logischen Ausdruck h, der die Individuenpopulation auf eine Teilmenge einschränkt. Die Konjunktion $pop \wedge h$ wird im folgenden als Teilgruppenbeschreibung bezeichnet, h als die eigentliche Hypothese, da das populationsdefinierende Literal pop fest vorgegeben ist und in jeder Teilgruppenbeschreibung vorkommen muß. Teilgruppenbeschreibungen und Hypothesen sind Konjunktionen von Literalen, $pop \wedge l_1 \wedge \ldots \wedge l_m$ beziehungsweise $l_1 \wedge \ldots \wedge l_m$. Sie werden auch als Literalmengen $\{pop, l_1, \ldots, l_m\}$ beziehungsweise $\{l_1, \ldots, l_m\}$ geschrieben.

Die Abdeckung einer Hypothese ist in Bezug auf die Individuenvariablen definiert.

Definition 10 *Die Individuenabdeckung oder Extension $cov_{V_1,\ldots,V_n}(h)$ einer Hypothese h ist die Menge der Individuen $(V_1, \ldots, V_n)\sigma$, für die $\exists(pop \wedge h)\sigma$ in der Datenbank D gilt. Für $cov_{V_1,\ldots,V_n}(h)$ wird vereinfachend auch $cov(h)$ geschrieben.*

$$cov_{V_1,\ldots,V_n}(h) = \{(V_1, \ldots, V_n)\sigma | D \models \exists(pop \wedge h)\sigma \text{ und } (V_1, \ldots, V_n)\sigma \text{ ist grund}\}.$$

Alle Variablen außer den Individuenvariablen sind existenzquantifiziert. Die Individuenabdeckung oder Extension einer Teilgruppenbeschreibung $pop \wedge h$ ist gleich der Extension der Hypothese h, also $cov(h \wedge pop) = cov(h)$. Eine Teilgruppe $cov(h)$ wird auch geschrieben als H.

Mit diesen Definitionen ist der Ansatz in der nichtmonotonen ILP angesiedelt.

Beispiel 28 *Gegeben sei die phonetische Datenbank $Phon$ aus Beispiel 27 mit der Individuenpopulation*

$$Syls = \{Sid\sigma | Phon \models syl(Sid, Acc, Len)\sigma \text{ und } \sigma \text{ grund}\}.$$

Die Teilgruppe der Population, die aus den langen Silben besteht, wird beschrieben durch die Teilgruppenbeschreibung $syl(Sid, Acc, Len) \wedge lang(Len)$. Die Teilgruppe der langen Silben ist

$$LSids = \{(Sid)\sigma | Phon \models (syl(Sid, Acc, Len) \wedge lang(Len))\sigma \text{ und } Sid\sigma \text{ ist grund.}\}$$

Zielgruppe

Es gibt viele Möglichkeiten, die Interessantheit von Teilgruppen zu bewerten. Ein grundlegendes und gebräuchliches Interessantheitskriterium ist die Häufigkeit einer Teilgruppe, also die Anzahl der darin enthaltenen Individuen. Wenn beispielsweise Beschreibungen oder Zusammenfassungen der Daten gesucht werden, können besonders häufige Teilgruppen interessant sein. Besteht die Aufgabe dagegen darin, Ausreißer oder Außenseiter zu entdecken, sind seltene Teilgruppen interessanter. In vielen Anwendungen wird die Interessantheit einer Teilgruppe auch bezüglich einer Referenzgruppe beurteilt. Je nach Zielsetzung der Anwendung kann eine besonders hohe oder zusätzlich auch eine besonders geringe Übereinstimmung einer Teilgruppe mit der Zielgruppe als interessant gelten. Die Zielgruppe T ist eine Teilgruppe der Individuenpopulation.

Definition 11 *Die Zielgruppe T wird definiert durch eine Teilgruppenbeschreibung $pop \wedge t$. Der logische Ausdruck t heißt Zieleigenschaft.*

$$T = \{(V_1 \ldots, V_m)\sigma \mid D \models \exists(pop \wedge t)\sigma \text{ und } (V_1 \ldots, V_m)\sigma \text{ ist grund}\}$$

Beispiel 29 *Eine mögliche Zielgruppe zur Phonetik-Datenbank $Phon$ mit der Population $Sids$ aus Beispiel 27 ist die Menge der Silben, die einen Wortakzent tragen. Diese Zielgruppe T wird definiert durch die Teilgruppenbeschreibung $syl(Sid, Acc, Len) \wedge Acc \neq 0$:*

$$T = \{(Sid)\sigma | Phon \models (syl(Sid, Acc, Len) \wedge Acc \neq 0)\sigma \text{ und } Sid\sigma \text{ ist grund.}\}$$

2.4.2 Ordnung von ILP-Hypothesenräumen

θ–Subsumtion zwischen Klauseln

In der ILP haben Hypothesen meist die Form von Klauseln. Klauseln sind Disjunktionen von Literalen. Sie werden auch als Literalmengen dargestellt. Die natürliche Strukturierung des ILP-Hypothesenraums ist die logische Implikation. Danach ist eine Klausel c_1 allgemeiner als eine Klausel c_2, wenn c_2 aus c_1 folgt, also $c_1 \models c_2$. Die logische Implikation zwischen Klauseln ist im allgemeinen aber nicht entscheidbar, auch nicht zwischen Hornklauseln [MDR94]. Deshalb verwenden die meisten ILP-Systeme anstelle der Implikation die etwas schwächere θ–Subsumtion, um den Hypothesenraum zu strukturieren. Die θ–Subsumtion wurde von Plotkin [Plo70, Plo71] eingeführt. Sie bietet ein syntaktisches Kriterium für Allgemeiner-Beziehungen zwischen Klauseln.

Definition 12 (θ–Subsumtion) *Es seien c_1 und c_2 Klauseln. c_1 θ–subsumiert c_2, geschrieben $c_1 \succ_\theta c_2$, wenn es eine Substitution θ gibt, so daß $c_1\theta \subseteq c_2$.*
c_1 θ–subsumiert c_2 echt genau dann, wenn $c_1 \succ_\theta c_2$ und $c_2 \not\succ_\theta c_1$.
Zwei Klauseln c_1 und c_2 sind äquivalent, geschrieben $c_1 \approx_\theta c_2$, wenn sie sich gegenseitig subsumieren, also wenn gilt: $c_1 \succ_\theta c_2$ und $c_2 \succ_\theta c_1$.
Eine Klausel c_1 ist nicht-reduziert, wenn sie zu einer ihrer echten Teilmengen äquivalent ist, also wenn gilt $c_3 \subset c_1$ und $c_3 \neq c_1$ und $c_3 \approx_\theta c_1$.

Beispiel 30

$$\begin{array}{lll}
c_1 & = & \{p(X), \neg q(X,Y)\} \\
c_2 & = & \{p(X), \neg q(X,Y), \neg r(Y,Z)\} \\
c_3 & = & \{p(X), \neg q(X,X)\} \\
c_4 & = & \{p(X), \neg q(X,Y), \neg q(X,Z)\} \\
c_1 \succ_\theta c_2, & weil & c_1 \subset c_2 \\
c_1 \succ_\theta c_3, & weil & c_1\theta = c_3 \text{ mit } \theta = [Y/X] \\
c_1 \approx_\theta c_4 & weil & c_1 \subset c_4 \text{ und } c_4\theta \subseteq c_1 \text{ für } \theta = [Z/Y]
\end{array}$$

Wenn eine Klausel c_1 eine Klausel c_2 θ–subsumiert, so impliziert sie diese auch: wenn $c_1 \succ_\theta c_2$, dann $c_1 \models c_2$. Die umgekehrte Aussage gilt nicht für selbst-rekursive Klauseln; aus $c_1 \models c_2$ folgt also nicht $c_1 \succ_\theta c_2$. Damit ist die θ–Subsumtion schwächer als die Implikation, das heißt, nicht alle Implikationsbeziehungen zwischen Klauseln lassen sich mit dem Kriterium der θ–Subsumtion erkennen.

Beispiel 31

$$\begin{aligned} c_1 &= nat(s(X)) \leftarrow nat(X) \\ c_2 &= nat(s(s(Y))) \leftarrow nat(Y) \end{aligned}$$

Hier folgt c_2 logisch aus c_1, also $c_1 \rightarrow c_2$, aber es gilt nicht $c_1 \succ_\theta c_2$ [Mug95].

Im Gegensatz zur Implikation ist die θ–Subsumtion zwischen zwei Klauseln entscheidbar. Im allgemeinen Fall erfordert es aber exponentiellen Aufwand, festzustellen, ob zwei Klauseln in einer θ–Subsumtionsbeziehung zueinander stehen. Für Spezialfälle gibt es Algorithmen, die θ–Subsumtionsbeziehungen zwischen Klauseln effizient berechnen [KL94].

θ-Subsumtion zwischen Teilgruppenbeschreibungen

In einer individuenzentrierten Anwendung sind die Hypothesen und Teilgruppen natürlicherweise geordnet über die extensionale Subsumtion. Eine Hypothese h_1 ist danach allgemeiner als eine Hypothese h_2, wenn $cov(h_2)$ eine Teilmenge von $cov(h_1)$ ist.

Definition 13 (Extensionale Subsumtion) *Eine Hypothese h_1 subsumiert eine Hypothese h_2 extensional, geschrieben $h_1 \succ_e h_2$, wenn h_1 alle Individuen abdeckt, die auch h_2 abdeckt, also*

$$h_1 \succ_e h_2 \text{ wenn } cov(h_2) \subseteq cov(h_1).$$

Die θ–Subsumtion ist definiert für Klauseln, also für Disjunktionen von Literalen. Diese werden auch als Literalmengen geschrieben. Teilgruppenbeschreibungen, die ebenfalls als Literalmengen repräsentiert werden können, sind Konjunktionen von Literalen. Eine der θ–Subsumtion für Klauseln entsprechende Relation läßt sich auch für Teilgruppenbeschreibungen definieren. Die θ–Subsumtion bietet für Teilgruppenbeschreibungen ein syntaktisches Kriterium für extensionale Subsumtion.

Definition 14 *Eine Teilgruppenbeschreibung h θ–subsumiert eine Teilgruppenbeschreibung h', geschrieben $h \succ_\theta h'$, wenn es eine Substitution θ gibt, die die Individuenvariablen nicht verändert, so daß $h\theta \subseteq h'$.*
Wenn $h \succ_\theta h'$, heißt h allgemeiner als h', und h' heißt spezieller als h.

Proposition 1 *Wenn eine Teilgruppenbeschreibung h eine Teilgruppenbeschreibung h' θ–subsumiert, dann ist h auch allgemeiner als h' bezüglich der extensionalen Subsumtion, das heißt, wenn $h \succ_\theta h'$, dann $cov(h) \supseteq cov(h')$.*

Beweis 1 *Aus der Definition folgt: wenn $h \succ_\theta h'$, gibt es eine Substitution θ, so daß $h\theta \subseteq h'$. Das heißt, es gibt eine Substitution θ und Literale $l_i, \ldots, l_j$, so daß $h' = h\theta \wedge l_i \wedge \ldots \wedge l_j$.*
Also ist zu zeigen: $cov(h) \supseteq cov(h\theta \wedge l_i \wedge \ldots \wedge l_j)$.
Laut Definition:

$$cov_{V_1,\ldots,V_n}(k) = \{(V_1, \ldots, V_n)\sigma | D \models \exists((k)\sigma) \text{ und } (V_1, \ldots, V_n)\sigma \text{ grund}\}.$$

(1) $cov(k) \supseteq cov(k \wedge l)$, *weil*
wenn $D \models \exists((k \wedge l)\sigma)$, *dann* $D \models \exists(k\sigma)$.

(2) $cov(k) \supseteq cov(k\theta), weil$
wenn $D \models \exists(k\theta\sigma)$ *dann* $D \models \exists(k\sigma)$.

Aus (1) und (2) folgt die Behauptung:

$$cov(h) \overset{(2)}{\supseteq} cov(h\theta) \overset{(1)}{\supseteq} cov(h\theta \wedge l_i) \overset{(1)}{\supseteq} \ldots \overset{(1)}{\supseteq} cov(h\theta \wedge l_i \wedge \ldots \wedge l_j) = cov(h')$$

$\diamondsuit$

Weil die θ–Subsumtion für Teilgruppenbeschreibungen ein syntaktisches Kriterium für die extensionale Subsumtion zwischen Teilgruppenbeschreibungen bietet, sind auch die auf der θ–Subsumtion basierenden Suchmethoden und Verfeinerungsoperatoren der ILP geeignet und angemessen für die Suche von Teilgruppenbeschreibungen. Die folgenden Bemerkungen zur θ–Subsumtion und darauf beruhenden Spezialisierungsoperatoren gelten für Teilgruppenbeschreibungen wie für Klauseln. Suchräume, die aus Teilgruppenbeschreibungen oder aus Klauseln bestehen und durch θ–Subsumtion geordnet sind, werden im folgenden als ILP-Suchräume bezeichnet.

2.4.3 Suche im ILP-Hypothesenraum

Spezialisierungsoperatoren

Lern- und Entdeckungssysteme implementieren die Suche im Hypothesenraum oft als die Anwendung eines Verfeinerungsoperators. Abhängig von der Suchrichtung erzeugt der Verfeinerungsoperator aus einer gegebenen Hypothese speziellere oder allgemeinere Hypothesen. Durch systematische rekursive Anwendung des Verfeinerungsoperators auf die neu erzeugten Hypothesen wird der gesamte Hypothesenraum durchlaufen. Ein Spezialisierungsoperator ist ein Verfeinerungsoperator, der den Hypothesenraum vom Allgemeinen zum Speziellen hin durchläuft. Er startet mit der oder den allgemeinsten Hypothesen im Hypothesenraum und generiert davon ausgehend immer speziellere Hypothesen.

Definition 15 *Es sei L_h eine Menge, auf der die Halbordnung $\succ$ definiert ist. Ein Spezialisierungsoperator für $(L_h, \succ)$ ist eine Funktion ρ so, daß $\rho(h) \subseteq \{h' | h \succ h'\}$ für jedes $h \in L_h$.*
Die Mengen der 1-Schritt- und n-Schritt–Spezialisierungen sind

$$\begin{aligned} \rho^1(h) &= \rho(h) \\ \rho^n(h) &= \{h' | h' \in \rho(h'') \textit{ für ein } h'' \in \rho^{n-1}(h), n \geq 2\} \end{aligned}$$

Die Menge aller Spezialisierungen von h ist $\rho^*(h) = \forall_{n \geq 1} \bigcup \rho^n(h)$. Eine 1-Schritt–Spezialisierung heißt auch elementare Spezialisierung oder elementare Spezialisierungsoperation.
Eine Hypothese h', die durch eine elementare Spezialisierungsoperation aus der Hypothese h hervorgegangen ist, heißt direkte Spezialisierung von h.

Elementare Spezialisierungsoperationen im ILP-Hypothesenraum

Ein auf der θ–Subsumtion beruhender Spezialisierungsoperator ρ_θ für Klauseln oder Teilgruppenbeschreibungen erzeugt aus einer Hypothese h speziellere Hypothesen, die von h θ–subsumiert sind, das heißt, $\rho_\theta(h) \subseteq \{h' | h \succ_\theta h'\}$. Dabei gibt es zwei elementare Spezialisierungsoperationen, nämlich Anfügen eines Literals und Ausführen einer Substitution.

Beispiel 32
Anfügen des Literals $q(Y)$ *an* $h = \{p(X,Y)\}$ *ergibt* $h' = \{p(X,Y), q(Y)\}$.
Ausführen der Substitution $[Y/X]$ *auf* $h = \{p(X,Y)\}$ *ergibt* $h' = \{p(X,X)\}$.

Einige Spezialisierungsoperatoren für ILP-Suchräume verwenden als einzige elementare Spezialisierungsoperation das Anfügen eines Literals, indem sie eine Substitution $[X/Y]$ oder $[X/a]$ explizit repräsentieren durch das Literal $X = Y$ beziehungsweise $X = a$ und dieses an die Hypothese, die spezialisiert werden soll, anfügen, ohne die betreffende Variable X tatsächlich zu substituieren. ILP-Spezialisierungsoperatoren können auch so definiert werden, daß sie nicht eine Hypothese $h = l_1, \ldots, l_n$ erzeugen und diese später durch Ausführen einer Substitution θ zur Hypothese $h' = h\theta = (l_1, \ldots, l_n)\theta$ spezialisieren, sondern die Hypothese h' durch Aneinanderfügen der Literale $l_i\theta$ direkt aufbauen (zum Beispiel WARMR [DT01]). Auch diese ILP-Spezialisierungsoperatoren kennen nur die elementare Spezialisierungsoperation Anfügen eines Literals.

Deklarativer Sprachbias

Welche Literale an die Hypothesen angefügt und welche Substitutionen ausgeführt werden können, hängt von der Hypothesensprache ab, die erzeugt werden soll. Viele ILP-Systeme bieten einen deklarativen Sprachbias, mit dem die Hypothesensprache für jede einzelne Anwendung vom Anwender individuell festgelegt werden kann. In der ILP wurde eine Vielzahl von Ansätzen zur Deklaration von Hypothesensprachen entwickelt. Einen Überblick geben [NRA+96, Tau94]. Häufig wird zum Beispiel ein Modus-basierter Sprachbias verwendet [Mug95, DT01]. Beim Modus-basierten Sprachbias deklariert der Anwender das Ein-/Ausgabe-Verhalten von Prädikaten, etwa durch Modus-Literale. Zum Beispiel gibt das Modus-Literal $mult(+,+,-)$ vor, daß ein Literal mit dem Prädikat $mult/3$ nur an eine Hypothese h angefügt werden darf, wenn an seinen ersten beiden, mit $+$ markierten Argumentstellen eine Variable steht, die schon in h vorkommt, während an der dritten Argumentstelle eine neue Variable stehen muß. Eine genauere Spezifikation der gewünschten Hypothesensprache ermöglichen Argumenttypen in den Modusdeklarationen, zum Beispiel der Argumtenttyp int im Modusliteral $mult(+int, +int, -int)$. Eine Variable darf in den Hypothesen nur an Stellen mit dem selben Argumenttyp vorkommen. Ein Spezialisierungsoperator für so deklarierte Hypothesensprachen spezialisiert eine Hypothese h, indem er Literale generiert und an h anhängt, die den Modus-Deklarationen genügen. Weitere Formen von deklarativem Sprachbias, die in der ILP für die Teilgruppenanalyse gebräuchlich sind, werden in Abschnitt 4.4 diskutiert.

Definition von ILP-Spezialisierungsoperatoren

Ein ILP-Spezialisierungsoperator soll die Elementaroperationen so auswählen, daß die beabsichtigte Hypothesensprache vollständig, aber ohne Mehrfachvorkommen von Hypothesen erzeugt wird. Die Definition eines solchen ILP-Spezialisierungsoperators ist selbst für vergleichsweise einfache, endliche ILP-Hypothesensprachen nicht trivial (siehe zum Beispiel [NCdW97, BS99]).

Individuenverbundenheit. Eine Klausel heißt verbunden, wenn jedes ihrer Literale über eine Kette von Variablengleichheiten mit dem Klauselkopf in Verbindung steht. Eine Teilgruppenbeschreibungen wird hier als individuenverbunden bezeichnet, wenn jedes ihrer Literale über eine Kette von Variablengleichheiten mit dem Populationsliteral und damit mit den Individuenvariablen verbunden ist.

Definition 16 (Individuenbezogene Verbundenheit) *Eine Teilgruppenbeschreibung* $h = \{l_1, l_2, \ldots, l_m\}$ *mit designierten Individuenvariablen* $V_1, \ldots, V_n$ *heißt individuenverbunden, falls alle Literale in* h *individuenverbunden sind. Ein Literal* $l_i \in h$ *ist individuenver-*

bunden, wenn l_i eine Individuenvariable enthält oder wenn l_i mindestens eine Variable V enthält, die in einem individuenverbundenen Literal $l_j \in h$ vorkommt.

Beispiel 33 *Die Individuenvariable sei V. Dann ist die Teilgruppenbeschreibung $h_1 = p(V, X, Y), w(X, Z)$ individuenverbunden.*
Die Teilgruppenbeschreibung $h_2 = p(V, X, Y), w(W, Z)$ ist nicht individuenverbunden.

Nicht-individuenverbundene Teilgruppenbeschreibungen sind bei der Suche nach interessanten Teilgruppen nicht nützlich. Sie sind entweder redundant, weil ihre Abdeckung identisch ist mit der Abdeckung ihrer größten verbundenen Teilmenge, oder sie beschreiben eine leere Teilgruppe.

Beweis 2 *Es sei $h = h_1 \cup h_2$ eine Teilgruppenbeschreibung und h_1 sei der verbundene Teil von h und h_2 sei der nicht-verbundene Teil von h. Die Substitution σ ist eine Grundsubstitution für die Individuenvariablen $V_1, \ldots, V_n$.*

$$\begin{aligned} cov_{V_1,\ldots,V_n}(h) &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h\sigma)\} \\ &= \{(V_1,\ldots,V_n)\sigma | D \models \exists((h_1 \wedge h_2)\sigma)\} \\ &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma \wedge h_2\sigma)\} \\ &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma \wedge h_2)\} \end{aligned}$$

Weil σ eine Substitution für die Variablen $V_1, \ldots, V_n$ ist, die in h_2 nicht vorkommen (sonst wäre h_2 verbunden), gilt $h_2\sigma = h_2$.

$$\begin{aligned} &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma) \wedge \exists(h_2)\} \\ &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma) \textit{ und } D \models \exists(h_2)\}, \end{aligned}$$

weil h_1 und h_2 wegen der Unverbundenheit keine gemeinsamen Variablen haben. Wenn $D \models \exists(h_2)$ eine wahre Aussage ist, ist

$$\begin{aligned} cov_{V_1,\ldots,V_n}(h) &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma) \textit{ und } wahr\} \\ &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma)\} \\ &= cov_{V_1,\ldots,V_n}(h_1) \end{aligned}$$

Wenn $D \models \exists(h_2)$ eine falsche Aussage ist, ist

$$\begin{aligned} cov_{V_1,\ldots,V_n}(h) &= \{(V_1,\ldots,V_n)\sigma | D \models \exists(h_1\sigma) \textit{ und } falsch\} \\ &= \emptyset \end{aligned}$$

$\diamondsuit$

Um ausschließlich individuenverbundene Teilgruppenbeschreibungen herzustellen, darf ein Spezialisierungsoperator nur Literale an eine Hypothese h anfügen, die eine Variable enthalten, die in h bereits vorkommt. Eine solche Variable wird oft als „alte Variable“ bezeichnet; Variablen im neu angefügten Literal, die nicht in h vorkommen, heißen „neue“ Variablen.

Reduzierte Klauseln. Um Redundanz im Hypothesenraum zu vermeiden, versucht man die Spezialisierungsoperatoren so zu definieren, daß sie genau einen Vertreter aus jeder Klasse subsumtions-äquivalenter Hypothesen erzeugen. Dabei ist es üblich, die syntaktisch kleinsten Vertreter zu generieren. In der ILP heißen diese kleinsten Vertreter einer Äquivalenzklasse reduziert. Sie sind bis auf Variablenumbenennungen und Literalreihenfolge eindeutig. Nicht-reduzierte Hypothesen enthalten immer mehrere Literale mit demselbem Prädikatsymbol und gleicher Stelligkeit.

Definition 17 ([NCdW97]) *Eine Klausel h heißt reduziert, wenn sie nicht zu einer echten Teilmenge von sich selbst subsumtions-äquivalent ist.*

Diese Definition läßt sich auf Teilgruppenbeschreibungen übertragen.

Beispiel 34 *Die Teilgruppenbeschreibung $h_1 = \{p(X), q(X,a), q(X,Y)$ ist nicht reduziert, weil sie sich durch die Substitution $[Y/a]$ in ihre Teilmenge $h_0 = \{p(X), q(X,a)$ überführen läßt. Die Teilgruppenbeschreibung $h_2 = \{p(X), q(X,a), q(X,Y), q(Y,Z)$ ist reduziert.*

Eine Schwierigkeit bei der Definition von ILP-Spezialisierungsoperatoren liegt darin, daß sich nicht alle reduzierten Hypothesen durch eine einzige elementare Spezialisierungsoperation aus einer ebenfalls reduzierten Hypothese erzeugen lassen. Vielmehr kann es nötig werden, mehrere elementare Spezialisierungsoperationen in einem Spezialisierungsschritt gemeinsam vorzunehmen, weil jede elementare Spezialisierungsoperation einzeln ausgeführt eine nicht-reduzierte Hypothese ergeben würde [BS99].

Beispiel 35

$$\begin{aligned} h_0 &= \{p(X), q(X,a)\} \\ h_1 &= \{p(X), q(X,a), q(X,Y)\} \\ h_2 &= \{p(X), q(X,a), q(X,Y), q(Y,Z)\} \end{aligned}$$

Gegeben sei die Hypothese h_0. Anfügen des Literals $q(X,Y)$ an h_0 ergibt die nicht-reduzierte Hypothese h_1. Anfügen des Literals $q(Y,Z)$ an h_1 ergibt wieder eine reduzierte Hypothese, nämlich h_2:

$$\begin{aligned} h'_0 &= \{p(X), q(X,Y,Z), q(X,Z,W)\} \\ h'_1 &= \{p(X), q(X,Z,Z), q(X,Z,W)\} \\ h'_2 &= \{p(X), q(X,Z,Z), q(X,Z,X)\} \end{aligned}$$

Gegeben sei die Hypothese h'_0. Ausführen der Substitution $[Y/Z]$ ergibt die nicht-reduzierte Hypothese h'_1. Ausführen der Substitution $[W/X]$ erzeugt aus h'_1 die reduzierte Hypothese h'_2.

2.4.4 Ein grundlegender Suchalgorithmus.

Wenn die Suche als Anwendung eines Spezialisierungsoperators implementiert ist, kann der Suchraum beschränkt werden, indem Hypothesen, die ein Beschränkungskriterium erfüllen, von der weiteren Spezialisierung ausgeschlossen werden. Dadurch werden alle ihre Spezialisierungen aus dem Suchraum entfernt. Ein Beschränkungskriterium ist sicher für ein Akzeptanzkriterium, wenn es nur für solche Hypothesen zutrifft, deren Spezialisierungen das Akzeptanzkriterium sicher nicht erfüllen können.

Definition 18 *Ein Beschränkungskriterium prune(h, D) ist sicher für ein Akzeptanzkriterium acc im Suchraum $(L, \succ)$, wenn gilt: wenn prune$(h, D) =$ wahr, dann gilt für alle $h' \in L$ mit $h \succ h'$ $acc(h', D) =$ falsch.*

Einen einfachen Suchalgorithmus, der auf einem Spezialisierungsoperator beruht, zeigt Tafel 2.16. In den Algorithmus eingegeben werden eine Datenbank D, die Deklaration der Individuenpopulation in Form des Literals pop, sowie ein Spezialisierungsoperator ρ, der implizit die Hypothesensprache festlegt. Der Algorithmus alterniert zwischen einer Generierungs- und einer Auswertungsphase. In der Generierungsphase cand_gen werden neue Hypothesen erzeugt, in der Evaluierungsphase cand_test werden sie per Datenbankanfrage ausgewertet. Die Menge S enthält die Hypothesen, die in der jeweils nächsten

Tafel 2.16 Basis-Algorithmus für die Suche nach interessanten Teilgruppen

Eingabe: Datenbank D, pop, ρ
Ausgabe: Menge akzeptierter Regeln H

```
1.  search(D, pop, ρ): H
2.  H = ∅;
3.  S = {{pop}};
4.  while S ≠ ∅ do
5.          S' = cand_gen(S);
6.          S  = cand_test(S', H, D);
7.  return H;

8.  cand_gen(S, ρ): S'
9.  S' = ∅;
10. for all h ∈ S : S' = S' ∪ ρ^1(h);
11. return S';

12. cand_test(S', H, D): S
13. S := ∅;
14. for all h ∈ S'
15.         erg := eval(h, D);
16.         if erg = accept then
17.                 accept(h, H);
18.                 if prune_accepted then erg := prune else erg := keep;
19.         if erg = keep then S := S ∪ {h};
20. return S;
```

Generierungsphase durch den Spezialisierungsoperator weiter spezialisiert werden sollen. Der Suchalgorithmus in Tafel 2.16 durchsucht den Hypothesenraum vom Allgemeinen zum Speziellen. Zu Beginn wird S daher initialisiert mit der allgemeinstmöglichen Teilgruppenbeschreibung pop, die die gesamte Population abdeckt. Ergebnis der Generierungsphase ist die Menge S' der Hypothesen, die in der nachfolgenden Evaluierungsphase ausgewertet werden sollen. Die Prozedur $eval(h, D)$ wertet eine Hypothese auf der Datenbank aus und vergleicht das Resultat der Auswertung mit dem Akzeptanzkriterium und mit dem Beschränkungskriterium. Die Auswertung einer Hypothese kann verschiedene Ergebnisse liefern:

- accept: die Hypothese erfüllt das Akzeptanzkriterium und wird in die Lösungsmenge h aufgenommen.
- prune: die Hypothese erfüllt das Beschränkungskriterium und darf nicht weiter spezialisiert werden,
- keep: die Hypothese kann zwar nicht akzeptiert werden, muß aber weiter spezialisiert werden.

Eine Hypothese, die das Beschränkungskriterium erfüllt, heißt kappbar. Eine Hypothese, die möglicherweise akzeptable Spezialisierungen hat und deshalb nicht kappbar ist, heißt spezialisierbar. Hypothesen, die weiter spezialisiert werden sollen, werden in der Menge S gesammelt. Diese bildet die Eingabe für die nächste Generierungsphase. Kappbare Hypothesen werden nicht in die Menge S aufgenommen und daher im weiteren Verlauf der Suche nicht weiter spezialisiert. Der Suchalgorithmus beschränkt so den Suchraum. Die Suche endet, wenn die Evaluierungsphase eine leere Menge S liefert.

Varianten von Akzeptanzkriterien

accept(h, H): Es gibt zwei Varianten der Teilgruppenanalyse. Die erste Variante verlangt als Resultat der Teilgruppenanalyse eine Lösungsmenge, die alle zur Hypothesensprache gehörigen Teilgruppen enthält, deren Interessantheit einen bestimmten Schwellwert d_{min} erreicht. Diese Variante wird im folgenden als Teilgruppenanalyse mit Schwellwert-Akzeptanz bezeichnet. Bei der Schwellwert-Akzeptanz sammelt die Funktion accept(h, H) im Schritt 17 des Algorithmus alle Hypothesen mit ausreichend großem Interessantheitswert in der Lösungsmenge H.

Die zweite Variante verlangt als Resultat eine Lösungsmenge, die die k interessantesten Teilgruppen in der Hypothesensprache enthält. Diese Variante heißt im folgenden Teilgruppenanalyse mit k-besten-Akzeptanz. Für die Teilgruppenanalyse mit k-besten-Akzeptanz ist eine Hypothese h akzeptabel, wenn unter den bisher evaluierten Hypothesen höchstens $k-1$ Hypothesen eine größere Interessantheit aufweisen als h, also wenn h zu den k interessantesten unter den bisher evaluierten Hypothesen gehört. Bei der k-besten-Akzeptanz füllt die Funktion accept(h, H) die Lösungsmenge H mit Hypothesen auf, bis sie k Elemente enthält. Wenn im Verlauf der Suche eine Hypothese h mit höherem Interessantheitswert als die am wenigsten interessante Hypothese h' in H gefunden wird, so ersetzt accept(h, H) die schwächste Hypothese h' durch die interessantere h.

Bei der k-besten-Akzeptanz wirkt die Interessantheit der am wenigsten interessanten Hypothese h' in H als Schwellwert, den Hypothesen übertreffen müssen, um akzeptabel zu sein. Diese Akzeptanzschwelle wird während der Suche nach akzeptablen Hypothesen immer höher, wenn nach und nach interessantere Hypothesen gefunden werden, während bei der Schwellwert-Akzeptanz die Akzeptanzschwelle konstant bleibt.

prune_accepted: Eine weitere Variante des Findens interessanter Teilgruppen definiert nur allgemeinste Teilgruppen als interessant; Spezialisierungen akzeptierter Teilgruppen werden nicht akzeptiert. Diese Variante wird hier als Ausschluß akzeptierter Hypothesen oder Akzeptierten-Ausschluß bezeichnet. Der Algorithmus 2.16 realisiert eine Suche mit Ausschluß akzeptierter Hypothesen, wenn die Variable prune_accepted mit dem Wert wahr belegt ist. Bei der Teilgruppenanalyse mit Auschluß akzeptierter Teilgruppen können alle Spezialisierungen einer akzeptierten Hypothese von der weiteren Suche ausgeschlossen werden. Der Algorithmus löscht dazu im Schritt 18 die akzeptierten Hypothesen aus der Menge S der weiter zu spezialisierenden Hypothesen.

Ebenenweise Suche

Der in Tafel 2.16 beschriebene Suchalgorithmus realisiert eine ebenenweise Suche. Ihr Kennzeichen ist die Einteilung des Suchraums in Ebenen. Im Algorithmus 2.16 bilden die Hypothesen eine Suchebene, die in derselben Generierungsphase erzeugt und in der anschließenden Auswertungsphase gemeinsam ausgewertet werden. Ein Suchalgorithmus ohne ebenenweise Suche, der dem Algorithmus 2.16 entspricht, würde auf die Trennung zwischen Generierungs- und Auswertungsphase verzichten und jede Hypothese unmittelbar nach ihrer Generierung durch eine Datenbankanfrage auswerten.

Das Prinzip der ebenenweisen Suche haben Mannila und Toivonen in [MT97b] eingeführt und Eigenschaften der ebenenweisen Suche theoretisch untersucht. Der Vorteil der ebenenweisen Suche liegt darin, daß die gemeinsame ebenenweise Auswertung von Hypothesen vor allem bei großen Datenbanken und bei multi-relationalen Datenbanken Zeit sparen kann, denn wenn geeignete Auswertungsmechanismen und Datenformate verfügbar sind, können alle Hypothesen einer Ebene in einem gemeinsamen Durchlauf über die Datenbank ausgewertet werden. Bei großen Datenbanken, die nicht komplett in den Hauptspeicher des Rechners passen, vermeidet die gemeinsame Auswertung der Hypothesen wiederholtes Laden und Auslagern von Teilen der Datenbank [DDR97]. Die Anzahl der nötigen Durchläufe über die Datenbank entspricht dann der Anzahl der Suchebenen.

2.4.5 Verwandte Ansätze

Die Teilgruppenanalyse im individuenzentrierten nicht-monotonen Rahmen der ILP ist Gegenstand mehrerer neuerer Forschungsarbeiten. Sie ist von Bedeutung, weil sie die direkte Anwendung der Teilgruppenanalyse auf (multi-)relationalen Datenbanken erlaubt, die in der Praxis häufig vorkommen und nicht immer in ein aussagenlogisches, einrelationales Format zu überführen sind. Sie ist flexibel, weil sie erlaubt, verschiedene Individuenpopulationen in einer Datenbank zu analysieren, ohne die Datenbank konvertieren zu müssen. Zu den neueren Systemen, die für diese Aufgabe entwickelt wurden, gehören Wrobels Midos, Flachs und Lachiches Tertius und Dehaspes WARMR [Wro97, FL01, Deh98]. Das System Midos hat die Teilgruppenanalyse von Attribut–Wert–Repräsentationen zu relationalen Datenbanken verallgemeinert. Es beinhaltet einen Spezialisierungsoperator und einen Sprachbias, die besonders auf relationale Datenbanken ausgerichtet sind. WARMR hat die Data Mining–Aufgabe frequent query discovery für relationale Datenbanken eingeführt. Das System überträgt eine Methode zur Auswertung von Hypothesen, die der bekannte Data Mining–Algorithmus Apriori für aussagenlogische Transaktionsdatenbanken eingeführt hat, auf relationale Datenbanken [DT01]. Beim System Tertius liegt ein Schwerpunkt auf der Entwicklung eines Interessantheitsmaßes, das sich besonders für die Bewertung allgemeiner Klauseln beim Entdecken interessanter Klauseln in logischen Theorien eignet. Die vorliegende Arbeit thematisiert Möglichkeiten zur sicheren Beschränkung des Suchraums bei der Teilgruppenanalyse im individuenzentrierten nicht-monotonen Rahmen der ILP.

Kapitel 3

Interessantheitsfunktionen und Optimumschätzfunktionen

Zur Bewertung der Interessantheit von Hypothesen in individuenzentrierten Lernproblemen existiert eine Vielzahl von Interessantheitsfunktionen. Einen Vergleich und eine Diskussion gebräuchlicher Interessantheitsfunktionen bietet zum Beispiel [Klö96]. Abhängig von der jeweiligen Zielsetzung können geeignete Interessantheitsfunktionen für eine Anwendung ausgewählt werden. Besonders bei Anwendungen mit sehr großem Hypothesenraum ist für die Wahl einer Interessantheitsfunktion auch von Bedeutung, daß die Interessantheitsfunktion eine sichere Beschränkung des Suchraums erlaubt. In diesem Kapitel werden einige bekannte Interessantheitsfunktionen zur Bewertung von Teilgruppen diskutiert und Möglichkeiten zur Herleitung von Optimumschätzfunktionen gezeigt.

3.1 Interessantheit von Teilgruppen

Ziel von KDD- und Data Mining-Aktivitäten ist es, Wissen zu entdecken, das sich praktisch anwenden und umsetzen läßt. Um solches Wissen aufzufinden und zu erkennen, sind subjektive Interessantheitskriterien erforderlich, die sich an der Zielsetzung der Anwendung und den Eigenheiten des Anwendungsgebiets orientieren. Solche subjektiven Interessantheitskriterien sind schwer zu formalisieren und lassen sich naturgemäß kaum anwendungsübergreifend definieren. KDD-Systeme bieten deshalb objektive Interessantheitsfunktionen an, die sich möglicherweise durch Parametereinstellungen an die Zielsetzung einer Anwendung anpassen lassen. Die objektiven Interessantheitskriterien können eine Vorauswahl von potentiell interessanten Beobachtungen leisten; unter diesen selektieren die Anwendungsexperten solche, die auch nach subjektiven Maßstäben interessant, verständlich und umsetzbar sind. Die objektiven Kriterien unterstützen die subjektive Auswahl durch Aussagen über die Validität der Entdeckungen [Mül99, ST96].

3.1.1 Berechnung der Interessantheit

Die Interessantheit von Hypothesen kann aus verschiedenen Werten und Größen abgeleitet werden. So betrachten einige Interessantheitsfunktionen die syntaktische Form der Hypothesen, beispielsweise ihre Komplexität oder Länge. Die hier betrachteten objektiven Interessantheitsfunktionen verwenden keine Eigenschaften der Teilgruppenbeschreibung h, sondern lediglich Eigenschaften der von h abgedeckten Teilgruppe $cov(h) = H$. Die Interessantheit einer Teilgruppe H wird aus einem oder mehreren der folgenden Werte berechnet.

Tafel 3.1 Vierfelder-Tafel

	H	$\overline{H}$	
T	$\lvert H \cap T\rvert$	$\lvert\overline{H} \cap T\rvert$	$\lvert T\rvert$
$\overline{T}$	$\lvert H \cap \overline{T}\rvert$	$\lvert\overline{H} \cap \overline{T}\rvert$	$\lvert\overline{T}\rvert$
	$\lvert H\rvert$	$\lvert\overline{H}\rvert$	$\lvert Pop\rvert$

- die Mächtigkeit oder Häufigkeit der Teilgruppe H

$$|H| = |\{(V_1, \ldots, V_n)\sigma \mid D \models \exists((pop \wedge h)\sigma) \text{ und } (V_1, \ldots, V_n)\sigma \text{ ist grund}\}|$$

- der Mächtigkeit der Gesamtpopulation

$$|Pop| = |\{(V_1 \ldots, V_n)\sigma \mid D \models \exists(pop\sigma) \text{ und } (V_1 \ldots, V_n)\sigma \text{ ist grund}\}|$$

- der Mächtigkeit der Zielgruppe

$$|T| = |\{(V_1, \ldots, V_n)\sigma \mid D \models \exists((pop \wedge t)\sigma) \text{ und } (V_1, \ldots, V_n)\sigma \text{ ist grund}\}|$$

- der Anzahl der Individuen aus der Zielgruppe, die die Teilgruppe abdeckt

$$|H \cap T| = |\{(V_1, \ldots, V_n)\sigma \mid D \models \exists((pop \wedge t \wedge h)\sigma) \text{ und } (V_1, \ldots, V_n)\sigma \text{ ist grund}\}|$$

und ihren Komplementen,

- der Mächtigkeit der Gesamtpopulation ohne die Teilgruppe, $|\overline{H}| = |Pop \setminus H|$,
- der Mächtigkeit der Gesamtpopulation ohne die Zielgruppe, $|\overline{T}| = |Pop \setminus T|$,
- der Anzahl $|\overline{H} \cap T|$ der Individuen aus der Zielgruppe, die die Teilgruppe nicht abdeckt,
- der Anzahl $|H \cap \overline{T}|$ der nicht zur Zielgruppe gehörenden Individuen, die die Teilgruppe abdeckt,
- der Anzahl $|\overline{H} \cap \overline{T}|$ der nicht zur Zielgruppe gehörenden Individuen, die die Teilgruppe nicht abdeckt.

Diese Größen lassen sich zu einer Vierfeldertafel wie in Tafel 3.1 anordnen.

3.1.2 Interpretation der Hypothesen als Regeln

Wenn eine Hypothese h und die Definition der Zielgruppe t außer den Individuenvariablen keine weiteren Variablen gemeinsam haben, kann man den Zusammenhang zwischen einer Teilgruppe $H = cov(h)$ und der Zielgruppe T auch als Regel $i \in H \rightarrow i \in T$ darstellen, die besagt:

> „wenn ein Individuum $(V_1, \ldots, V_n)\theta$ zur Teilgruppe H gehört, gehört es auch zur Zielgruppe T."

Der Teil der Regel links vom Folgerungspfeil heißt Prämisse, Voraussetzung, Bedingungsteil oder Regelrumpf; der rechte Teil heißt Folgerung, Dann-Teil oder Regelkopf. Die Individuen, für die Prämisse und Folgerung zutreffen, unterstützen die Regel, während die Individuen, für die nur die Prämisse, aber nicht die Folgerung zutreffen, der Regel widersprechen. Entsprechend heißt die Menge $H \cap \overline{T}$ die Widersprechermenge der Regel $H \rightarrow T$, und $H \cap T$ heißt die Unterstützermenge der Regel $H \rightarrow T$.

Beim Finden interessanter Teilgruppen ändert sich der Kopf der betrachteten Regeln nicht und ist also für die gesamte Anwendung fixiert. Der Hypothesenraum besteht deshalb aus Teilgruppenbeschreibungen h und das Interessantheitskriterium wird für die von h beschriebenen Teilgruppen berechnet. Die Zielgruppe, also der Regelkopf, kann als Bestandteil des Interessantheitskriteriums betrachtet werden. Im Unterschied dazu bewertet eine Aufgabenstellung mit variablen Folgerungen, wie beispielsweise das Finden von Assoziationsregeln, nicht Teilgruppen, sondern Zusammenhänge zwischen Teilgruppenpaaren.

3.1.3 Datenbank-Anfragen zur Ermittlung der benötigten Werte

Zur Berechnung der Interessantheit einer Teilgruppenbeschreibung werden die Mächtigkeiten der Mengen H, T, Pop und $H \cap T$ durch Datenbank-Anfragen ermittelt. Für eine deduktive Datenbank kann dazu ein Prolog-Interpreter verwendet werden. Das dreistellige Prolog-Prädikat $findall$ berechnet alle Grundinstanzen der Variablen in dem Ausdruck an erster Argumentstelle, für die der Ausdruck an zweiter Argumentstelle im Logikprogramm gültig ist. An der dritten Argumentstelle liefert es die Menge dieser Instanziierungen zurück. Alle Variablen, die nicht in dem Ausdruck an erster Argumentstelle vorkommen, sind existenzquantifiziert. Wenn das erste Argument die Individuenvariablen angibt, liefert der Aufruf von $findall$ mit pop als zweitem Argument genau die Population Pop. Mit (pop, t) als zweitem Argument berechnet $findall$ die Zielgruppe. Die folgenden Prolog-Anfragen bestimmen also die Population Pop und die Zielgruppe T.

$findall((V_1, \ldots, V_n), (pop), Pop).$
$findall((V_1, \ldots, V_n), (pop, t), T).$

Die Abdeckung H einer Hypothese h und die von der Hypothese h abgedeckten Zielgruppenelemente lassen sich mit folgenden Anfragen ermitteln:

$findall((V_1, \ldots, V_n), (h), H).$
$findall((V_1, \ldots, V_n), (h, t), H \cap T).$

Die Elemente in den berechneten Abdeckungen kann man zählen mit dem Prolog-Prädikat $length(Liste, Länge)$.

Die Anfragen lassen sich auch in SQL formulieren. Die SQL-Anfragen können so gestellt werden, daß sie nicht die Mengen selbst, sondern ihren Umfang zurückgeben. Sie haben die Form

SELECT COUNT DISTINCT $V_1, \ldots, V_n$ FROM $\langle preds \rangle$ WHERE $\langle conds \rangle$;

Dabei sind $V_1, \ldots, V_n$ die Individuenvariablen. Der Ausdruck $\langle preds \rangle$ muß die Datenbank-Prädikate angeben, die in den Literalen (pop), (pop, t), (pop, h) beziehungsweise (pop, h, t) vorkommen. Die Variablen-Bindungen und -Gleichheiten in diesen Literalkonjunktionen müssen in entsprechende SQL-Bedingungen $\langle conds \rangle$ transformiert werden.

Beispiel 36 (Fortsetzung von Beispiel 27) *Folgende Prolog-Anfragen berechnen die Population und die Zielgruppe sowie die Teilgruppe der langen Silben und ihre Überschneidung mit der Zielgruppe.*

$findall(S, (silbe(S, Accent, Dauer)), Pop).$
$findall(S, (silbe(S, Accent, Dauer), Accent = yes), T).$
$findall(S, (silbe(S, Accent, Dauer), Dauer = lang), H)$

$findall(S, (silbe(S, Accent, Dauer), Accent = yes, Dauer = lang), H \cap T)$.

Die SQL-Anfragen zur Berechnung der entsprechenden Mächtigkeiten sind

SELECT COUNT DISTINCT s FROM silbe;
SELECT COUNT DISTINCT s FROM silbe WHERE accent IS 'yes';
SELECT COUNT DISTINCT s FROM silbe WHERE dauer IS 'lang';
SELECT COUNT DISTINCT s FROM silbe WHERE accent IS 'yes' AND dauer IS 'lang';

3.1.4 Optimumschätzfunktionen

Eine sichere Beschränkung des Suchraums ist möglich, wenn die verwendete Interessantheitsfunktion zur Ordnung des Suchraums monoton ist.

Definition 19 *Eine Interessantheitsfunktion d ist monoton zur Ordnung $\succ$ auf einem Suchraum $(L_h, \succ)$, wenn für alle $h, h' \in L_h$ und alle Datenbanken D gilt: wenn $h \succ h'$, dann $d(h) \geq d(h')$.*

Viele gebräuchliche Interessantheitsfunktionen sind nicht monoton zur extensionalen Subsumtion, erlauben aber die Herleitung von Optimumschätzfunktionen. Eine zu einer Interessantheitsfunktion d gehörende Optimumschätzfunktion d_{oe} berechnet für eine Hypothese h die maximale Interessantheit $d_{oe}(h)$, die Spezialisierungen von h bestenfalls erreichen können.

Definition 20 *Eine Optimumschätzfunktion d_{oe} zu einem Interessantheitskriterium d ist eine Funktion, die, gegeben eine Hypothese h, eine obere Schranke für die Interessantheit aller Spezialisierungen h' von h berechnet, also so, daß gilt*

$\forall h'$ *mit* $h \succ h'$ $d(h') \leq d_{oe}(h)$.

Optimumschätzfunktionen erlauben eine sichere Beschränkung des Suchraums, wenn der Optimumschätzwert einer Hypothese h unter der geforderten Akzeptanzschwelle liegt. Dann können die Spezialisierungen von h von der weiteren Suche ausgeschlossen werden, da sie die für Akzeptanz minimal erforderliche Interessantheit sicher nicht erreichen können. Für eine monotone Interesantheitsfunktion bildet die Negation des Akzeptanzkriteriums ein sicheres Beschränkungskriterium: wenn eine Hypothese das Akzeptanzkriterium nicht erfüllt, erfüllt sie damit das Beschränkungskriterium.

3.2 Häufigkeit, Bestätigung und Genauigkeit

Ein sehr grundlegendes Interessantheitskriterium für Teilgruppen ist ihre Mächtigkeit. Danach gilt eine Teilgruppe als um so interessanter, je mehr Individuen sie umfaßt. Als Häufigkeit einer Teilgruppe bezeichnet man die Anzahl ihrer Elemente relativ zum Umfang der Grundpopulation.

Definition 21 *Die Häufigkeit freq(H) einer Teilgruppe H*

$$freq(H) = \frac{|H|}{|Pop|}.$$

Die Häufigkeit freq(h) einer Teilgruppenbeschreibung h ist gleich der Häufigkeit freq(H) der von ihr abgedeckten Teilgruppe $H = cov(h)$.

Die Häufigkeit freq ist trivialerweise monoton zur extensionalen Subsumtion.

Die Interessantheitsfunktionen Bestätigung und Genauigkeit sind abgeleitet aus den Kriterien *support* beziehungsweise *confidence*, die bei der Suche nach Assoziationsregeln Verwendung finden. Sie nehmen Werte aus dem Intervall $[0, 1]$ an.

Definition 22 *Die Bestätigung $supp(H)$ einer Teilgruppe H ist die Anzahl der in ihr enthaltenen Zielindividuen relativ zum Umfang der gesamten Zielgruppe,*

$$supp(H) = \frac{|H \cap T|}{|T|}.$$

Die Genauigkeit conf(H) einer Teilgruppe H ist gleich dem Anteil der in ihr enthaltenen Zielindividuen an ihren Elementen, das heißt,

$$conf(H) = \frac{|H \cap T|}{|H|}.$$

Die Bestätigung $supp(h)$ einer Teilgruppenbeschreibung h ist gleich der Bestätigung der von ihr abgedeckten Teilgruppe H, also $supp(h) = supp(H)$ für $H = cov(h)$. Die Genauigkeit conf(h) einer Teilgruppenbeschreibung h ist gleich der Genauigkeit der von ihr abgedeckten Teilgruppe H, also conf$(h) =$ conf(H) für $H = cov(h)$.

Die Bestätigung ist ebenfalls monoton zur extensionalen Subsumtion. Die Genauigkeit ist nicht monoton zur extensionalen Subsumtion. Es kann auch keine nicht-triviale Optimumschätzfunktion für die Genauigkeit geben, da immer die Chance besteht, daß eine Spezialisierungsoperation genau die widersprechenden Instanzen aus der Abdeckung der Teilgruppe entfernt und dadurch eine Teilgruppe mit maximaler Genauigkeit 1 erzeugt.

Häufigkeit, Bestätigung und Genauigkeit sind sehr grundlegende Interessantheitskriterien und entsprechend leicht zu interpretieren. Sie haben jedoch den Nachteil, daß jedes dieser Kriterien für sich betrachtet nicht genügt, um die Qualität einer Hypothese zu beurteilen. Zum Beispiel erreichen triviale Teilgruppenbeschreibungen, die die gesamte Individuenpopulation abdecken, maximale Häufigkeit und Bestätigung, während speziellste Hypothesen, die genau ein Zielindividuum abdecken, von maximaler Genauigkeit sind. Ein sinnvolles Interessantheitskriterium sollte also immer die zwei Aspekte Allgemeinheit (in Form von Häufigkeit oder Bestätigung) und Genauigkeit einer Teilgruppe berücksichtigen.

3.3 Verteilungsungewöhnlichkeit

Die Verteilungsungewöhnlichkeit (*distributional unusualness*) dient dazu, Teilgruppen mit abweichendem Verhalten innerhalb einer Grundmenge von Individuen zu erkennen. Wrobel hat sie in [Wro97] vom aussagenlogischen Fall für das Entdecken interessanter Teilgruppen in mehrrelationalen Datenbanken adaptiert. Die Verteilungsungewöhnlichkeit bewertet, wie stark der Anteil der Zielgruppenelemente in einer Teilgruppe vom Anteil der Zielgruppenelemente in der gesamten Population abweicht. Die Apriori-Verteilung der Zielgruppe T in der Gesamtpopulation Pop ist $\frac{|T|}{|Pop|}$. Die Verteilung der Zielgruppe in einer Teilgruppe H ist $\frac{|H \cap T|}{H}$. Die Abweichung der Verteilung der Zielgruppe in einer Teilgruppe von der Apriori-Verteilung ist die Differenz

$$\left(\frac{|H \cap T|}{|H|} - \frac{|T|}{|Pop|} \right).$$

Eine Teilgruppe mit ungewöhnlichem Zielgruppenanteil ist umso interessanter, je mehr Individuen sie umfaßt. Die Bedeutung der Allgemeinheit von Teilgruppen für ihre Interessantheit wird mit folgendem Faktor gewichtet:

$$\left(\frac{|H|}{|Pop|} \right)^{\alpha}.$$

Hier werden nur Definitionen mit $\alpha = 1$ betrachtet. Eine Schranke σ_m, die die minimale Abdeckung (relativ zur Gesamtpopulation) als interessant erachteter Teilgruppen angibt, dient dazu, allzu spezielle Hypothesen auszuschließen.

Die Verteilungsungewöhnlichkeit gibt es in mehreren Varianten. Die asymmetrische Version da beurteilt Teilgruppen mit ungewöhnlich hohem Zielgruppenanteil als interessant, die symmetrische Version ds beurteilt Teilgruppen mit ungewöhnlich hohem oder ungewöhnlich niedrigem Zielgruppenanteil als interessant.

Definition 23 *Die asymmetrische Verteilungsungewöhnlichkeit $da(H)$ und die symmetrische Verteilungsungewöhnlichkeit $ds(H)$ einer Teilgruppe H sind definiert wie folgt.*

$$da(H) = \begin{cases} -1 & \text{if } \frac{|H|}{|P|} < \sigma_m \\ \frac{|H|}{|P|} \cdot \left(\frac{|H \cap T|}{|H|} - \frac{|T|}{|P|} \right) & \text{else} \end{cases}$$

$$ds(H) = \begin{cases} 0 & \text{if } \frac{|H|}{|P|} < \sigma_m \\ \frac{|H|}{|P|} \cdot \left| \frac{|H \cap T|}{|H|} - \frac{|T|}{|P|} \right| & \text{else} \end{cases}$$

Die asymmetrische Verteilungsungewöhnlichkeit $da(h)$ einer Teilgruppenbeschreibung h ist gleich der asymmetrischen Verteilungsungewöhnlichkeit $da(H)$ der von h beschriebenen Teilgruppe $H = cov(h)$. Die symmetrische Verteilungsungewöhnlichkeit $ds(h)$ einer Teilgruppenbeschreibung h ist gleich der symmetrischen Verteilungsungewöhnlichkeit $ds(H)$ der von h beschriebenen Teilgruppe $H = cov(h)$.

3.3.1 Verhalten der Verteilungsungewöhnlichkeit

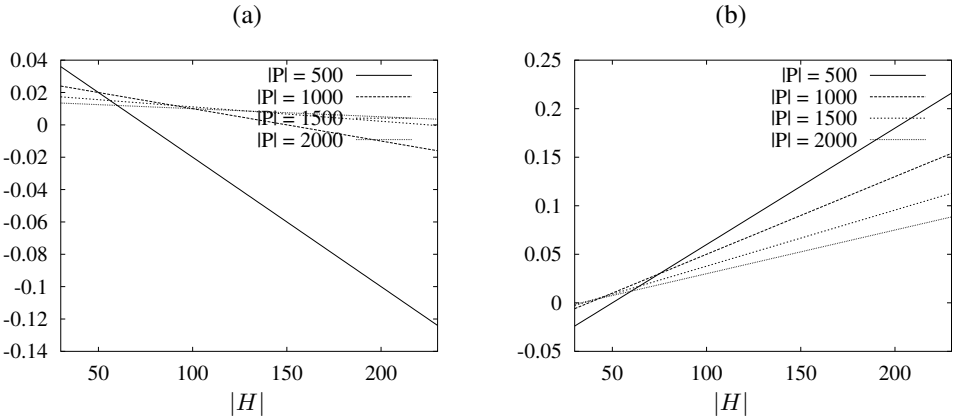
Die Abbildungen 3.1 (a) und (b) veranschaulichen das Verhalten der asymmetrischen Verteilungsungewöhnlichkeit von Teilgruppen H abhängig von ihrer Größe $|H|$ und der Größe ihrer Unterstützermenge $|H \cap T|$. Auf der y-Achse ist die asymmetrische Verteilungsungewöhnlichkeit $da(h)$ abgetragen. Die Population P besteht aus 500, 1000, 1500 und 2000 Individuen, die Zielgruppe T enthält immer 200 Individuen. In beiden Abbildungen (a) und (b) läuft die Größe der Teilgruppe $|H|$ von 30 bis 230, abgetragen auf der x-Achse. In Abbildung (a) decken alle Teilgruppen H 30 Zielindividuen ab ($|H \cap T| = 30$). Abbildung (a) zeigt, daß die asymmetrische Verteilungsungewöhnlichkeit fällt, wenn die Größe der Abdeckung $|H|$ wächst, aber die Anzahl abgedeckter Zielindividuen gleich bleibt.

In Abbildung (b) ist der Umfang der Unterstützermenge $|H \cap T| = |H| - 30$, das heißt, die Widersprechermenge $|H \cap \overline{T}|$ enthält immer 30 Individuen. Abbildung (b) zeigt, daß die asymmetrische Verteilungsungewöhnlichkeit steigt mit der Größe der Abdeckung $|H|$, wenn auch die Anzahl abgedeckter Zielindividuen zunimmt.

3.3.2 Optimumschätzfunktionen

Offensichtlich ermöglicht der erste Zweig der Definition von da eine häufigkeitsabhängige Beschränkung des Suchraums. Der zweite Zweig der Definition ist nicht monoton zur Häufigkeit der Hypothesen, erlaubt jedoch die Definition von Optimumschätzfunktionen [Wro97]. Die hier hergeleiteten Optimumschätzfunktionen da_{oe1} und da_{oe2} verwenden, daß die Anzahl $|cov(h)|$ der von einer Teilgruppenbeschreibung h abgedeckten Individuen sowie die Anzahl der von h abgedeckten Zielgruppenelemente $|cov(h) \cap T|$ eine obere Schranke bilden für die Anzahl der Individuen $|cov(h')|$ beziehungsweise die Anzahl der Zielgruppenelemente $|cov(h') \cap T|$, die jede Spezialisierung h' von h abdecken kann.

Abbildung 3.1 Verhalten der Interessantheitsfunktion $da(h)$ (a) abhängig von der Größe der Teilgruppe $|H| \in [30, 230]$ bei festem $|H \cap T| = 30$, und (b) abhängig von der Größe der Teilgruppe $|H| \in [30, 230]$ bei festem $|H \cap \overline{T}| = 30$. Die Population P besteht aus 500, 1000, 1500 und 2000 Individuen, die Zielgruppe T enthält 200 Individuen.



Erste Optimumschätzfunktion. Die Optimumschätzfunktion da_oe1 beruht darauf, daß eine bezüglich der Verteilungsungewöhnlichkeit optimale Hypothese h^* nur Zielgruppenelemente abdeckt, und zwar möglichst viele. Die Abdeckung H^* einer optimalen Spezialisierung h^* einer Teilgruppenbeschreibung h ist gleich ihrer Abdeckung an Zielgruppenindividuen $H^* \cap T$, also $\frac{|H^* \cap T|}{|H^*|} = 1$. Außerdem kann eine optimale Spezialisierung h^* einer Hypothese h höchstens so viele Zielgruppenelemente abdecken, wie h abdeckt, also $|H^*| = |cov(h) \cap T|$. Diese Überlegungen ergeben folgende Optimumschätzfunktion da_{oe1}.

Definition 24

$$da_{oe1}(h) = \frac{|cov(h) \cap T|}{|P|} \cdot \left(1 - \frac{|T|}{|P|}\right)$$

Proposition 2 *Für $\frac{|cov(h) \cap T|}{|P|} \geq \sigma_m$ berechnet die Optimumschätzfunktion da_{oe1} eine obere Schranke für die Verteilungsungewöhnlichkeit, die Spezialisierungen h' einer Hypothese h im besten Fall erreichen können. Das heißt:*

$da(h') \leq da_{oe1}(h)$ *für alle h' mit $h \succ h'$.*

Beweis 3 *Es sei $H = cov(h)$ und $H' = cov(h')$. Voraussetzung ist, daß $\frac{|H|}{|P|} \geq \sigma_m$. Der zweite Zweig der asymmetrischen Verteilungsungewöhnlichkeit da läßt sich umschreiben wie folgt.*

$$\begin{aligned}
da(h') &= \frac{|H'|}{|P|} \cdot \left(\frac{|H' \cap T|}{|H'|} - \frac{|T|}{|P|}\right) \\
&= \frac{1}{|P|} \cdot \left(|H'| \cdot \frac{|H' \cap T|}{|H'|} - \frac{|T|}{|P|} \cdot |H'|\right) \\
&= \frac{1}{|P|} \left(|H' \cap T| - \frac{|T|}{|P|} \left(|H' \cap T| + |H' \cap \overline{T}|\right)\right) \\
&= \frac{1}{|P|^2} \left(|P| \cdot |H' \cap T| - |T| \cdot |H' \cap T| - |T| \cdot |H' \cap \overline{T}|\right) \\
&= \frac{1}{|P|^2} \left((|P| - |T|) \cdot |H' \cap T| - |T| \cdot |H' \cap \overline{T}|\right) \\
&= \frac{1}{|P|^2} \left(|\overline{T}| \cdot |H' \cap T| - |T| \cdot |H' \cap \overline{T}|\right)
\end{aligned}$$

Durch Spezialisierung kann sich die Anzahl der Elemente, die eine Hypothese h abdeckt, nur verkleinern. Offensichtlich wird der Ausdruck in der Klammer dann maximal, wenn $|H \cap T|$ maximal und $|H' \cap \overline{T}|$ minimal ist. Für alle Spezialisierungen h' von h muß daher gelten:

$$\begin{aligned} |H' \cap T| &\leq |H \cap T| \\ |H' \cap \overline{T}| &\geq 0 \end{aligned}$$

Damit gilt:

$$\begin{aligned} da(h') &\leq \frac{1}{|P|^2}\left(|\overline{T}| \cdot |H \cap T| - |T| \cdot |H' \cap \overline{T}|\right) \\ &\leq \frac{1}{|P|^2} \cdot |\overline{T}| \cdot |H \cap T| \\ &= \frac{|H \cap T|}{|P|} \cdot \frac{|\overline{T}|}{|P|} \\ &= \frac{|H \cap T|}{|P|} \cdot \frac{|P| - |T|}{|P|} \\ &= \frac{|H \cap T|}{|P|} \cdot \left(1 - \frac{|T|}{|P|}\right) \end{aligned}$$

♢

Zweite Optimumschätzfunktion. Die asymmetrische Verteilungsungewöhnlichkeit erreicht ein Maximum für eine Hypothese, die möglichst viele Zielgruppenelemente abdeckt. Eine Spezialisierung h' einer Hypothese h kann maximal soviele Zielgruppenelemente abdecken wie h, also $|H' \cap T| \leq |H \cap T|$. Die Spezialisierung h' muß jedoch immer die Mindesthäufigkeit erreichen; sonst ist ihre Verteilungsungewöhnlichkeit minimal. Daher kann man eine Optimumschätzfunktion da_{oe2} auf die Mindesthäufigkeitsbedingung $\frac{|cov(h)|}{|P|} \geq \sigma_m$ aufbauen.

Definition 25

$$da_{oe2}(h) = \frac{|cov(h) \cap T|}{|P|} - \sigma_m \cdot \frac{|T|}{|P|}$$

Proposition 3 *Die Funktion da_{oe2} ist eine Optimumschätzfunktion da_{oe2} für die Verteilungsungewöhnlichkeit, das heißt:*

$da(h') \leq da_{oe2}(h)$ *für alle h' mit $h \succ h'$.*

Beweis 4 *Es sei $H = cov(h)$ und $H' = cov(h')$. Umformulieren ergibt folgende Gleichung für den zweiten Zweig der Definition von da.*

$$\begin{aligned} da(h') &= \frac{|H'|}{|P|} \cdot \frac{|H' \cap T|}{|H'|} - \frac{|H'|}{|P|} \cdot \frac{|T|}{|P|} \\ &= \frac{|H' \cap T|}{|P|} - \frac{|H'|}{|P|} \cdot \frac{|T|}{|P|} \end{aligned}$$

Wenn $\frac{|H'|}{|P|} \geq \sigma_m$, dann gilt wegen $|H' \cap T| \leq |H \cap T|$:

$$\begin{aligned} da(h') &\leq \frac{|H' \cap T|}{|P|} - \sigma_m \cdot \frac{|T|}{|P|} \\ &\leq \frac{|H \cap T|}{|P|} - \sigma_m \cdot \frac{|T|}{|P|} \end{aligned}$$

Wenn $\frac{|H|}{|P|} < \sigma_m$, wird $da(h')$ nach dem ersten Zweig der Definition 23 berechnet und ist daher minimal. Damit gilt insgesamt: $da(h') \leq da_{oe2}(h)$. ♢

Vergleich von da_{oe1} und da_{oe2}: Es ist $H = cov(h)$.

$$\begin{aligned} da_{oe1}(h) &= \frac{|H \cap T|}{|P|} \cdot \left(1 - \frac{|T|}{|P|}\right) \\ &= \frac{|H \cap T|}{|P|} - \frac{|H \cap T|}{|P|} \cdot \frac{|T|}{|P|} \\ da_{oe2}(h) &= \frac{|H \cap T|}{|P|} - \sigma_m \cdot \frac{|T|}{|P|} \end{aligned}$$

Man sieht direkt, daß da_{oe1} günstiger ist (d.h., den kleineren und damit genaueren Maximalwert ergibt), falls $\frac{|H \cap T|}{P} > \sigma_m$. Dieser Fall ist dann wahrscheinlich, wenn σ_m niedrig gewählt ist und die Zielgruppe T einen großen Anteil der Grundgesamtheit P bildet. Im anderen Fall ist eher die Optimumschätzfunktion da_{oe2} günstiger. Diese Überlegungen werden gestützt durch die in [Web98a] berichteten Experimente und durch Tabelle 6.11.

Dritte Optimumschätzfunktion Wrobel [Wro97] nennt folgende Optimumschätzfunktion da_{oe3} für da, die auf derselben Idee beruht wie da_{oe1}, aber die Abdeckung $|H'|$ der Spezialisierungen durch die Abdeckung $|H|$ der betrachteten Hypothese h und nicht durch ihre Zielgruppenabdeckung $|H \cap T|$ abschätzt.

$$da_{oe3}(h) = \frac{|H|}{|P|} \cdot \left(1 - \frac{|T|}{|P|}\right)$$

Weil $|H \cap T| \leq |H|$, kann da_{oe1} in vielen Fällen einen niedrigeren und damit genaueren Schätzwert liefern als da_{oe3}.

3.4 Folgerungsstärke

Die Folgerungsstärke (*implication intensity*) ist ein statistisches Kriterium zur Bewertung eines Zusammenhangs zwischen dem Voraussetzungs- und dem Folgerungsteil einer Regel $A \rightarrow B$. Die Folgerungsstärke geht zurück auf die Arbeit von Gras und Larher [GL93]. Fleury et al. untersuchen in [FDPB95] ihre Anwendung für die KDD. Demnach ist die Folgerungsstärke als statistisches Kriterium besonders geeignet für fehlerbehaftete Daten und Datenbanken geringeren Umfangs [FDPB95].

Der Folgerungsstärke liegt die Idee zugrunde, festzustellen, ob die Menge der Individuen $A \cap \overline{B}$, die einer Regel $A \rightarrow B$ widersprechen, signifikant kleiner ist als bei einer statistischen Unabhängigkeit zwischen A und B zu erwarten wäre. Die Folgerungsstärke berechnet näherungsweise die Wahrscheinlichkeit, daß die beobachtete oder eine kleinere Anzahl von Fällen der Regel $A \rightarrow B$ widersprechen, wenn A und B unabhängig sind. Die Folgerungsstärke *II* ist 1 minus dieser Wahrscheinlichkeit.

Die Folgerungsstärke dient hier dazu, die Stärke des Zusammenhangs zwischen einer Teilgruppe H und der Zielgruppe T zu bewerten. Da in der hier betrachteten Definition der Teilgruppenanalyse die Zielgruppe T fixiert ist, bewertet die Folgerungsstärke hier also die Qualität des Regelbedingungsteils H.

Definition 26 *Es sei $\overline{T} = P \setminus T$, $\lambda = |H| \cdot \frac{|\overline{T}|}{|P|}$, und Φ die Verteilungsfunktion $\int \phi$ der Standardnormalverteilung ϕ. Die Folgerungsstärke II(H) einer Teilgruppe H ist damit definiert wie folgt.*

$$II(H) = \begin{cases} 1 - \sum_{i=0}^{|\overline{T} \cap H|} \frac{\lambda^i}{i!} \cdot \mathbf{e}^{-\lambda} & \text{if } \lambda \leq 3 \\ 1 - \Phi\left(\frac{|\overline{T} \cap H| - \lambda}{\sqrt{\lambda}}\right) & else \end{cases}$$

Die Folgerungsstärke II(h) einer Teilgruppenbeschreibung h ist gleich der Folgerungsstärke II(H) der Teilgruppe $H = cov(h)$.

3.4.1 Herleitung der Folgerungsstärke

Die Definition der Folgerungsstärke kommt wie folgt zustande [FDPB95]. X und Y sind zwei zufällige Teilmengen einer Population mit gleichen Mächtigkeiten wie H beziehungsweise T. $\left|X \cap \overline{Y}\right|$ ist eine Zufallsvariable. Zur Bewertung der Folgerungsstärke einer Regel $i \in H \rightarrow i \in T$ vergleicht man die beobachtete Widersprechermenge $\left|H \cap \overline{T}\right|$ mit der Mächtigkeit der Widersprechermenge $\left|X \cap \overline{Y}\right|$, die für unabhängige Ereignisse X und Y zu erwarten ist. Nach [FDPB95] genügt $\left|X \cap \overline{Y}\right|$ einer hypergeometrischen Verteilung. Für große Datenbanken (also große $|P|$) läßt sich die hypergeometrische durch eine Binomialverteilung annähern, die wiederum durch eine Poissonverteilung zu approximieren ist. Die Wahrscheinlichkeit, daß eine Poisson-verteilte Zufallsvariable $\left|X \cap \overline{Y}\right|$ den Wert k annimmt, ist

$$\begin{aligned} Pr(\left|X \cap \overline{Y}\right| = k) &= \frac{\lambda^k}{k!} e^{-\lambda} \\ \lambda &= \frac{|H| \cdot |\overline{T}|}{|P|}. \end{aligned}$$

(Pr ist eine Abkürzung für *probability*, Wahrscheinlichkeit.) Der Zusammenhang zwischen H und T ist umso stärker, je stärker die Mächtigkeit der beobachteten Widersprechermenge $\left|H \cap \overline{T}\right|$ vom erwarteten Widerspruch $\left|X \cap \overline{Y}\right|$ nach unten abweicht. Die Wahrscheinlichkeit, daß die Zufallsvariable $\left|X \cap \overline{Y}\right|$ den beobachteten oder einen noch kleineren Wert annimmt, ist

$$\begin{aligned} Pr(\left|X \cap \overline{Y}\right| \leq \left|H \cap \overline{T}\right|) &= \sum_{i=0}^{|H \cap \overline{T}|} \frac{\lambda^i}{i!} e^{-\lambda} \quad mit \\ \lambda &= \frac{|H| \cdot |\overline{T}|}{|P|} \end{aligned}$$

Da die Berechnung der kumulierten Poissonverteilung sehr aufwendig ist, approximieren sie Fleury et al. für große λ ($\lambda \geq 3$) durch eine Normalverteilung. Der Erwartungswert und die Varianz der Poissonverteilung sind beide gleich λ, so daß die Verteilungfunktion Φ der Standardnormalverteilung wie folgt verwendet werden kann:

$$\begin{aligned} Pr(\left|X \cap \overline{Y}\right| \leq \left|H \cap \overline{T}\right|) &\approx \Phi\left(\frac{\left|H \cap \overline{T}\right| - \lambda}{\sqrt{\lambda}}\right) \quad mit \\ \lambda &= \frac{|H| \cdot |\overline{T}|}{|P|} \geq 3 \end{aligned}$$

3.4.2 Verhalten der Folgerungsstärke

Die Abbildungen 3.2 (a) und (b) veranschaulichen das Verhalten der Folgerungsstärke für Regeln $i \in H \rightarrow i \in T$ abhängig von der Größe $|H|$ und der Größe ihrer Unterstützermenge $|H \cap T|$. Die Abbildungen sind analog zu den Abbildungen 3.1. In beiden Abbildungen (a) und (b) läuft die Größe der Teilgruppe $|H|$ von 30 bis 230, abgetragen auf der x-Achse. Auf der y-Achse ist die Folgerungsstärke II(h) abgetragen. In Abbildung (a) decken alle Teilgruppen H 30 Zielindividuen ab ($|H \cap T| = 30$). In Abbildung (b) ist der Umfang der Unterstützermenge $|H \cap T| = |H| - 30$, das heißt, die Widersprechermenge $|H \cap \overline{T}|$ enthält immer 30 Individuen. Die Population P besteht aus 500, 1000, 1500 und 2000 Individuen, die Zielgruppe T enthält immmer 200 Individuen.

Abbildung 3.2 Verhalten der Interessantheitsfunktion $II_\phi(h)$ (a) abhängig von der Größe der Teilgruppe $|H| \in [30, 230]$ bei festem $|H \cap T| = 30$, und (b) abhängig von der Größe der Teilgruppe $|H| \in [30, 230]$ bei festem $|H \cap \overline{T}| = 30$. Die Population P besteht aus 500, 1000, 1500 und 2000 Individuen, die Zielgruppe T enthält 200 Individuen.

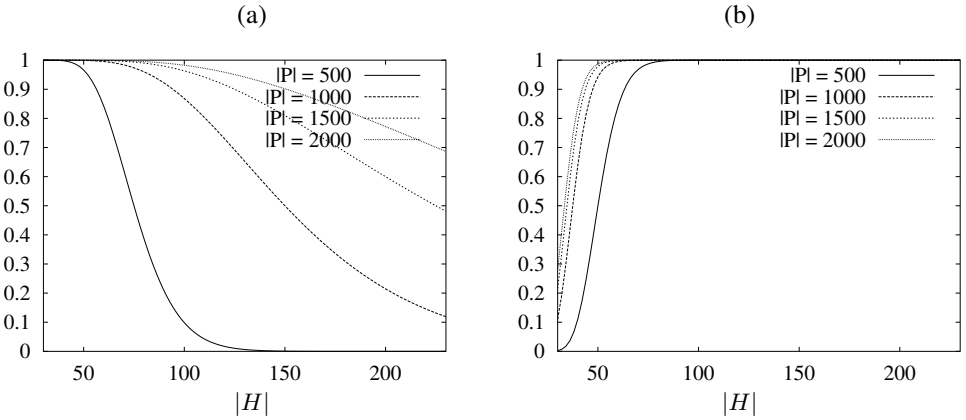


Abbildung (a) zeigt, daß die Folgerungsstärke fällt, wenn die Größe der Abdeckung $|H|$ wächst, aber die Unterstützermenge $|H \cap T|$ gleich groß bleibt. Abbildung (b) zeigt, daß die asymmetrische Verteilungsungewöhnlichkeit steigt mit der Größe der Abdeckung $|H|$, wenn auch die Unterstützermenge $|H \cap T|$ wächst.

3.4.3 Eine Optimumschätzfunktion

Wie für die Verteilungsungewöhnlichkeit kann man auch für die Folgerungsstärke eine Optimumschätzfunktion aus dem Prinzip ableiten, daß die bestmögliche Spezialisierung einer Hypothese h die Abdeckung genau der Individuen verhindert, die der Regel $i \in H \rightarrow i \in T$ widersprechen, so daß die resultierende Regel $i \in H' \rightarrow i \in T$ widerspruchsfrei wird und dabei möglichst viele Individuen abdeckt. Für den Normalverteilungszweig der Folgerungsstärke ergibt dies die folgende Optimumschätzfunktion II_{oeN}.

Definition 27

$$II_{oeN}(h) \quad = \quad 1 - \Phi\left(-\sqrt{|cov(h) \cap T| \cdot \frac{|\overline{T}|}{|P|}}\right).$$

Proposition 4 *Die Optimumschätzfunktion II_{oeN} bestimmt eine obere Schranke für die Folgerungsstärke berechnet nach dem Normalverteilungszweig II_N, die Spezialisierungen h' einer Hypothese h im besten Fall erreichen können, sofern h Zielgruppenelemente abdeckt, also $cov(h) \cap T \neq \emptyset$. Das heißt:*

$II_N(h') \leq II_{oeN}(h)$ *für alle h' mit $h \succ h'$.*

Die Folgerungsstärke $II(h)$ einer Teilgruppenbeschreibung hängt ab von der Mächtigkeit ihrer Abdeckung $|cov(h)|$ und von der Mächtigkeit ihrer Widersprechermenge $|\overline{T} \cap cov(h)|$. Die Größen $|P|$ und $|\overline{T}|$ sind für h und alle Spezialisierungen h' von h mit $h \succ h'$ konstant. Für den Beweis sollen gelten:

$$d = |\overline{T} \cap cov(h)|$$
$$a = |cov(h)|$$
$$c = \frac{|\overline{T}|}{|P|}$$

Damit kann $II(h)$ geschrieben werden als $II(d, a)$. Durch Spezialisierung von h zu h' kann sich die Abdeckung von $cov(h) = a$ zu $cov(h') = a - x$ verringern und die Widersprechermenge von $\overline{T} \cap cov(h) = d$ zu $\overline{T} \cap cov(h') = d - y$. Die Folgerungsstärke von Spezialisierungen h' von h kann damit geschrieben werden als

$$II(d - y, a - x) = 1 - \Phi\left(\frac{(d - y) - (a - x)c}{\sqrt{(a - x)c}}\right).$$

Dabei ist immer $x \geq 0$ und $y \geq 0$ und $x \geq y$, weil sich die Abdeckung bei Spezialisierung mindestens so stark verkleinert wie die Widersprechermenge. Außerdem gelten $y \leq d$ und $x < a$ und nach Voraussetzung $a > 0$ und $a > d$.

Die Optimumschätzfunktion berechnet die Folgerungsstärke einer hypothetischen optimalen Spezialisierung h^* von h, die genau die Widersprechermenge aus der Abdeckung von h entfernt. Die Folgerungsstärke von h^* ist damit

$$II_{oeN}(h) = II(h^*) = II(d - d, a - d) = 1 - \Phi\left(\frac{(d - d) - (a - d)c}{\sqrt{(a - d)c}}\right).$$

Um zu zeigen, daß $II(h^*)$ eine Optimumschätzfunktion ist, ist also zu beweisen, daß unter den genannten Randbedingungen gilt:

Proposition 5 *Es seien $x \geq 0$, $y \geq 0$, $x \geq y$, $x < a$, $y \leq d$, $a > 0$ und $a > d$.*

$$\begin{array}{rrl} & II_N(h') \leq & II_{oeN}(h) = II(h^*) \\ \iff \quad 1 - \Phi\left(\frac{(d - y) - (a - x)c}{\sqrt{(a - x)c}}\right) \leq & & 1 - \Phi\left(\frac{(d - d) - (a - d)c}{\sqrt{(a - d)c}}\right). \end{array}$$

Für den Beweis von Proposition 5 werden zunächst die beiden folgenden Ungleichungen 3.1 und 3.2 bewiesen.

$$(3.1)\quad 1 - \Phi\left(\frac{d - ac}{\sqrt{ac}}\right) \geq 1 - \Phi\left(\frac{d - (a - 1)c}{\sqrt{(a - 1)c}}\right)$$

$$(3.2)\quad 1 - \Phi\left(\frac{d - ac}{\sqrt{ac}}\right) \leq 1 - \Phi\left(\frac{(d - 1) - (a - 1)c}{\sqrt{(a - 1)c}}\right)$$

Ungleichung (3.1) besagt, daß die Folgerungsstärke abnimmt, wenn die Abdeckung H um ein Element kleiner wird und der Umfang der Widersprechermenge $d = |H \cap \overline{T}|$ gleich bleibt. Diese Ungleichung beschreibt das Verhalten der Folgerungsstärke bei einer (hypothetischen) minimalen ungünstigen Spezialisierung einer Hypothese, die genau ein Individuum aus $|H \cap T| = a$ ausschließt, also ein Individuum, das die Regel erfüllt.

Ungleichung (3.2) besagt, daß die Folgerungsstärke steigt, wenn sowohl die Abdeckung einer Regel als auch ihre Widersprechermenge um 1 kleiner werden. Diese Ungleichung beschreibt das Verhalten der Folgerungsstärke bei einer (hypothetischen) minimalen günstigen Spezialisierung einer Regel, die genau ein der Regel widersprechendes Individuum aus ihrer Widersprechermenge d ausschließt.

Beweis 5 (Ungleichung (3.1)) *Es gilt* $\Phi(-z) = 1 - \Phi(z)$.

$$
\begin{aligned}
& 1 - \Phi\left(\frac{d - ac}{\sqrt{ac}}\right) \geq 1 - \Phi\left(\frac{d - (a-1)c}{\sqrt{(a-1)c}}\right) \\
\iff \quad & \Phi\left(\frac{ac - d}{\sqrt{ac}}\right) \geq \Phi\left(\frac{(a-1)c - d}{\sqrt{(a-1)c}}\right) \\
\iff \quad & \frac{ac - d}{\sqrt{ac}} \geq \frac{(a-1)c - d}{\sqrt{(a-1)c}} \\
\iff \quad & \frac{ac}{\sqrt{ac}} - \frac{d}{\sqrt{ac}} \geq \frac{(a-1)c}{\sqrt{(a-1)c}} - \frac{d}{\sqrt{(a-1)c}} \\
\iff \quad & \underbrace{\sqrt{ac}}_{i} - \underbrace{\frac{d}{\sqrt{ac}}}_{ii} \geq \underbrace{\sqrt{(a-1)c}}_{i'} - \underbrace{\frac{d}{\sqrt{(a-1)c}}}_{ii'}
\end{aligned}
$$

Jetzt wird gezeigt, daß $i \geq i'$ *und* $ii \leq ii'$.

$$
\begin{aligned}
i \geq i' : \quad & \sqrt{ac} \geq \sqrt{(a-1)c} && && \surd \\
ii \leq ii' : \quad & \frac{d}{\sqrt{ac}} \leq \frac{d}{\sqrt{(a-1)c}} && \iff \quad \sqrt{a} \geq \sqrt{a-1} && \surd
\end{aligned}
$$

$\diamondsuit$

Beweis 6 (Ungleichung (3.2))

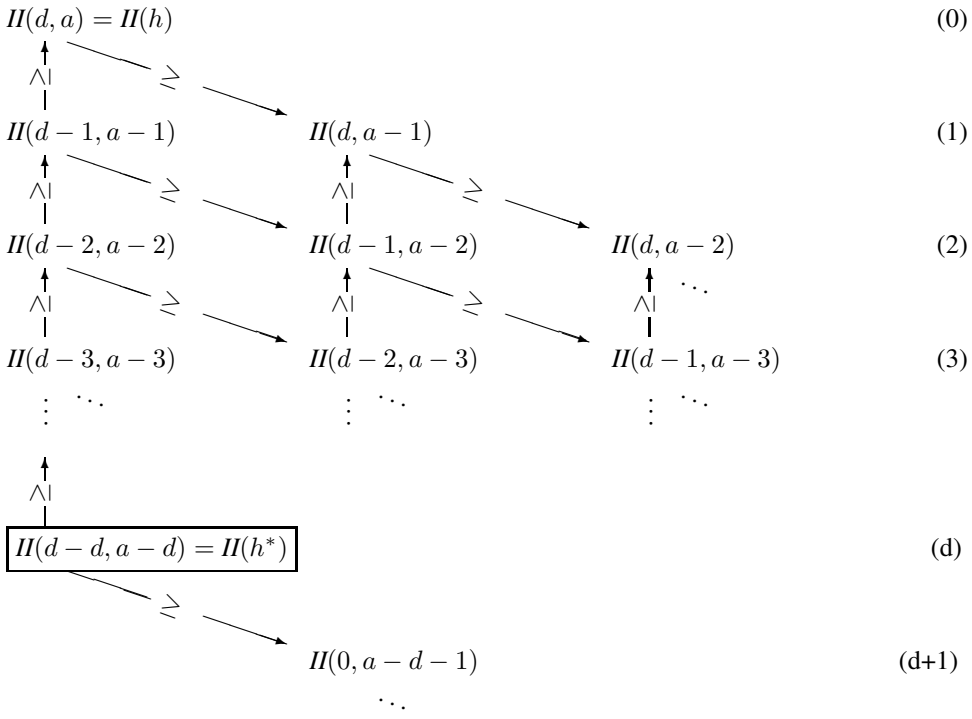
$$
\begin{aligned}
& 1 - \Phi\left(\frac{d - ac}{\sqrt{ac}}\right) && \leq && 1 - \Phi\left(\frac{(d-1) - (a-1)c}{\sqrt{(a-1)c}}\right) \\
\iff \quad & \Phi\left(\frac{d - ac}{\sqrt{ac}}\right) && \geq && \Phi\left(\frac{(d-1) - (a-1)c}{\sqrt{(a-1)c}}\right) \\
\iff \quad & \frac{d - ac}{\sqrt{ac}} && \geq && \frac{(d-1) - (a-1)c}{\sqrt{(a-1)c}} \\
\iff \quad & \frac{d - ac}{\sqrt{a}} && \geq && \frac{(d-1) - (a-1)c}{\sqrt{a-1}}
\end{aligned}
$$

Da $d \leq a$, *kann man* $d = a - n$ *mit* $n \geq 0$ *setzen.*

$$
\begin{aligned}
& \frac{a - n - ac}{\sqrt{a}} && \geq && \frac{(a - n - 1) - (a-1)c}{\sqrt{a-1}} \\
\iff \quad & \frac{a - ac - n}{\sqrt{a}} && \geq && \frac{(a-1) - (a-1)c - n}{\sqrt{a-1}} \\
\iff \quad & \frac{a(1-c) - n}{\sqrt{a}} && \geq && \frac{(a-1)(1-c) - n}{\sqrt{a-1}} \\
\iff \quad & \frac{a(1-c)}{\sqrt{a}} - \frac{n}{\sqrt{a}} && \geq && \frac{(a-1)(1-c)}{\sqrt{a-1}} - \frac{n}{\sqrt{a-1}} \\
\iff \quad & \underbrace{\sqrt{a}(1-c)}_{i} - \underbrace{\frac{n}{\sqrt{a}}}_{ii} && \geq && \underbrace{\sqrt{a-1}(1-c)}_{i'} - \underbrace{\frac{n}{\sqrt{a-1}}}_{ii'}
\end{aligned}
$$

Jetzt wird gezeigt, daß $i \geq i'$ *und* $ii \leq ii'$.

Abbildung 3.3 Effekte hypothetischer minimaler Spezialisierungsoperationen auf die Folgerungsstärke *II*.



$$
\begin{array}{lrcll}
i \geq i': & \sqrt{a}(1-c) \geq \sqrt{a-1}(1-c) & \Longleftrightarrow & \sqrt{a} \geq \sqrt{a-1} & \surd \\
ii \leq ii': & \dfrac{n}{\sqrt{a}} \leq \dfrac{n}{\sqrt{a-1}} & \Longleftrightarrow & \sqrt{a-1} \leq \sqrt{a} & \surd
\end{array}
$$

$\diamondsuit$

Beweis 7 (zu Satz 5) *Die Abbildung 3.3 illustriert, wie der Satz 5 aus den Ungleichungen (3.1) und (3.2) folgt.* $II(d, a)$ *steht für die Folgerungsstärke, berechnet nach dem Normalverteilungszweig, einer Teilgruppenbeschreibung* h *mit* $cov(h) = a$ *und* $cov(h) \cap \overline{T} = d$*. Oben links in der Abbildung steht die Folgerungsstärke der noch nicht spezialisierten Regel,* $II(h) = II(d, a)$*, mit einer Abdeckung* $cov(h)$ *mit Mächtigkeit* a *und einer Widersprechermenge mit Umfang* d*. Darunter sind zeilenweise die Folgerungsstärken für die Abdeckungen und Widersprechermengen angeordnet, die sich durch Spezialisierungsoperationen ergeben können. In einer Zeile stehen jeweils die Folgerungsstärken, die aus Spezialisierungsoperationen resultieren, die gleich viele Fälle aus der Abdeckung eliminieren. Die Anzahl der eliminierten Fälle ist ganz rechts in der Zeile in Klammern angegeben. Innerhalb der Zeilen nimmt die Anzahl der eliminierten widersprechenden Fälle von links nach rechts ab. Ungünstige minimale Spezialisierungen, die die Abdeckung* a*, aber nicht die Widersprechermenge* d *einer Regel um 1 vermindern, sind schräg übereinander angeordnet (* $\searrow$ *); die Folgerungsstärke der spezialisierten Regel steht unten. Laut Ungleichung (3.1) senken ungünstige Spezialisierungen die Folgerungsstärke.*

Günstige minimale Spezialisierungen, die die Abdeckung a *und die Widersprechermenge* d *der Regel um 1 vermindern, sind senkrecht übereinander angeordnet (*$\uparrow$*); die Fol-*

gerungsstärke der spezialisierten Regel steht unten. Nach Ungleichung (3.2) erhöht eine solche günstige Spezialisierung die Folgerungsstärke.

Die Folgerungsstärke der optimalen Spezialisierung h^*, die den Optimumschätzwert ergibt, ist eingerahmt. Die Pfeile, die den Ungleichungen (3.1) und (3.2) entsprechen, zeigen, daß $II(h^*)$ der maximale Wert der Folgerungstärke ist, den eine Spezialisierung von h annehmen kann.

Nicht eingezeichnet ist der mimimale Wert $II(d, d)$. Dies ist die Folgerungsstärke der Regel, deren Abdeckung gleich der Widersprechermenge d der ursprünglichen Regel ist. Dieser Wert ergibt sich, wenn eine Spezialisierung genau die unterstützenden Fälle aus der Abdeckung der ursprünglichen Regel eliminiert.

In vielen Anwendungen ist die Hypothesensprache nicht so ausdrucksstark, daß sie es erlaubt, Teilgruppen mit jeder möglichen Kombination von d (der Umfang der Widersprechermenge) und a (der Umfang der Abdeckung) zu beschreiben. Die Abbildung zeigt jedoch, daß $II_{oeN}(h) = II(0, |cov(h) \cap T|)$ für $|cov(h) \cap T| > 0$ von keiner Spezialisierung von h übertroffen werden kann. Damit ist gezeigt, daß Satz 5 gilt und folglich II_{oeN} eine Optimumschätzfunktion für den Normalverteilungszweig der Folgerungsstärke ist.

3.4.4 Sichere Anwendung der Optimumschätzfunktion

Die Folgerungsstärke bewertet auch sehr spezielle Hypothesen mit geringer Abdeckung sehr hoch, sofern sie eine sehr kleine Widersprechermenge aufweisen. Daher ist es oft sinnvoll, die Folgerungsstärke mit einer Mindesthäufigkeitsforderung zu kombinieren, die eine Schwelle σ_{II} für die minimale relative Abdeckung $\frac{|H|}{|P|}$ einer akzeptablen Hypothese H festlegt.

Wenn die Mindesthäufigkeitsschwelle σ_{II} ausreichend groß ist, wird die Folgerungsstärke aller akzeptablen Regeln nach dem Normalverteilungszweig in Definition 26 berechnet. Die Mindesthäufigkeit σ_{II} ist ausreichend hoch, wenn $\sigma_{II} > \frac{3}{|\overline{T}|}$ gilt, denn dann ist $\lambda = |cov(h)| \cdot \frac{|\overline{T}|}{|P|}$ für alle akzeptablen Hypothesen größer als 3.

$$\begin{aligned}
\lambda &= |cov(h)| \cdot \frac{|\overline{T}|}{|P|} \\
&\geq \sigma_{II} \cdot |P| \cdot \frac{|\overline{T}|}{|P|} \\
&\overset{\left(\sigma_{II} > \frac{3}{|\overline{T}|}\right)}{\geq} \frac{3}{|\overline{T}|} \cdot |P| \cdot \frac{|\overline{T}|}{|P|} \\
&= 3.
\end{aligned}$$

Beispiel 37 *Die Population P besteht aus $|P| = 1000$ und die Zielgruppe T aus $|T| = 500$ Individuen. Wenn $\sigma_{II} \geq \frac{7}{1000}$, gilt für alle akzeptablen Regeln $\lambda \geq 3$.*

$$\lambda \;=\; |H| \cdot \frac{|\overline{T}|}{|P|} \;\geq\; 7 \cdot \frac{500}{1000} \;>\; 3.$$

Wenn $\sigma_{II} > \frac{3}{|\overline{T}|}$, ist die Optimumschätzfunktion II_{oeN} für alle potenziell akzeptablen Regeln anwendbar und ermöglicht eine sichere Beschränkung das Suchraums, das heißt, sie eliminert keine Regeln, die später der Poissonzweig der Folgerungsstärke akzeptieren könnte.

3.4.5 Verbesserung der Optimumschätzfunktion

Die Optimumschätzfunktion läßt sich noch weiter verbessern, wenn man die Mindesthäufigkeit der bestmöglichen Regel berücksichtigt. Die Optimumschätzfunktion II_{oeN} berechnet als Optimum der Folgerungsstärken aller Spezialisierungen einer Regel h mit einer Abdeckung vom Umfang a und einer Widerspruchsmenge vom Umfang d die Folgerungsstärke der (hypothetischen) Regel h_{opt} mit einer Abdeckung vom Umfang $a - d$ und einer Widersprechermenge vom Umfang 0. Die Mächtigkeit der Abdeckung a soll der Mindesthäufigkeitsforderung genügen. Dabei kann jedoch $a - d$ kleiner sein als die Mindesthäufigkeit. Die hypothetische beste Regel h_{opt} ist dann nicht akzeptabel, weil sie die Mindesthäufigkeit nicht erreicht. In diesem Fall ist die Anzahl der Fälle, die die Regel h unterstützen, kleiner als die minimal geforderte Anzahl von abgedeckten Fällen. Jede akzeptable Spezialisierung der Regel h muß also auch widersprechende Fälle abdecken. Es sei $mincov$ die Anzahl der Fälle, die eine Regel mindestens abdecken muss, um das Mindesthäufigkeitskriterium zu erfüllen. Für die Regel h gelte $a - d < mincov$. Alle akzeptablen Spezialisierungen von h müssen also mindestens $mincov - a$ widersprechende Fälle abdecken. Aus der Abbildung 3.3 ist nun ersichtlich, daß die bestmögliche Spezialisierung h^*_σ von h mit ausreichender Abdeckung $mincov$ diejenige Regel ist, die möglichst wenige widersprüchliche Fälle abdeckt, nämlich $mincov - a$, also gerade so viele, daß zusammen mit den unterstützenden Fällen die Mindestabdeckung erreicht wird. In der Abbildung 3.3 sind alle Regeln mit gleich großer Abdeckung in einer Zeile angeordnet. Die jeweils höchste Folgerungsstärke auf jeder Zeile hat die Regel mit der kleinsten Widersprechermenge, die am weitesten links steht. Die Folgerungsstärke dieser Regel ist auch größer als die Folgerungsstärken aller Regeln mit größerer Abdeckung. Damit ergibt sich die folgende Optimumschätzfunktion $II_{oeN,\sigma}$ unter Berücksichtigung einer Mindesthäufigkeit σ_{II}.

Definition 28

$$\begin{aligned} II_{oeN,\sigma}(d,a) &= II(mincov - a, mincov) \\ &= 1 - \Phi\left(\frac{(mincov - a) - \lambda}{\sqrt{\lambda}}\right) \end{aligned}$$

mit $\lambda = mincov \cdot \frac{|\overline{T}|}{|P|}$ ist eine Optimumschätzfunktion für die Folgerungsstärke, wenn eine Mindesthäufigkeit $mincov = \lceil \sigma_{II} \cdot |P| \rceil$ für akzeptable Regeln gefordert ist und $\sigma_{II} > \frac{3}{|\overline{T}|}$ gilt.

3.5 Zusammenfassung

In diesem Kapitel wurden Interessantheitsfunktionen genannt, die in der KDD gebräuchlich sind. Für einige der Interessantheitsfunktionen konnten Optimumschätzfunktionen hergeleitet werden. Am Beispiel dieser Interessantheitsfunktionen wurden Möglichkeiten zur Herleitung und Verbesserung von Optimumschätzfunktionen aufgezeigt. Sie sollen in nachfolgenden Experimenten dazu dienen, den Nutzen von Optimumschätzfunktionen für die Beschränkung des Suchraums zu evaluieren.

Kapitel 4

Ebenenweise Suche in ILP–Suchräumen

Dieses Kapitel präsentiert einen Ansatz, der das Vorgehen des Apriori-Algorithmus bei der Suche im Hypothesenraum und bei der Beschränkung des Suchraums auf die komplexeren Hypothesenräume beim Finden interessanter Teilgruppen in relationalen Datenbanken überträgt.

4.1 Ein erweiterter Suchalgorithmus

4.1.1 Beschränkung des Suchraums

In Abschnitt 2.4.3 wurde ein einfacher Algorithmus zur Suche interessanter Teilgruppen eingeführt. Er realisiert die Suche mittels eines Spezialisierungsoperators ρ und beschränkt den Suchraum, indem er Teilgruppenbeschreibungen, die das Beschränkungskriterium erfüllen, von der weiteren Spezialisierung ausschließt. Dieses Vorgehen hat jedoch die in Abschnitt 1.4.2 erläuterte Schwäche, nur die Teilgruppenbeschreibungen aus dem Suchraum zu entfernen, die in einer Sequenz von erlaubten Elementarspezialisierungen von einer kappbaren Teilgruppenbeschreibung ausgehend erzeugt werden. Teilgruppenbeschreibungen, die ebenfalls von der kappbaren Hypothese subsumiert sind, für deren Erzeugung aber eine andere Sequenz von Elementarspezialisierungen vorgesehen ist, werden nicht aus dem Suchraum eliminiert.

Um alle Subsumtionsbeziehungen im Hypothesenraum zur Beschränkung des Suchraums nützen zu können, kann man sie explizit überprüfen. Eine entsprechend erweiterte Version des Grund-Algorithms 2.16 zeigt Tafel 4.1. Der erweiterte Algorithmus sammelt in der Auswertungsphase alle kappbaren Hypothesen in einer Menge P (Zeile 19'), und gleicht in der Generierungsphase alle neu erzeugten Hypothesen mit der Menge P ab. Eine Hypothese wird nur in die Menge S aufgenommen, wenn sie nicht von einer Hypothese $h \in P$ θ-subsumiert sind. In Zeile 10' werden neu erzeugte Hypothesen mit den gesammelten kappbaren Hypothesen abgeglichen und nötigenfalls aus der Menge S' entfernt.

Im allgemeinen ist es jedoch sehr aufwendig festzustellen, ob zwei Hypothesen in einer θ-Subsumtionsbeziehung zueinander stehen. Die Kosten der θ-Subsumtionstests können sogar die durch die Suchraumbeschränkung erzielten Aufwandseinsparungen zunichte machen [Lin94]. Daher verzichten einige ILP–Systeme auf die Suchraumbeschränkung durch explizite Subsumtionstests [Wro97, FL01].

Tafel 4.1 Algorithmus für die ebenenweise Suche nach interessanten Teilgruppen in einem durch θ-Subsumtion geordneten Suchraum.

Eingabe: Datenbank D, pop, ρ
Ausgabe: Menge akzeptierter Regeln H

1. search(D, pop, ρ): H
2. $H = \emptyset$;
2.' $P := \emptyset$;
3. $S = \{\{pop\}\}$;
4. while $S \neq \emptyset$ do
5. $S' =$ cand_gen(S);
6. $S =$ cand_test(S', H, D);
7. return H;

8. cand_gen(S, ρ): S'
9. $S' = \emptyset$;
10. for all $h \in S : S' = S' \cup \rho^1(h)$;
10'. for all $h \in S'$: if $\exists h' \in P : h' \succ_\theta h$ then $S' := S' \setminus \{h\}$;
11. return S';

12. cand_test(S', H, D): S
13. $S := \emptyset$;
14. for all $h \in S'$
15. $erg :=$ eval(h, D);
16. if $erg = accept$ then
17. accept(h, H);
18. if prune_accepted then $erg := prune$ else $erg := keep$;
19. if $erg = keep$ then $S := S \cup \{h\}$;
19'. if $erg = prune$ then $P := P \cup \{h\}$;
20. return S;

4.1.2 Suchraumbeschränkung durch die Teilmengenbedingung

Einen Suchalgorithmus, der die Teilmengenbedingung zur Suchraumbeschränkung verwendet, zeigt Tafel 4.2. Die Hypothesensprache besteht aus Teilmengen einer Literalmenge $Lits$. Sie enthält alle möglichen Teilmengen von $Lits$ und ist damit gleich der Potenzmenge 2^{Lits}. Eine Hypothese h subsumiert eine Hypothese h', wenn h, als Literalmenge betrachtet, eine Teilmenge von h' ist. Eine elementare Spezialisierungsoperation besteht darin, einer Hypothese h ein Literal $l \in Lits$ anzufügen. Um Redundanz zu vermeiden, ist eine Reihenfolge $>_l$ für die Literale in $Lits$ definiert. Der Spezialisierungsoperator fügt Literale nur in der Reihenfolge $>_l$ aneinander.[1]

Definition 29 *Der Spezialisierungsoperator, der in Schritt 10 des Algorithmus eingesetzt wird, ist wie folgt definiert:*

$$\rho(h, Lit) = \{h \cup \{l\} | l \in Lit \text{ und } \forall l' \in h \;\; l >_l l'\}$$

Die Teilmengen von $Lits$, die i Literale enthalten, bilden die Suchebene i. In Schritt 10' überprüft der Algorithmus 4.2. für jede neu generierte Hypothese die Teilmengenbedingung, die besagt, daß alle zur vorhergehenden Suchebene $i-1$ gehörenden Hypothesen h'

[1] Der Apriori-Algorithmus realisiert die Hypothesengenerierung etwas anders. Eine neue Hypothese h_3 kommt zustande, indem zwei Hypothesen $h_1 = \{l_1, \ldots, l_n, l_{n+1}\}$ und $h_2 = \{l_1, \ldots, l_n, l_{n+2}\}$, die sich nur im letzten Literal unterscheiden, zu $h_3 = \{l_1, \ldots, l_n, l_{n+1}, l_{n+2}\}$ zusammengeführt werden.

Tafel 4.2 Suchalgorithmus mit Teilmengenbedingung.

Eingabe: Datenbank D, pop, ρ
Ausgabe: Menge akzeptierter Regeln H

1. search(D, pop, ρ): H
2. $H = \emptyset$;
3. $S = \{\{pop\}\}$;
4. while $S \neq \emptyset$ do
5. $\quad S' =$ cand_gen(S);
6. $\quad S =$ cand_test(S', H, D);
7. return H;

8. cand_gen(S, ρ): S'
9. $S' = \emptyset$;
10. for all $h \in S : S' = S' \cup \rho^1(h)$;
10'. for all $h \in S'$: for all $l \in h$: if $h \setminus \{l\} \notin S$ then $S' := S' \setminus \{h\}$;
11. return S';

12. cand_test(S', H, D): S
13. $S := \emptyset$;
14. for all $h \in S'$
15. $\quad erg :=$ eval(h, D);
16. $\quad$ if $erg = accept$ then
17. $\quad\quad$ accept(h, H);
18. $\quad\quad$ if prune_accepted then $erg := prune$ else $erg := keep$;
19. $\quad$ if $erg = keep$ then $S := S \cup \{h\}$;
20. return S;

mit $h' \succ h$ spezialisierbar sein müssen. Dies sind genau die Hypothesen h', für die gilt: $h' = h \setminus \{l\}$ für jedes $l \in h$. Alle zur Ebene $i-1$ gehörenden spezialisierbaren Hypothesen befinden sich zu während der Generierungsphase der Ebene i in der Menge S. Anders als der Algorithmus 4.1 muß der Algorithmus 4.2 daher nicht alle kappbaren Hypothesen in einer Menge P speichern. Daher kommen im Algorithmus 4.2 die Schritte 2' und 19' des Algorithmus 4.1 nicht vor.

Der Suchalgorithmus 4.2 und die Teilmengenbedingung sind anwendbar, weil die Hypothesensprache 2^{Lits} unter der Teilmengenbeziehung $\subseteq$ abgeschlossen ist, das heißt, mit einer Hypothese h sind auch alle ihre Teilmengen h' mit $h' \subseteq h$ in der Hypothesensprache enthalten. Wegen der Abgeschlossenheit der Sprache genügt die Menge S, um die durchsuchten und die noch zu durchsuchenden Bereiche des Hypothesenraums zu repräsentieren [MT97a].

Wie in Abschnitt 2.4.3 erläutert wurde, sind ILP–Hypothesensprachen im allgemeinen jedoch nicht unter der θ-Subsumtion oder der Teilmengenbeziehung abgeschlossen. So gehören nicht-(individuen)verbundene oder nicht-reduzierte Klauseln oder Teilgruppenbeschreibungen nicht zur beabsichtigten Hypothesensprache, auch wenn sie von Hypothesen in der Hypothesensprache subsumiert sind. Im folgenden Abschnitt werden Möglichkeiten zur Einschränkung der Hypothesensprache eingeführt, die es erlauben, bestimmte Teilmengen der Literalliste $Lits$ aus der Hypothesensprache auszuschließen und dennoch die Teilmengenbedingung einzusetzen.

4.1.3 Einschränkungen der Hypothesensprache

Um Teilmengen der Literalliste $Lits$ aus der Hypothesensprache ausschließen zu können, werden excludes- und requires-Listen eingeführt. Excludes- oder requires-Listen sind einem Literal zugeordnet. Die excludes-Liste *excl*(l) eines Literals l führt Literale auf, die nicht gemeinsam mit dem Literal l in einer Hypothese vorkommen dürfen. Sie dient dazu, widersprüchliche oder irrelevante Kombinationen von Literalen in Hypothesen zu verhindern. Die requires-Liste *req*(l) eines Literals l ist eine Liste von Literalen oder Listen von Literalen. Ein einzelnes Literal l' als Element einer requires-Liste ist gleichwertig zu einer ein-elementigen Literalliste $\{l'\}$. Eine requires-Liste *req*(l) für ein Literal l besagt, daß l nur in solchen Hypothesen h vorkommen darf, die mindestens ein Literal l' aus jeder Literalliste L in *req*(l) enthalten.

Definition 30 *Gegeben seien eine Literalliste $Lits$ sowie excludes- und requires-Listen $excl(l)$ beziehungsweise $req(l)$ für die Literale $l \in Lits$. Das Prädikat $legal$, das die Zulässigkeit einer Hypothese $\subseteq Lits$ feststellt, ist definiert wie folgt:*

$$legal(h) \text{ gdw } \forall l \in h : h \setminus \{l\} \cap \textit{excl}(l) = \emptyset \text{ und } \forall L \in \textit{req}(l) : h \cap L \neq \emptyset$$

Hypothesen, die *excl*- oder *req*-Bedingungen verletzen, heißen unzulässig. Hypothesen, die den *excl*- oder *req*-Bedingungen genügen, heißen zulässig.

Requires- und excludes-Listen können für alle Literale in $Lits$ angegeben werden. Die *req*-Listen dürfen keine Zyklen enthalten und müssen zur Ordnung $>_l$ auf der Literalliste $Lits$ passen, das heißt, die requires-Liste eines Literals l darf nur Literale l' mit $l >_l l'$ enthalten.

Anpassung des Algorithmus für *excl*-und *req*-Bedingungen

Die Einschränkungen der Hypothesensprache, die sich aus den *excl*- und *req*-Listen ergeben, lassen sich ohne Verlust der Vollständigkeit und Korrektheit in den Suchalgorithmus integrieren, also so, daß Hypothesen weder fälschlicherweise erzeugt und ausgewertet noch fälschlicherweise aus dem Suchraum ausgeschlossen werden. Dazu müssen der Spezialisierungsoperator und die Überprüfung der Teilmengenbedingung so angepaßt werden, daß der Spezialisierungsoperator nur noch zulässige Hypothesen erzeugt und die Teilmengenbedingung nur noch das Vorkommen von zulässigen Hypothesen in der Menge S überprüft.

Definition 31 *Der modifizierte Spezialisierungsoperator, der in Schritt 10 des Algorithmus angewendet wird, lautet:*

$$\rho(h, Lit) = \{h \cup \{l\} \mid l \in Lit \text{ und } \forall_{l' \in h} l >_l l' \text{ und } legal(h)\}.$$

Der Schritt 10' des Algorithmus, der die Teilmengenbedingung überprüft, wird abgeändert wie folgt:

10'. for all $h \in S'$: for all $l \in h$:
if legal($h \setminus \{l\}$) und $h \setminus \{l\} \notin S$ then $S' := S' \setminus \{h\}$;

Korrektheit und Vollständigkeit des modifizierten Algorithmus

Vollständigkeit des Algorithmus. Vollständigkeit bedeutet hier, daß der Algorithmus alle zulässigen potentiell akzeptablen Hypothesen erzeugt und sie nicht fälschlicherweise aus dem Suchraum ausschließt. Die folgenden Überlegungen zeigen, daß die *excl*-Bedingungen und ihre Behandlung im Algorithmus die Vollständigkeit der Hypothesenerzeugung nicht

beeinträchtigen. Der Spezialisierungsoperator erzeugt keine Hypothese h, die *excl*-Bedingungen verletzt. Eine solche Hypothese h kann daher auch nicht weiter spezialisiert werden. Wenn eine Hypothese h eine *excl*-Bedingung verletzt, so verletzen auch alle Hypothesen $h \cup \{l\}$ für beliebiges l diese *excl*-Bedingung.

Genauso führen auch die *req*-Bedingungen nicht zum Verlust möglicher Lösungen, denn es kann nicht vorkommen, daß eine zulässige Hypothese h eine unzulässige Spezialisierung $h \cup \{l\}$ besitzt, deren Spezialisierung $h \cup \{l, \ldots, l'\}$ wieder zulässig ist. Denn dazu müßte l' die *req*-Bedingungen für $h \cup \{l\}$ nachträglich erfüllen. Das heißt, daß l' in der *req*-Liste von l vorkommt. Laut Voraussetzung darf aber jedes Literal nur bezüglich der Literalordnung kleinere Literale erfordern. Wenn das Literal l' in der *req*-Liste von l vorkommt, muß also l' kleiner sein als l und daher auch vor l an die Hypothese h angefügt werden.

Die modifizierte Teilmengenbedingung, die nur das Vorkommen zulässiger Hypothesen in der Menge S überprüft, ist weniger restriktiv als die nicht modifizierte Teilmengenbedingung. Sie schließt nur einen Teil der Hypothesen aus dem Suchraum aus, die die nicht modifizierte Teilmengenbedingung ausschließen würde. Die Modifikation kann die Vollständigkeit des Algorithmus also nicht verletzen.

Korrektheit des Algorithmus. Korrektheit des Algorithmus bedeutet hier, daß er nicht zu viele Hypothesen erzeugt und auswertet. Das heißt erstens, daß der Algorithmus nur zulässige Hypothesen generiert, und zweitens, daß er keine Hypothese h erzeugt und auswertet, wenn er bereits eine kappbare allgemeinere Hypothese h' mit $h \subseteq h'$ entdeckt hat.

Die erste Bedingung ist offensichtlich erfüllt, da der Spezialisierungsoperator explizit keine unzulässigen Hypothesen generiert. Der Algorithmus führt also nie unzulässige Hypothesen ein und wertet nur zulässige Hypothesen aus. Die modifizierte Teilmengenbedingung überprüft für eine Hypothese h das Vorkommen aller allgemeineren zulässigen Hypothesen $h \setminus \{l\}$ in S und verwendet damit wie die ursprüngliche Teilmengenbedingung alle Subsumtionsbeziehungen zu bereits ausgewerteten Hypothesen zur Beschränkung des Suchraums.

4.2 Ein ILP–Sprachbias

Die Hypothesensprache, die der Algorithmus 4.2 durchsucht, ist bestimmt durch die Menge $Lits$ und die *excl*- und *req*-Bedingungen, die zu den Literalen in $Lits$ gehören. Die Literalmenge $Lits$ und die dazugehörigen *excl*- und *req*-Bedingungen müssen vom Anwender des Verfahrens geeignet vorgegeben werden. Der in diesem Abschnitt eingeführte ILP–Sprachbias soll es ermöglichen und erleichtern, mit diesen Sprachmitteln geeignete ILP–Hypothesensprachen zu spezifizieren. Zunächst ist die ILP–Hypothesensprache eine Sprache von Konjunktionen von Literalen. Anschließend wird die Anwendung des Ansatzes auf in Prolog oder SQL notierte Sprachen erläutert.

4.2.1 Bestandteile der Hypothesensprache

Verbinder und Merkmale. Um für die Suche in multi-relationalen Datenbanken Hypothesensprachen definieren zu können, die nur individuenverbundene Hypothesen enthalten, werden Verbinder eingeführt. Das folgende Beispiel verdeutlicht das Problem.

Beispiel 38 *Gegeben ist eine relationale Datenbank det-Phon mit den beiden Relationen* $syl(S, W, Acc)$ *und* $word(W, Tag)$*. Zwischen den beiden Relationen besteht eine Schlüssel–Fremdschlüssel–Beziehung über das Attribut* W*, den Identifikator (Primärschlüssel) der Relation* $word$*. In der Relation* syl *bezeichnet er das Wort, in dem eine Silbe enthalten ist. Da jede Silbe in genau einem Wort vorkommt, ist die Verbindung deterministisch.*

Gesucht sind interessante Teilmengen von Silben. Das Populationsliteral ist $pop = syl(S, W, Acc)$. *Die Individuenpopulation mit* S *als Individuenvariable ist*

$$Pop = \{S\theta | \textit{det-Phon} \models syl(S, W, Acc)\}.$$

Die Hypothesensprache soll unter anderem folgende Teilgruppenbeschreibung enthalten:

$$syl(S, W, Acc), word(W, Tag), Tag = noun.$$

Die Literalmenge $Lits$ *muß also die beiden Literale* $word(W, Tag)$ *und* $Tag = noun$ *enthalten. Wenn für diese Literale keine einschränkenden Bedingungen definiert werden, erlaubt die Literalmenge* $\{word(W, Tag), Tag = noun\}$ *folgende Teilgruppenbeschreibungen mit dem Populationsliteral* $syl(S, W, Acc)$ *zu generieren:*

$$\begin{aligned} h_1 &= syl(S, W, Acc), word(W, Tag) \\ h_2 &= syl(S, W, Acc), Tag = noun \\ h_3 &= syl(S, W, Acc), word(W, Tag), Tag = noun \end{aligned}$$

Davon sind aber die zwei Teilgruppenbeschreibungen h_1 *und* h_2 *unerwünscht: Weil jede Silbe in genau einem Wort vorkommt, beschreibt* h_1 *die gesamte Population, ist also keine echte Spezialisierung der Gesamtpopulation bezüglich* $\succ_e$. *Darum ist es überflüssig, die Teilgruppenbeschreibung* h_1 *zu erzeugen und auszuwerten.*

Die Teilgruppenbeschreibung h_2 *ist nicht individuenverbunden und deshalb ebenfalls keine echte Spezialisierung der Gesamtpopulation bezüglich* $\succ_e$.

Im Beispiel 38 erfüllen die Literale $Tag = noun$ und $word(W, Tag)$ in den Teilgruppenbeschreibungen unterschiedliche Funktionen. Das Literal $word(W, Tag)$ stellt die Verbindung der Relation $word$ mit dem Populationsliteral und den Individuenvariablen her, ohne eine echte Teilgruppe der Population zu beschreiben. Das Literal $Tag = noun$ spezialisiert die Population zu einer echten Teilgruppe, jedoch nur, wenn es über das Literal $word(W, Tag)$ individuenverbunden ist. Entsprechend der verschiedenen Funktionen, die Literale erfüllen können, werden zwei Arten von Literalen unterschieden und im Suchalgorithmus unterschiedlich behandelt.

Verbinder führen Datenbankrelationen ein und legen ihre Verbindungen zur Individuenpopulation fest. In der Terminologie der relationalen Datenbank heißt das, daß Verbinder Verbundoperationen (Joins) zwischen den Relationen definieren. Ihr Prädikat ist ein Datenbankprädikat, das heißt, der Name einer Relation, die in der Datenbank vorkommt. Verbinder definieren die Verbindungen durch gemeinsame Variablen an den entsprechenden Argumentpositionen, zum Beispiel an der Schlüssel- und Fremdschlüssel-Position der betreffenden Relationen.

Merkmale beschreiben Eigenschaften von Individuen, die eine Teilgruppe der Individuenpopulation definieren. Merkmale enthalten Variablen, die in Verbindern vorkommen. Merkmale haben typischerweise die Form $eigenschaft(V)$, $V = const$, $V \geq const$, $V_1 = V_2$ etc.

Zur Deklaration jedes Merkmals gehört eine als *FromListe* bezeichnete Liste der Verbinder, aus denen seine Variablen stammen. Im obigen Beispiel enthält die *FromList* des Literals $Tag = Noun$ das Literal $word(W, Tag)$. Eine entsprechende Liste muß auch in der Deklaration eines Verbinders l_v vorkommen, in der wiederum die Verbinder stehen, die die Verbindung von l_v zum Populationsliteral pop herstellen.

Komplexe Merkmale. Komplexe Merkmale sind Merkmale, deren Definition aus einer Konjunktion von Literalen besteht. Sie erlauben, mehrere elementare Spezialisierungsoperationen zu einem einzigen Spezialisierungsschritt zusammenzufassen.

4.2.2 Deklaration der Hypothesensprache

Verbinder und Merkmale. Durch die Unterscheidung von Verbindern und Merkmalen und die Einführung komplexer Merkmale werden die logischen Repräsentationen von Teilgruppenbeschreibungen als Literalkonjunktionen losgelöst von den Repräsentationen der Hypothesen in Form von Merkmalslisten, mit denen der Suchalgorithmus umgeht. Daher wird bei der Deklaration der Hypothesensprache zwischen einem Merkmal aus Sicht des Spezialisierungsoperators und der ihm entsprechenden logischen Repräsentation unterschieden. Merkmale und Verbinder bekommen einen Namen, um sie identifizieren zu können. Die logischen Literale, die das Merkmal ausmachen, heißen Definition des Merkmals.

Die Deklaration einer Hypothesensprache besteht aus einer Menge von Merkmalsdeklarationen $Lits$, auf der eine Reihenfolge definiert ist, und einer Menge von Verbinderdeklarationen $Links$.

Definition 32 *Die Deklaration $dec(l)$ eines Merkmals l hat folgende From:*

$$lit(l, [\textit{def}(\textit{DefList}), from(\textit{FromList}), excl(\textit{ExclList}), req(\textit{ReqList})]).$$

DefList ist eine Liste von Literalen. FromList ist eine Liste von Identifikatoren von Verbindern. ExclList ist eine Liste von Merkmalsidentifikatoren und ReqList ist eine Liste von Listen von Merkmalsidentifikatoren. Für ein Merkmal l ist def(l) die DefList von l und heißt die Definition von l, $from(l)$ ist die FromList von l, $excl(l)$ die ExclList und $req(l)$ die ReqList. Jede Merkmalsdeklaration muß DefList enthalten, die übrigen Deklarationsbestandteile sind optional. Der Identifikator eines Merkmals l wird auch als $name(l)$ referenziert.

Definition 33 *Die Deklaration $dec(v)$ eines Verbinders v hat folgende From:*

$$link(v, [rel(Relation), \textit{def}(\textit{DefList}), from(\textit{FromList})]).$$

$Relation$ ist ein Literal, dessen Prädikat ein Datenbankprädikat ist, DefList ist eine List von Literalen und FromList ist eine List von Identifikatoren von Verbindern.

Für einen Verbinder v ist $rel(v)$ die Relation von v, def(v) die DefList von v und $from(v)$ die FromList von v. In einer Verbinderdeklaration ist die $Relation$ obligatorisch. Der Identifikator eines Verbinders v wird auch als $name(v)$ referenziert.

Beispiel 39 *Gegeben sei eine Datenbank mit den beiden Relationen $syl(Syl, \_, Acc)$ und $phon(Phon, Syl, Type)$. Mögliche Deklarationen von Verbindern und Merkmalen sind:*

$$\begin{array}{ll} link(syl, & [rel(''syl(Syl, \_, Acc)'')]). \\ link(phon1, & [rel(''phon(Phon, Syl, Type)'')]), from([syl])]). \\ lit(short, & [\textit{def}([''Type = `short`'']), from([syl, phon1])]). \end{array}$$

4.2.3 Anpassung des Suchalgorithmus

Der Suchalgorithmus operiert auf Merkmalen und auf ihren *req*- und *excl*-Listen. Eine Hypothese aus Sicht des Suchalgorithmus ist eine Menge von Merkmalen, also $h \subseteq Lits$. Eine zulässige Hypothese ist damit eine Liste von Merkmalen, die den *req*- und *excl*-Einschränkungen ihrer Merkmale genügt. Eine elementare Spezialisierungsoperation besteht darin, ein Merkmal an eine Hypothese anzufügen. Eine Hypothese gehört zu der Suchebene, die der Anzahl ihrer Merkmale (nicht ihrer Literale) entspricht, und die Teilmengenbedingung wird nur für Merkmale überprüft. Komplexe Merkmale werden dabei behandelt wie einfache Merkmale: aus Sicht der Ebeneneinteilung des Suchraums und der Teilmengenbedingung entspricht das Anfügen eines komplexes Merkmals genau wie das

Tafel 4.3 Generator für die Logikrepräsentation lr einer Hypothese h. Die Argumente der Funktion $concat$ sind eine Liste von Zeichenketten L und eine einzelne Zeichenkette S. Sie konkateniert die Elemente von L und fügt dabei S zwischen je zwei aufeinanderfolgende Elemente von L ein.

Eingabe:	Deklarationen $Decs = Lits \cup Links$, Hypothese h
Ausgabe:	Logikrepräsentation $lr(h)$

1. lr_generate($Decs, h$): lr
2. $decs := \{dec(l) \in Decs | l \in h\} \cup \{dec(v) \in Decs | \exists d \in decs : \ v \in from(d)\}$;
3. $reldefs := \{rel(c) | c \in decs\} \cup \{\text{def}(c) | c \in decs\}$
4. $lr :=$ concat($reldefs$, ','$)$;
5. return lr;

Anfügen eines einfachen Merkmals einer elementaren (also kleinstmöglichen) Spezialisierungsoperation.

Die Repräsentation einer Hypothese als logische Formel setzt sich aus den Definitionen der in der Hypothese vorkommenden Merkmale und den Relationen und Definitionen der in den Merkmalsdeklarationen aufgeführten Verbinder zusammen. Die Verbinder sollen keine Spezialisierung einer Hypothese bewirken, sondern die logische Repräsentation einer Hypothese zu einer individuenverbundenen Teilgruppenbeschreibung ergänzen. Vollständige *From*-Listen für Merkmale und Verbinder vorausgesetzt, ergeben sich nur individuenverbundene Teilgruppenbeschreibungen.

Beispiel 40 *Gegeben seien die Deklarationen aus Beispiel 39. Die damit einzig mögliche Hypothese ist* $\{short\}$*. Ihre Repräsentation als logische Formel lautet:*

$$syl(Syl, Acc), phon(Phon, Syl, Type), Type = \text{`}short\text{`}.$$

4.2.4 Generierung der Logikrepräsentation einer Hypothese

Aus den Deklarationen der in einer Hypothese h aufgeführten Merkmale und Verbinder kann ein Formelgenerator die logische Repräsentation von h und die zu ihrer Auswertung benötigten Datenbankanfragen erzeugen. Einen Formelgenerator zur Generierung der Logikrepräsentation einer Hypothese zeigt Tafel 4.3. Die Eingabe des Formelgenerators ist eine Hypothese h in Form einer Menge von Merkmalen. Der Formelgenerator sammelt in einer Menge $decs(h)$ die Deklarationen aller Merkmale von h und rekursiv die Deklarationen aller Verbinder, die in der *FromList* einer Deklaration in $decs$ vorkommen. Die Logikrepäsentation der Hypothese ist die Konkatenation aller Relationen und Definitions-Teile, durch Komma getrennt, die in den Deklarationen in $decs(h)$ vorkommen.

4.2.5 Prolog als Anfragesprache

Aus der Abstraktion von Merkmalen von ihrer Definition beziehungsweise der Verbinder von den ihnen zugrundeliegenden Datenbankrelationen, die in Abschnitt 4.2.2 eingeführt wurde, ergibt sich die konzeptuelle Trennung zwischen der Repräsentation einer Hypothese aus Sicht des Suchalgorithmus und Spezialisierungsoperators (nämlich als Liste von Merkmalen) und der Repräsentation der Hypothese als Teilgruppenbeschreibung mit dem Vokabular und Format der Datenbank. Der Suchalgorithmus läßt sich deshalb auf relationale Daten in unterschiedlichen Formaten anwenden. Die bisherige Erörterung hat sich auf Hypothesensprachen in der prädikatenlogischen Sprache der Logikprogammierung konzentriert. Für die Speicherung der Daten und die Auswertung der Hypothesen in diesem Rahmen eignet sich die Programmiersprache Prolog. Die Deklarationen von Verbindern und Merkmalen in Beispiel 39 erlauben die Auswertung durch Prolog-Anfragen.

Eine weitere Konsequenz der Abstraktion der Merkmale und Verbinder von ihren Definitionen beziehungsweise den zugrundeliegenden Datenbankrelationen ist, daß die Hypothesensprache direkt in Prolog deklariert werden kann. Das heißt, daß die Definitionen von Merkmalen und Verbindern nicht mehr auf Literale beschränkt sind, sondern aus Fragmenten von Prologcode bestehen können. Dabei lassen sich beliebige Sprachmittel von Prolog einsetzen, beispielsweise Disjunktionen oder eingebaute Prädikate, solange die Codefragmente zusammengesetzt syntaktisch korrekte und im Sinne von Abschnitt 2.2.2 erlaubte Anfragen ergeben, die die Prolog-Beweisprozedur vollständig und korrekt auswerten kann. Die Fragmente von Prologcode werden buchstabengetreu deklariert.

Beispiel 41 *Das folgende Merkmal ist für die Experimente in Kapitel 6 deklariert. Es führt einfache Berechnungen aus.*

$link(krk,$ $[rel(''krk(Id, C, \mathit{WKC}, \mathit{WKR}, \mathit{BKC}, \ldots)'')]).$
$lit(adj,$ $[\mathit{def}([''N\ is\ \mathit{WKC} - \mathit{BKC}, (N = 1; N = -1; N = 0)''$
$from([krk])]).$

Deklaration von Population und Zielgruppe

Prolog-Anfragen zur Auswertung einer Hypothese lassen sich ähnlich wie die Logikrepräsentation automatisch aus den Deklarationen der Hypothesensprache generieren. Dazu werden geeignete Deklarationen von Population und Zielgruppe benötigt.

Definition 34 *Die Deklarationen von Individuenpopulation und Zielgruppe haben folgendes Format:*

$link(pname,$ $[rel(RLit)]).$
$count(vname,$ $[att(Vars),$ $from([pname])]).$
$class(cname,$ $[classes(\mathit{ClassList}),$ $from([\mathit{FromList}]),$
$targetclass(tname)]).$

Die $ClassList$ hat folgende Form:

$[name1 - def1, name2 - def2]$

Das Populationsliteral wird deklariert als Verbinder $pname$. Es erscheint in der *FromList* des $count$-Ausdrucks. Die att-Komponente des $count$-Ausdrucks deklariert die Individuenvariablen $Vars$. Der $class$-Ausdruck dient zur Deklaration der Zielgruppe. In der $classes$-Komponente werden zwei Teilgruppen der Population deklariert, die sich zur gesamten Population ergänzen. In der *ClassList* sind $name1$ und $name2$ Identifikatoren für die beiden Teilgruppen, $def1$ ist die Definition der Klasse $name1$ und wird mit *def*$(name1)$ bezeichnet, *def*2 ist die Definition der Klasse $name2$ und wird entsprechend mit *def*$(name2)$ bezeichnet. Die Definitionen *def*1 und *def*2 entsprechen der Definition eines Merkmals. Eine dieser beider Teilgruppen wird im $targetclass$-Ausdruck als Zielgruppe spezifiziert durch die Angabe ihres Identifikators in der Form $targetclass(name1)$ beziehungsweise $targetclass(name2)$.

Beispiel 42 *Die folgenden Deklarationen definieren Population und Zielgruppe für die Generierung von Prolog-Anfragen.*

$link(syl,$ $[rel(''syl(Syl, \_, Acc)'')]).$
$count(sylid,$ $[att(''Syl''),$ $from([syl])]).$
$class(accent,$ $targetclass(acc),$ $from([syl]),$
$[classes($ $[acc -'' Acc = 'acc''',$
$no\_acc -'' Acc = '0'''])]).$

Der att-Teil des $count$–Ausdruck gibt Syl als Individuenvariable an. Der $classes$-Teil des $class$-Ausdrucks gibt an, daß in dieser Anwendung zwei Klassen namens acc und no_acc unterschieden werden. Die Ausdrücke hinter dem Bindestrich, also $''Acc = 'acc''$ beziehungsweise $''Acc = '0''$ sind die Definition der jeweiligen Klasse. Die Klasse acc besteht aus den Instanzen der Individuenvariable Syl, für die Acc den Wert $'acc'$ annimmt. Die Instanzen der Individuenvariable Syl mit Acc gleich $'0'$ gehören zur Klasse no_acc. Der $targetclass$-Teil des $class$-Ausdrucks weist acc als Zielgruppe aus.

Bemerkung. Zielgruppe und Population könnten auch anders definiert werden; insbesondere könnte man auf die explizite Definition des Komplements der Zielgruppe verzichten. Diese Art der Deklaration wurde hier so gewählt, weil sie kompatibel ist zu den unten beschriebenen, entsprechenden Deklarationen für einen SQL–Suchraum. Die explizite Definition des Zielgruppen-Komplements bietet außerdem den Vorteil, daß in einer Anwendung das Zielgruppen-Komplement einfach durch Anpasssung des $targetclass$-Ausdrucks zur Zielgruppe erklärt und dadurch der Fokus der Suche gewechselt werden kann.

Generierung der Anfragen

Zur Auswertung einer Hypothese h erzeugt der Prolog-Anfragegenerator zwei Anfragen, die die Schnittmenge zwischen den von h abgedeckten Individuen und den Partitionen $name1$ beziehungsweise $name2$ der Population berechnen. Die Anfragen haben die Form

$$findall(Vars, (pr(\{h \cup name1\})), I1).$$
$$findall(Vars, (pr(\{h \cup name2\})), I2).$$

Dabei sind $Vars$ die im $count$-Ausdruck deklarierten Individuenvariablen und $I1$ beziehungsweise $I2$ Mengen der jeweils abgedeckten Individuen, also Mengen von Instanzen von $Vars$. $I1$ und $I2$ sind die Instanzen von Vars, also die Teilgruppen der Population, die von der Hypothese h abgedeckt sind und zur Klasse $name1$ beziehungsweise $name2$ gehören.

Die Prologanfragen $pr(\{h \cup name1\})$ und $pr(\{h \cup name2\})$ werden durch einen Anfragegenerator erzeugt, der den Formelgenerator $lr\_generate$ in Tafel 4.3 verwendet, und zwar ist $pr(\{h \cup name\}) = opt(lr(\{h \cup name\})$. Der einfache Anfrageoptimierer opt ordnet $lr(\{h \cup name\})$ nach folgenden Prinzipien um:

- Literale der Form $Var = const$, die Variablen binden, kommen an den Anfang der Anfrage,
- darauf folgen die Relationen der Verbinder, an erster Stelle das Populationsliteral,
- danach die Ausdrücke, die gebundene Variablen erfordern: das sind Literale, in denen eine der Zeichenketten "=:=", "\ ==", "\+", "<","=<", "not", ">=", "is", ";" vorkommt.

Beispiel 43 *Gegeben seien die Deklarationen aus Beispiel 39. Zur Auswertung der aus einem Merkmal bestehenden Hypothese $h = \{short\}$ werden folgende Prolog-Anfragen erzeugt.*

$$findall(Vars, (pr(h \cup acc\})), H1) \quad \hat{=}$$
$$findall(Vars, (pr(\{short, acc\})), H1) \quad \hat{=}$$
$$findall(Syl, (Acc = 'acc', Type = 'short', syl(Syl, Acc), phon(Phon, Syl, Type)), H1).$$

$$
\begin{array}{ll}
findall(Vars, (pr(h \cup \{no\_acc\})), H2) & \hat{=} \\
findall(Vars, (pr(\{short, no\_acc\})), H2) & \hat{=} \\
findall(Syl, (Acc =' 0', Type = `short`, syl(Syl, Acc), & \\
\qquad phon(Phon, Syl, Type)), H1). &
\end{array}
$$

4.2.6 SQL als Anfragesprache

Häufig werden mehrrelationale Datenbanken mit Datenbankmanagement–Systemen verwaltet, die SQL als Anfragesprache bieten. Auch SQL–Anfragen zur Auswertung von Hypothesen können aus entsprechenden Deklarationen automatisch erzeugt werden.

Deklaration von Verbindern und Merkmalen

Um SQL–Anfragen zu erzeugen, werden anstelle der Prolog-Codestücke SQL–Codestücke deklariert. Das Format einer Verbinderdeklaration für SQL ist weitgehend identisch zur Deklaration eines Verbinders für die Auswertung mit Prolog. Der Name des Verbinders wird in der SQL–Anfrage als ALIAS für die Datenbankrelation verwendet. Im *rel*-Teil der Deklaration wird die Datenbankrelation genannt; im def-Teil der Deklaration wird der Verbund (Join) zu einer anderen Relation festgelegt, die im $from$-Teil der Deklaration aufgeführt wird. Im def-Teil einer Merkmalsdeklaration stehen SQL–Ausdrücke, die, zusammen mit den def-Teilen der nötigen Verbinder, den *where*-Teil der daraus erzeugten SQL–Anfrage ausmachen.

Beispiel 44 *(SQL–Version zu Beispiel 39) Gegeben sei eine Datenbank mit den Relationen* $syl(syl, acc)$ *und* $phon(phonid, syl, type)$.

$$
\begin{array}{ll}
link(syl, & [rel(''syl'')]). \\
link(phon1, & [rel(''phon''), \\
 & from([syl]), \\
 & def([''phon1.syl = syl.syl'']). \\
lit(short, & [def([''phon1.type = `short`'']), \\
 & from([syl, phon1])]).
\end{array}
$$

Deklaration von Population und Zielgruppe

Für die Auswertung einer Hypothese h wird eine SQL–Anfrage gebildet, die die Anzahl $|H \cap T|$ der zur Zielgruppe gehörenden Individuen aus der Abdeckung von h sowie die Anzahl $|H \cap \overline{T}|$ der nicht zur Zielgruppe gehörenden Individuen aus der Abdeckung von h berechnet. Die Anfrage liefert eine 2×2–Tabelle wie folgt:

$name1$	$\|H \cap T\|$
$name2$	$\|H \cap \overline{T}\|$

Dabei entsprechen $name1$ und $name2$ den Klassenbezeichnern aus der $class$-Deklaration für Prolog-Anfragen in Abschnitt 4.2.5. Hier sind $name1$ und $name2$ die beiden Werte eines binären Attributs $ClassAtt$, das die Individuenpopulation in zwei Klassen partitioniert, von denen eine mit einer $targetclass$-Deklaration als Zielgruppe bestimmt wird. Statt der $classes$-Komponente enthält der $class$-Ausdruck zur Deklaration der Zielgruppe für die SQL–Version eine att-Komponente, die das Attribut $ClassAtt$ deklariert. Diese Konstruktion ermöglicht, beide zur Auswertung einer Hypothese benötigten Größen $|H \cap T|$ und $|H \cap \overline{T}|$ in einer Anfrage zu berechnen. Die Deklaration von Population und Zielgruppe erfolgt analog zur Prologversion. Anstelle der Individuenvariablen wird hier ein Attribut $CountAtt$ deklariert, dessen Werte die Individuenpopulation bilden.

Definition 35 *Die Deklarationen von Individuenpopulation und Zielgruppe haben folgendes Format:*

$$\begin{array}{lll} count(vname, & [att(CountAtt), & from([pname])]). \\ class(cname, & [att(ClassAtt), & from([FromList]), \\ & targetclass(tname)]). & \end{array}$$

Beispiel 45 *Die folgenden Deklarationen definieren Population und Zielgruppe für die Generierung von SQL–Anfragen. In diesem Beispiel wird angenommen, daß das Attribut $syllable.accent$, von dem die Zielguppenzugehörigkeit der Individuen abhängt, nicht binär ist, sondern erst mittels einer CASE-Anweisung auf ein binäres Attribut abgebildet wird.*

$$\begin{array}{lll} count(sylid, & [att(''syl.syl''), & from([syl])]). \\ class(accent, & [att(''\textit{CASE WHEN } syl.accent <>' 0' & \\ & \quad \textit{THEN } 'acc' \textit{ ELSE } 'no\_acc' \textit{ END}''), & \\ & [targetclass(acc), & from([syl])]). \end{array}$$

Der att-Teil des $count$–Ausdrucks bestimmt $syl.syl$ als das Attribut, dessen Werte die Individuenpopulation bilden. Der att-Ausdruck der $class$-Deklaration definiert ein binäres Attribut mit den Werten acc und no_acc, das den Wert acc aufweist, wenn das Attribut $syl.acc$ einen Wert ungleich 0 hat, und sonst den Wert no_acc. Der $targetclass$-Teil bestimmt die Individuen mit dem Wert acc zur Zielgruppe.

Generierung der Anfragen

Für die SQL–Auswertung einer Hypothese wird eine Anfrage folgender Form generiert:

$$\begin{array}{ll} \text{SELECT} & cname, \text{ COUNT (DISTINCT } vname) \\ \text{FROM} & (\text{SELECT } Atts \text{ FROM } From \text{ WHERE Where}) \\ & \text{AS EX GROUP BY } cname; \end{array}$$

Die Bezeichner $cname$ und $vname$ sind direkt aus den $class$- und $count$-Deklarationen übernommen. Prozeduren zur Erzeugung der Anfrage-Komponenten $From$ und $Where$ sind in Tafel 4.4 gezeigt. Der $Atts$-Komponente wird wie folgt aus Bestandteilen der entsprechenden Deklarationen zusammengesetzt.

$$Atts \;=\; \text{ClassAtt AS } cname, \quad \text{CountAtt AS } vname$$

Beispiel 46 *Mit den Deklarationen aus den Beispielen 44 und 45 ergibt sich folgende Anfrage zur Auswertung der Hypohese $\{short\}$:*

$$\begin{array}{llll} \textit{SELECT} & accent, & \textit{COUNT (DISTINCT } sylid) & \\ \textit{FROM} & (\textit{SELECT} & \textit{CASE WHEN } syl.accent <>' 0' & \\ & & \textit{THEN } 'acc' \textit{ ELSE } 'no\_acc' \textit{ END} & \textit{AS } accent, \\ & & syl.syl & \textit{AS } sylid \\ & \textit{FROM} & syl\ syl, phon\ phon1 & \\ & \textit{WHERE} & phon1.syl = syl.syl & \\ & \textit{AND} & phon1.type = \text{`}short\text{`}) & \textit{AS EX} \\ \textit{GROUP BY} & accent; & & \end{array}$$

Tafel 4.4 Generierung des $Where$-Teils und des $From$-Teils für die SQL–Anfrage zur Auswertung einer Hypothese h.

Eingabe: Deklarationen $Decs = Lits \cup Links$, Hypothese h
Ausgabe: Where(h), From(h)

1. sr_from($Decs, h$): $From$
2. $decs := \{dec(l) \in Decs | l \in h\} \cup \{dec(v) \in Decs | \exists d \in decs : \ v \in from(d)\}$;
3. $relfroms :=$ {concat($\{rel(v), name(v)\}$,$''$ $''$)$|v \in decs$};
4. $From :=$ concat($relfroms$,$''$,$''$);

4. sr_where($Decs, h$): $Where$
5. $reldefs := \{rel(c)|c \in decs\} \cup \{def(c)|c \in decs\}$
6. Where := concat($reldefs$, $''$ AND $''$};
7. return $Where$;

4.2.7 Anmerkungen

Um den aus dem Apriori-Algorithmus abgeleiteten Grundalgorithmus zur ebenenweisen Suche im Suchraum $\left(2^{Lits}, \subseteq\right)$ auf komplexere Suchräume zu übertragen, wurde er erweitert um *excl*- und *req*-Bedingungen, um die Unterscheidung zwischen Verbindern und Merkmalen und um komplexe Merkmale. Die *excl*- und *req*-Bedingungen sind sehr allgemeine Erweiterungen des Suchalgorithmus, die auch in nicht-relationalen Hypothesenräumen nützlich sein können, um Suchräume für spezifische Anwendungen zu beschreiben und so die Suche auf die Anwendungsziele ausrichten zu können. Einen verwandten Ansatz beschreibt [SVA97].

Die Unterscheidung zwischen Verbindern und Merkmalen und die Erweiterung um komplexe Literale sind auf die Erfordernisse von Suchräumen ausgerichtet, die nach der θ-Subsumtion geordnet sind. Die Verbinder und Merkmale erlauben die Deklaration von Hypothesenräumen, die ausschließlich (individuen-)verbundene Hypothesen enthalten. Der Effekt von Verbindern ist nicht zu erreichen, indem man anstelle der *from*-Definitionen *req*-Definitionen vorgibt, durch die ein Merkmal die Literale erfordert, aus denen seine Variablen stammen. Diese Literale müssen dazu nämlich als Merkmale definiert sein. Das Anfügen eines dieser Literale an eine Hypothese stellt dann eine elementare Spezialisierungsoperation dar, die eine neue Hypothese erzeugt. Die so erzeugten Hypothesen können nicht einfach unterdrückt werden, da sonst die Teilmengenbedingung inkorrekt wird.

Die Einführung von Verbindern bietet die Möglichkeit, mehrere Literale (nämlich die Merkmale und die dazugehörigen Verbinder) in einem einzigen Spezialisierungsschritt in eine Hypothese aufzunehmen, ohne daß das Anfügen eines dieser Literale als eine elementare Spezialisierung behandelt wird. Diese Funktion erfüllen auch die komplexen Merkmale. Komplexe Merkmale sind das allgemeinere Konstrukt, denn sie können die Funktion von Verbindern erfüllen, indem die Verbinder eines Merkmals nicht in dessen *from*-List aufgeführt werden, sondern indem die verbindenden Literale mit dem spezialisierenden Literal des Merkmals zu einem komplexen Merkmal zusammengesetzt werden. Beispielsweise könnte statt eines Merkmals $Type = short$ mit dem Verbinder $phon(P, S, Type)$ das komplexe Merkmal $(phon(P, S, Type), Type = short)$ deklariert werden. Die Unterscheidung zwischen Verbindern und Merkmalen wurde hier aus folgenden Gründen dennoch eingeführt.

Erstens ergibt sie knappere Deklarationen, denn oft werden dieselben Verbinder von mehreren Merkmalen benötigt und müßten bei der Lösung mit komplexen Merkmalen entsprechend mehrfach deklariert werden. Der Suchalgorithmus kann zwar mehrfache Vorkommen derselben Literale aus Hypothesen herausfiltern, die redundante Deklaration ist jedoch unübersichtlicher und fehlerträchtiger. Zweitens entspricht die Unterscheidung zwi-

schen Verbindern und Merkmalen der Syntax von SQL, so daß der Ansatz auch die Deklaration von SQL–Hypothesensprachen erlaubt. Ein dritter Vorteil ist, daß die Deklarationen mit wenig Aufwand an ein verändertes Format der Datenbank angepaßt werden können, da nur die Definitionen der Verbinder vom Datenbankschema abhängen, wenn dieses zum Beispiel zur Effizienzsteigerung geändert wird.

4.3 Anwendungsbeispiele

Im Vergleich zur θ-Subsumtion ist die Teilmengenbeziehung zwischen Hypothesen einfach zu überprüfen. Die Teilmengenbeziehung ist jedoch schwächer als die θ-Subsumtion, sie zeigt in den komplexen Hypothesenräumen der ILP weniger Subsumtionsbeziehungen an als die θ-Subsumtion. In diesem Abschnitt wird anhand von Anwendungsbeispielen demonstriert, daß der hier entwickelte Sprachbias und Spezialisierungsoperator erlauben, eine Vielzahl von Hypothesenräumen, die bei der Teilgruppenanalyse typischerweise auftreten können, redundanzfrei zu erzeugen und mittels der Teilmengenbedingung zu beschränken.

Wie in Abschnitt 2.3.3 dargelegt wurde, reicht die Aufteilung der Daten auf mehrere Relationen nicht aus, um die Aussagefähigkeit der Datenrepräsentation gegenüber einer Attribut-Wert-Darstellung wesentlich zu erhöhen. Erst wenn nicht-deterministische oder rekursive Verbindungen zwischen den Relationen bestehen, steigt die Ausdruckskraft gegenüber aussagenlogischen Repräsentationen wesentlich an. Die folgenden Anwendungsbeispiele behandeln daher verschiedene Möglichkeiten, Suchräume für Datenbanken mit einer nichtdeterministischen oder rekursiven Verbindung zu definieren.

Redundanzfreiheit bedeutet hier, daß der Hypothesenraum jeweils nur einen Vertreter einer Klasse zueinander subsumtionsäquivalenter Hypothesen enthalten soll. Dieser ausgewählte Vertreter der Äquivalenzklasse kann auch nicht-reduziert sein, daß heißt, unnötige Literale enthalten. Nicht-reduzierte Hypothesen bringen keine Redundanz in den Hypothesenraum, solange keine reduzierten oder andere nichtreduzierte Vertreter derselben Äquivalenzklasse im Hypothesenraum enthalten sind.

4.3.1 Beispiel-Datenbank

In allen Anwendungsbeispielen außer den letzten beiden wird die relationale Datenbank *Phon* durchsucht. Sie ist ausführlich beschrieben im Abschnitt 6.1.2. Um die Übersichtlichkeit zu erhöhen, werden die Relationenschemata für die Anwendungsbeispiele vereinfacht. Die Relationenschemata sind $syl(\underline{Syl}, Acc)$ und $phon(Syl, \underline{P}, Pv, Pn, Type)$. Die Relation syl beschreibt Silben, die aus Aufnahmen von sprachlichen Äußerungen im Rahmen phonetischer Untersuchungen gewonnen wurden. Das Attribut S ist der Primärschlüssel der Relation syl. Das binäre Attribut Acc hat den Wert acc, wenn die Silbe einen Wortakzent trägt, sonst den Wert 0. Das Attribut P ist der Primärschlüssel der Relation $Phon$. Das Attribut $Type$ beschreibt den Typ des Phonems, zum Beispiel Diphtong, kurzer Vokal, Frikativ oder Liquid. Zwischen den beiden Relationen besteht eine Schlüssel-Fremdschlüssel-Beziehung über das Attribut Syl, das in der Relation $phon$ die Silbe bezeichnet, in der ein Phonem enthalten ist. Da eine Silbe aus mehreren Phonemen bestehen kann, ist die Verbindung nicht-deterministisch. Die Attribute Pv und Pn geben die Phoneme an, die dem Phonem $Phon$ in der sprachlichen Äußerung vorausgehen beziehungsweise nachfolgen. Sie stellen rekursive Verbindungen her.

Untersuchungsgegenstand der Beispielanwendungen sind interessante Teilmengen von Silben, die Zielgruppe sind die akzenttragenden Silben. Individuenpopulation, Individuenvariable und Zielgruppe sind deklariert wie in Beispiel 39.

Für jedes Anwendungsbeispiel werden die Deklaration des Suchraums und der resultierende Suchraum oder ein Ausschnitt davon gezeigt, außerdem das Resultat eines Suchlaufs, der die zehn Hypothesen mit der größten Verteilungsungewöhnlichkeit im Suchraum bestimmt.

Tafel 4.5 Deklaration des Suchraums für die Suche nach einfachen Kombinationen.

$link(syl,$	$[rel(''syl(Syl, Acc)'')]).$	
$link(t1,$	$[rel(''phon(Syl, \_, \_, \_, T1)''),$	$from([syl])]).$
$link(t2,$	$[rel(''phon(Syl, \_, \_, \_, T2)''),$	$from([syl])]).$
$link(t3,$	$[rel(''phon(Syl, \_, \_, \_, T3)''),$	$from([syl])]).$
$link(t4,$	$[rel(''phon(Syl, \_, \_, \_, T4)''),$	$from([syl])]).$
$lit(diph,$	$[def([''T1 = diphtong'']),$	$from([t1])]).$
$lit(frik,$	$[def([''T2 = frikativ'']),$	$from([t2])]).$
$lit(short,$	$[def([''T3 = short'']),$	$from([t3])]).$
$lit(liq,$	$[def([''T4 = liquid'']),$	$from([t4])]).$

Zwischen zwei Relationen können mehrere verschiedene Verbindungsmöglichkeiten bestehen. Die Phonetik-Datenbank legt solche Verbindungen jedoch nicht nahe. Zur Ergänzung werden daher am Ende dieses Abschnitts weniger ausführlich und unabhängig von der Phonetik-Datenbank zwei Möglichkeiten gezeigt, Suchräume für eine Datenbank zu definieren, in der zwei alternative Verbindungsmöglichkeiten zwischen zwei Datenbank-Relationen bestehen.

4.3.2 Einfache Kombinationen

Die Hypothesen der Hypothesensprache dieses fiktiven Anwendungsbeispiels beschreiben Teilgruppen von Silben, die bestimmte Kombinationen von Phonemtypen enthalten, beispielsweise einen Diphtong und einen Frikativ. Die Reihenfolge, in der die Phonemtypen in den Silben aufeinanderfolgen, ist dabei nicht von Bedeutung. Der Kürze wegen werden nur vier Phonemtypen untersucht, nämlich Diphtonge, Frikative, kurze Vokale (short) und Gleitlaute (liquid). Ein Beispiel für eine Teilgruppenbeschreibung in dieser Hypothesensprache ist die folgende, die Silben mit einem Frikativ und einem Liquid beschreibt:

$$T2 = frikativ, T4 = liquid, syl(Syl, Acc), phon(Syl, \_, \_, \_, T4),$$
$$phon(Syl, \_, \_, \_, T2).$$

Tabelle 4.5 zeigt die Deklaration der Hypothesensprache für dieses Anwendungsbeispiel. Um bis zu vier Phoneme einer Silbe berücksichtigen zu können, sind entsprechend viele link-Ausdrücke erforderlich, die über die Variable Syl Verbindungen zwischen der Relation Phonem und dem Populationsliteral definieren. Zu jedem Verbinder gehört ein Merkmal, das die Variable Ti mit einem konstanten Wert belegt. Die Merkmale definieren die Silbeneigenschaften „enthält einen Diphtong“, „enthält einen Frikativ“, „enthält einen kurzen Vokal“ und „enthält einen Gleitlaut“. Diese Merkmale lassen sich beliebig miteinander kombinieren. Insgesamt sind daraus ohne die leere allgemeinste Hypothese $2^4 - 1 = 15$ Hypothesen bildbar. Den resultierenden Suchraum zeigt Tabelle 4.6. Darin ist jede Hypothese als Liste von Merkmalsnamen und als Logikrepräsentation wiedergegeben, die der Anfragegenerator lr_gen erzeugt. Die zehn besten Hypothesen mit der höchsten Bewertung, die bei dieser Beispielanwendung gefunden werden, zeigt Tafel 4.7.

4.3.3 Kombinationen mit Mehrfachvorkommen

Die Hypothesensprache dieses Anwendungsbeispiels unterscheidet sich von der Sprache des vorhergehenden Beispiels darin, daß sie auch Beschreibungen von Silben enthält, in

Tafel 4.6 Der Suchraum für die Suche nach einfachen Kombinationen.

Hypothese	Logikrepräsentation
1. Suchebene	
$diph.$	$T1 = diphtong, syl(Syl, Acc), phon(Syl, \_, \_, \_, T1).$
$frik.$	$T2 = frikativ, syl(Syl, Acc), phon(Syl, \_, \_, \_, T2).$
$short.$	$T3 = short, syl(Syl, Acc), phon(Syl, \_, \_, \_, T3).$
$liq.$	$T4 = liquid, syl(Syl, Acc), phon(Syl, \_, \_, \_, T4).$
2. Suchebene	
$diph, frik.$	$T1 = diphtong, T2 = frikativ,\ syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T2),\ phon(Syl, \_, \_, \_, T1).$
$diph, short.$	$T1 = diphtong, T3 = short,\ syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T3),\ phon(Syl, \_, \_, \_, T1).$
$diph, liq.$	$T1 = diphtong, T4 = liquid,\ syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T4),\ phon(Syl, \_, \_, \_, T1).$
$frik, short.$	$T2 = frikativ, T3 = short,\ syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T3),\ phon(Syl, \_, \_, \_, T2).$
$frik, liq.$	$T2 = frikativ, T4 = liquid,\ syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T4),\ phon(Syl, \_, \_, \_, T2).$
$short, liq.$	$T3 = short, T4 = liquid,\ syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T4),\ phon(Syl, \_, \_, \_, T3).$
3. Suchebene	
$diph, frik, short.$	$T1 = diphtong, T2 = frikativ, T3 = short, syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T3),\ phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1).$
$diph, frik, liq.$	$T1 = diphtong, T2 = frikativ, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T4),\ phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1).$
$diph, short, liq.$	$T1 = diphtong, T3 = short, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T4),\ phon(Syl, \_, \_, \_, T3), phon(Syl, \_, \_, \_, T1).$
$frik, short, liq.$	$T2 = frikativ, T3 = short, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, \_, \_, \_, T4),\ phon(Syl, \_, \_, \_, T3), phon(Syl, \_, \_, \_, T2).$
4. Suchebene	
$diph, frik, short, liq.$	$T1 = diphtong, T2 = frikativ, T3 = short,$ $T4 = liquid,\ syl(Syl, Acc),\ phon(Syl, \_, \_, \_, T4),$ $phon(Syl, \_, \_, \_, T3), phon(Syl, \_, \_, \_, T2),\ phon(Syl, \_, \_, \_, T1).$

denen zwei Phoneme des selben Typs vorkommen, wie zum Beispiel die folgende Beschreibung der Silben mit zwei Frikativen:

$$T2 = frikativ, T5 = frikativ, syl(Syl, Acc), phon(Syl, P5, \_, T5),$$
$$phon(Syl, P2, \_, T2), P5 \neq P2.$$

Das Merkmal, das die Silbeneigenschaft „enthält einen zweiten Frikativ“ beschreibt,

Tafel 4.7 Resultat der Suche nach einfachen Kombinationen.

Pos.	Hypothese	Bewertung	Pos.	Hypothese	Bewertung
1.	$short.$	0.022170	6.	$liq.$	0.012661
2.	$frik.$	0.021074	7.	$diph.$	0.012058
3.	$frik, short.$	0.016221	8.	$frik, short, liq.$	0.009974
4.	$frik, liq.$	0.015443	9.	$diph, frik.$	0.005695
5.	$short, liq.$	0.014779	10.	$diph, liq.$	0.002518

Tafel 4.8 Deklaration des Suchraums für die Suche nach Kombinationen mit Mehrfachvorkommen.

$link(syl,$ $[rel(''syl(Syl, Acc)'')]).$

$link(t1,$ $[rel(''phon(Syl, P1, \_, \_, T1)''),$ $from([syl])]).$

$link(t2,$ $[rel(''phon(Syl, P2, \_, \_, T2)''),$ $from([syl])]).$

$link(t3,$ $[rel(''phon(Syl, P3, \_, \_, T3)''),$ $from([syl])]).$

$link(t4,$ $[rel(''phon(Syl, P4, \_, \_, T4)''),$ $from([syl])]).$

$link(t5,$ $[rel(''phon(Syl, P5, \_, \_, T5)''),$ $from([syl])]).$

$link(t6,$ $[rel(''phon(Syl, P6, \_, \_, T6)''),$ $from([syl])]).$

$lit(diph,$ $[def([''T1 = diphtong'']),$ $from([t1]),$
$excludes([short])]).$

$lit(frik,$ $[def([''T2 = frikativ'']),$ $from([t2])]).$

$lit(short,$ $[def([''T3 = short'']),$ $from([t3]),$
$excludes([diph])]).$

$lit(liq,$ $[def([''T4 = liquid'']),$ $from([t4])]).$

$lit(frik2,$ $[def([''T5 = frikativ, P5 \neq P2'']),$ $from([t2, t5]),$
$requires([frik])]).$

$lit(liq2,$ $[def([''T6 = liquid, P6 \neq P4'']),$ $from([t4, t6]),$
$requires([liq])]).$

wird deklariert wie folgt:

$lit(frik2,$ $[def([''T5 = frikativ'','' P5 \neq P2'']), from([t2, t5]),$
$requires([frik])]).$

Die *req*-Bedingung von frik2 bewirkt, daß ein zweiter Frikativ nur in solchen Silben gesucht wird, die bereits einen ersten Frikativ enthalten. Die Bedingung $P5 \neq P2$ in der Definition von frik2 verhindert, daß die Variable $P5$ mit derselben Phonem-Id wie $P2$ instanziiert und dadurch jeder Frikativ zweimal gezählt wird. Außer doppelten Vorkommen von Frikativen in Silben werden in dieser Beispielanwendung auch doppelte Vorkommen von Liquiden abgefragt. Doppelte Vorkommen von Diphtongen und kurzen Vokalen sind nicht möglich, weil jede Silbe nur einen Vokal enthält. Als Verbesserung gegenüber der vorhergehenden Beispielanwendung werden hier deshalb außerdem kurze Vokale und Diphtonge als sich gegenseitig ausschließend deklariert. Die vollständigen Deklarationen sind in Tafel 4.8 gezeigt. Der Suchraum, den diese Deklarationen erzeugen, enthält 26 Hypothesen. Ein Ausschnitt davon ist in Tafel 4.9 wiedergegeben.

Tafel 4.9 Der Suchraum für die Suche nach Kombinationen mit Mehrfachvorkommen.

Hypothese	Logikrepräsentation
1. Suchebene	
$diph.$	$T1 = diphtong,\ syl(Syl, Acc), phon(Syl, P1, \_, \_, T1).$
$frik.$	$T2 = frikativ,\ syl(Syl, Acc), phon(Syl, P2, \_, \_, T2).$
$short$	$T3 = short,\ syl(Syl, Acc), phon(Syl, P3, \_, \_, T3).$
$liq.$	$T4 = liquid,\ syl(Syl, Acc), phon(Syl, P4, \_, \_, T4).$
2. Suchebene	
$diph, frik.$	$T1 = diphtong, T2 = frikativ, syl(Syl, Acc),$ $phon(Syl, P2, \_, \_, T2), phon(Syl, P1, \_, \_, T1).$
$diph, liq.$	$T1 = diphtong, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, P4, \_, \_, T4), phon(Syl, P1, \_, \_, T1).$
$frik, short.$	$T2 = frikativ, T3 = short, syl(Syl, Acc),$ $phon(Syl, P3, \_, \_, T3), phon(Syl, P2, \_, \_, T2).$
$frik, liq.$	$T2 = frikativ, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, P4, \_, \_, T4), phon(Syl, P2, \_, \_, T2).$
$frik, frik2.$	$T2 = frikativ, syl(Syl, Acc), phon(Syl, P5, \_, \_, T5),$ $phon(Syl, P2, \_, \_, T2), T5 = frikativ, P5 \neq P2.$
$short, liq.$	$T3 = short, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, P4, \_, \_, T4), phon(Syl, P3, \_, \_, T3).$
$liq, liq2.$	$T4 = liquid, syl(Syl, Acc), phon(Syl, P6, \_, \_, T6),$ $phon(Syl, P4, \_, \_, T4), T6 = liquid, P6 \neq P4.$
3. Suchebene	
$diph, frik, liq.$	$T1 = diphtong, T2 = frikativ, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, P4, \_, \_, T4), phon(Syl, P2, \_, \_, T2),$ $phon(Syl, P1, \_, \_, T1).$
$diph, frik, frik2.$	$T1 = diphtong, T2 = frikativ, syl(Syl, Acc),$ $phon(Syl, P5, \_, \_, T5), phon(Syl, P2, \_, \_, T2),$ $phon(Syl, P1, \_, \_, T1), T5 = frikativ, P5 \neq P2.$
...	
5. Suchebene	
$diph, frik, liq, frik2, liq2.$	$T1 = diphtong, T2 = frikativ, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, P6, \_, \_, T6), phon(Syl, P5, \_, \_, T5),$ $phon(Syl, P4, \_, \_, T4), phon(Syl, P2, \_, \_, T2),$ $phon(Syl, P1, \_, \_, T1), T5 = frikativ, P5 \neq P2,$ $T6 = liquid, P6 \neq P4.$
$frik, short, liq, frik2, liq2.$	$T2 = frikativ, T3 = short, T4 = liquid, syl(Syl, Acc),$ $phon(Syl, P6, \_, \_, T6), phon(Syl, P5, \_, \_, T5),$ $phon(Syl, P4, \_, \_, T4), phon(Syl, P3, \_, \_, T3),$ $phon(Syl, P2, \_, \_, T2), T5 = frikativ, P5 \neq P2,$ $T6 = liquid, P6 \neq P4.$

Tafel 4.10 Resultat der Suche nach Kombinationen mit Mehrfachvorkommen.

Pos.	Hypothese	Bewertung	Pos.	Hypothese	Bewertung
1.	$short.$	0.022170	6.	$liq.$	0.012661
2.	$frik.$	0.021074	7.	$diph.$	0.012058
3.	$frik, short.$	0.016221	8.	$frik, short, liq.$	0.009974
4.	$frik, liq.$	0.015443	9.	$diph, frik.$	0.005695
5.	$short, liq.$	0.014779	10.	$frik, frik2.$	0.005402

Die zehn Hypothesen mit der besten Bewertung dieser Beispielanwendung sind in Tafel 4.10 aufgeführt. Die neun besten Hypothesen sind dieselben wie bei der Suche nach einfachen Kombinationen in Tafel 4.7. Als zehntbeste Teilgruppe wurde in diesem Suchraum jedoch die Silben mit zwei Frikativen gefunden.

4.3.4 Einfache Sequenzen

Dieses Anwendungsbeispiel untersucht, ob Silben, die bestimmte Sequenzen von Phonemtypen enthalten, interessante Teilgruppen der Population bilden. Eine solche Sequenz ist beispielsweise die Abfolge Diphtong-Frikativ. Diese Hypothese beschreibt Silben, die einen Diphtong und direkt dahinter einen Frikativ enthalten. Diese Beispielanwendung verwendet die rekursive Beziehung in der Relation phon, um die Reihenfolge der Phoneme zu bestimmen. Der Suchraum dieser Beispielanwendung erfordert aufwendigere Deklarationen als in den vorgehenden Beispielen. Die maximale Länge der Sequenzen ist aus Platzgründen auf 3 beschränkt. Um Sequenzen aus bis zu drei Phonemen bilden zu können, sind drei Verbinder nötig, die über die Phonem–Nachfolger–Beziehung miteinander verbunden sind. Eine Sequenz kann mit einem beliebigen Phonem der Silbe beginnen, das jeweils zweite Phonem ist direkter Nachfolger des ersten, und das dritte ist direkter Nachfolger des zweiten Phonems. Alle drei Phoneme sollen zur selben Silbe gehören, daher kommt die Variable Syl in allen drei Verbindern vor.

Die Merkmale legen den Typ des Phonems auf einen Wert fest. Die betrachteten Phonemtypen sind dieselben wie in den vorhergehenden Beispielanwendungen. Die excludes-Restriktionen verhindern widersprüchliche Hypothesen oder schließen Sequenzen aus, die mehrere Vokale enthalten wie etwa *diph1-diph2* oder *diph1-short2*. Die Merkmalsdeklarationen für zweite und dritte Phoneme in einer Sequenz enthalten requires-Bedingungen, die sicherstellen, daß diese Merkmale nur in Hypothesen abgefragt werden, die Merkmale eines ersten beziehungsweise zweiten Phonems abfragen. Tafel 4.11 zeigt die Deklarationen. Der damit aufgespannte Hypothesenraum enthält 48 Hypothesen. Ein Ausschnitt ist im Beispiel 4.12 wiedergegeben.

Die Ergebnisse der Suche sind in Tafel 4.13 zu sehen. Die besten Bewertungen erreichen die degenerierten Serien, die aus nur einem Phonem bestehen, wohl, weil sie am häufigsten sind. Wie in den vorhergehenden Anwendungen erreichen auch hier die beiden Kombinationen beziehungsweise Serien „Silben mit kurzem Vokal“ und „Silben mit Frikativ“ die höchsten Bewertungen. Tendenziell erhalten die Hypothesen, die Kombinationen beschreiben, eher höhere Bewertungen als die Sequenzen, möglicherweise weil sie als die allgemeineren Beschreibungen eher mehr Fälle abdecken.

4.3.5 Verallgemeinerte Sequenzen

Die Deklarationen aus dem vorhergehenden Abschnitt 4.3.4 ergeben einen Suchraum, in dem Sequenzen von Phonemtypen bis zur Länge 3 ausgewertet werden. Diesen Suchraum kann man erweitern um Sequenzen, in denen nicht alle Phoneme in der Sequenz auf einen Typ festgelegt sind. Eine solche Sequenz beschreibt zum Beispiel die Hypothese

Tafel 4.11 Deklaration des Suchraums für die Suche nach einfachen Sequenzen.

$link(syl, \quad [rel(''syl(Syl, Acc)'')]).$

$link(t1, \quad [rel(''phon(Syl, P1, \_, P2, T1)''), \quad from([syl])]).$

$link(t2, \quad [rel(''phon(Syl, P2, \_, P3, T2)''), \quad from([t1])]).$

$link(t3, \quad [rel(''phon(Syl, P3, \_, \_, T3)''), \quad from([t2])]).$

$lit(diph1, \quad [def(['' T1 = diphtong''']), \quad from([t1]),$
$excludes([frik1, short1, liq1, short2, diph2, short3, diph3])]).$

$lit(frik1, \quad [def(['' T1 = frikativ''']), \quad from([t1]),$
$excludes([diph1, short1, liq1])]).$

$lit(short1, \quad [def(['' T1 = short''']), \quad from([t1]),$
$excludes([diph1, frik1, liq1, short2, diph2, short3, diph3])]).$

$lit(liq1, \quad [def(['' T1 = liquid''']), \quad from([t1]),$
$excludes([diph1, frik1, short1])]).$

$lit(diph2, \quad [def(['' T2 = diphtong''']), \quad from([t2]),$
$excludes([frik2, short2, liq2, short1, diph1, short3, diph3]),$
$requires([[frik1, liq1]])]).$

$lit(frik2, \quad [def(['' T2 = frikativ''']), \quad from([t2]),$
$excludes([diph2, short2, liq2]),$
$requires([[diph1, frik1, short1, liq1]])]).$

$lit(short2, \quad [def(['' T2 = short''']), \quad from([t2]),$
$excludes([diph2, frik2, liq2, short1, diph1, short3, diph3]),$
$requires([[frik1, liq1]])]).$

$lit(liq2, \quad [def(['' T2 = liquid''']), \quad from([t2]),$
$excludes([diph2, frik2, short2]),$
$requires([[diph1, frik1, short1, liq1]])]).$

$lit(diph3, \quad [def(['' T3 = diphtong''']), \quad from([t3]),$
$excludes([frik3, short3, liq3, diph1, diph2, short1, short1]),$
$requires([[frik2, liq2]])]).$

$lit(frik3, \quad [def(['' T3 = frikativ''']), \quad from([t3]),$
$excludes([diph3, short3, liq3]),$
$requires([[diph2, frik2, short2, liq2]])]).$

$lit(short3, \quad [def(['' T3 = short''']), \quad from([t3]),$
$excludes([diph3, frik3, liq3, short1, short2, diph1, diph2]),$
$requires([[frik2, liq2]])]).$

$lit(liq3, \quad [def(['' T3 = liquid''']), \quad from([t3]),$
$excludes([diph3, frik3, short3]),$
$requires([[diph2, frik2, short2, liq2]])]).$

Tafel 4.12 Der Suchraum bei der Suche nach einfachen Sequenzen.

Hypothese	Logikrepräsentation
1. Suchebene	
$diph1.$	$T1 = diphtong, syl(Syl, Acc), phon(Syl, P1, \_, P2, T1).$
$frik1.$	$T1 = frikativ, syl(Syl, Acc), phon(Syl, P1, \_, P2, T1).$
$short1.$	$T1 = short, syl(Syl, Acc), phon(Syl, P1, \_, P2, T1).$
$liq1.$	$T1 = liquid, syl(Syl, Acc), phon(Syl, \_, \_, P2, T1).$
2. Suchebene	
$diph1, frik2.$	$T1 = diphtong, T2 = frikativ,\ syl(Syl, Acc),$ $phon(Syl, P2, P3, T2), phon(Syl, \_, \_, P2, T1).$
$diph1, liq2.$	$T1 = diphtong, T2 = liquid,\ syl(Syl, Acc),$ $phon(Syl, P2, P3, T2), phon(Syl, \_, \_, P2, T1).$
$frik1, diph2.$	$T1 = frikativ, T2 = diphtong,\ syl(Syl, Acc),$ $phon(Syl, P2, P3, T2), phon(Syl, \_, \_, P2, T1).$
$frik1, frik2.$	$T1 = frikativ, T2 = frikativ,\ syl(Syl, Acc),$ $phon(Syl, P2, P3, T2), phon(Syl, \_, \_, P2, T1).$
$frik1, short2.$	$T1 = frikativ, T2 = short,\ syl(Syl, Acc),$ $phon(Syl, P2, P3, T2), phon(Syl, \_, \_, P2, T1).$
...	
3. Suchebene	
$diph1, frik2, frik3.$	$T1 = diphtong, T2 = frikativ, T3 = frikativ, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, \_, P2, T1).$
$diph1, frik2, liq3.$	$T1 = diphtong, T2 = frikativ, T3 = liquid, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, \_, P2, T1).$
$diph1, liq2, frik3.$	$T1 = diphtong, T2 = liquid, T3 = frikativ, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, P2, T1).$
$diph1, liq2, liq3.$	$T1 = diphtong, T2 = liquid, T3 = liquid, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, P2, T1).$
$frik1, diph2, frik3.$	$T1 = frikativ, T2 = diphtong, T3 = frikativ, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, P2, T1).$
...	
$liq1, liq2, short3.$	$T1 = liquid, T2 = liquid, T3 = short, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, \_, P2, T1).$
$liq1, liq2, liq3.$	$T1 = liquid, T2 = liquid, T3 = liquid, syl(Syl, Acc),$ $phon(Syl, P3, P4, T3), phon(Syl, P2, P3, T2),$ $phon(Syl, \_, \_, P2, T1).$

Tafel 4.13 Resultat der Suche nach einfachen Sequenzen.

Pos.	Hypothese	Bewertung	Pos.	Hypothese	Bewertung
1.	$short1.$	0.022170	6.	$frik1, short2.$	0.007095
2.	$frik1.$	0.021074	7.	$short1, liq2.$	0.006253
3.	$liq1.$	0.012661	8.	$frik1, diph2.$	0.003666
4.	$diph1.$	0.012058	9.	$frik1, short2, liq3.$	0.003492
5.	$liq1, short2.$	0.008434	10.	$frik1, liq2.$	0.003375

$p1, p2, short2$ mit der Bedeutung „Silben mit mindestens zwei Phonemen, deren letztes ein kurzer Vokal ist“ und der Logikrepräsentation

$$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), T2 = short, syl(Syl, Acc).$$

Die Deklaration des Suchraums zeigt Tafel 4.14. Die Literale $phon(Syl, \_, \_, P2, T1)$, $phon(Syl, P2, P3, T2)$ und $phon(Syl, P3, \_, \_, T3)$, die im Anwendungsbeispiel 4.11 als Verbinder deklariert waren, sind hier als Merkmale deklariert. Das Merkmal $p1$ beschreibt die Silbeneigenschaft „enthält ein Phonem“. Die Merkmale $p2$ und $p3$ beschreiben die Silbeneigenschaften „enthält danach ein zweites Phonem“ beziehungsweise „enthält danach ein drittes Phonem“ und setzen $p1$ beziehungsweise $p2$ voraus. Das Anfügen eines dieser Literale realisiert in diesem Anwendungsbeispiel einen eigenständigen Spezialisierungsschritt. Die Definitionen der übrigen hier deklarierten Merkmale $diph1, frik1$, etc. sind Literale der Form $Ti = Wert$. Ihre Variablen stammen aus den Merkmalen $p1, p2$ und $p3$. Diese Merkmale führen die Merkmale $p1, p2$ und $p3$, aus denen ihre Variablen stammen, in ihrer requires-Liste und benötigen keine weiteren $from$-Deklarationen. Der Suchraum besteht aus 155 Hypothesen. Hypothesen, die Silben mit mehreren Vokalen beschreiben, sind dabei nicht ausgeschlossen. Beispiel 4.15 zeigt einen Ausschnitt aus dem Suchraum.

4.3.6 Grenzen des Ansatzes

Der hier entwickelte Suchalgorithmus führt keine Tests auf θ-Subsumtionsbeziehungen zwischen Hypothesen durch, um Redundanz bei der Hypothesenerzeugung zu vermeiden oder den Suchraum zu beschränken, sondern verwendet dazu lediglich die Ordnung des Hypothesenraums durch die Teilmengenbeziehung. Die Teilmengenbeziehung zwischen Hypothesen ist einfacher zu bestimmen als die θ-Subsumtion, deren Berechnung so teuer ist, daß sie den Vorteil der damit erzielten Verkleinerung des Suchraums zunichte machen kann. Da die Teilmengenbeziehung für ILP-Hypothesensprachen jedoch prinzipiell schwächer ist als die θ-Subsumtion, kann der Suchalgorithmus möglicherweise nicht alle im Suchraum existierenden θ-Subsumtionsbeziehungen zwischen Hypothesen erkennen. Genauso ist in diesem Ansatz nicht garantiert, daß die erzeugten Suchräume redundanzfrei sind. Redunzfreiheit und Erkennbarkeit der Subsumtionsbeziehungen durch die Teilmengenbedingung hängen von der Qualität der Deklarationen ab.

Die bisher behandelten Anwendungsbeispiele zeigen eine Auswahl von Hypothesenräumen, die sich mit dem hier entwickelten Sprachbias und Spezialisierungsoperator redundanzfrei erzeugen und mit der Teilmengenbeziehung beschränken lassen. Das folgende Beispiel 47 beschreibt eine hypothetische Anwendung mit einem Suchraum, in dem nicht alle Subsumtionsbeziehungen anhand der Teilmengenbedingung zu erkennen sind. In der darin angenommenen Datenbank gibt es zwei verschiedene Verknüpfungmöglichkeiten zwischen denselben Relationen.

Tafel 4.14 Deklarationen für die Suche nach verallgemeinerte Sequenzen.

$lit(p1,$	$[def([''phon(Syl,\_,\_,P2,T1)'']),$	$from([syl])]).$
$lit(p2,$	$[def([''phon(Syl,P2,P3,T2)'']),$	$from([syl]),$
	$requires([p1])]).$	
$lit(p3,$	$[def([''phon(Syl,P3,\_,\_,T3)'']),$	$from([syl]),$
	$requires([p2])]).$	
$lit(diph1,$	$[def([''T1 = diphtong'']),$	$requires([p1]),$
	$excludes([frik1,short1,liq1])]).$	
$lit(frik1,$	$[def([''T1 = frikativ'']),$	$requires([p1]),$
	$excludes([diph1,short1,liq1])]).$	
$lit(short1,$	$[def([''T1 = short'']),$	$requires([p1]),$
	$excludes([diph1,frik1,liq1])]).$	
$lit(liq1,$	$[def([''T1 = liquid'']),$	$requires([p1]),$
	$excludes([diph1,frik1,short1])]).$	
$lit(diph2,$	$[def([''T2 = diphtong'']),$	$requires([p2]),$
	$excludes([frik2,short2,liq2])]).$	
$lit(frik2,$	$[def([''T2 = frikativ'']),$	$requires([p2]),$
	$excludes([diph2,short2,liq2])]).$	
$lit(short2,$	$[def([''T2 = short'']),$	$requires([p2])$
	$excludes([diph2,frik2,liq2])]).$	
$lit(liq2,$	$[def([''T2 = liquid'']),$	$requires([p2]),$
	$excludes([diph2,frik2,short2])]).$	
$lit(diph3,$	$[def([''T3 = diphtong'']),$	$requires([p3]),$
	$excludes([frik3,short3,liq3])]).$	
$lit(frik3,$	$[def([''T3 = frikativ'']),$	$requires([p3]),$
	$excludes([diph3,short3,liq3])]).$	
$lit(short3,$	$[def([''T3 = short'']),$	$requires([p3]),$
	$excludes([diph3,frik3,liq3])]).$	
$lit(liq3,$	$[def([''T3 = liquid'']),$	$requires([p3]),$
	$excludes([diph3,frik3,short3])]).$	

Tafel 4.15 Der Suchraum bei der Suche nach verallgemeinerten Sequenzen.

Hypothese	Logikrepräsentation
1. Suchebene	
$p1.$	$phon(Syl, P1, P2, T1), syl(Syl, Acc).$
2. Suchebene	
$p1, p2.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), syl(Syl, Acc).$
$p1, diph1.$	$phon(Syl, P1, P2, T1), T1 = diphtong, syl(Syl, Acc).$
$p1, frik1.$	$phon(Syl, P1, P2, T1), T1 = frikativ, syl(Syl, Acc).$
...	
3. Suchebene	
$p1, p2, p3.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), phon(Syl, P3, \_, T3),$ $syl(Syl, Acc).$
$p1, p2, diph1.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), T1 = diphtong,$ $syl(Syl, Acc).$
...	
$p1, p2, diph2.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), T2 = diphtong,$ $syl(Syl, Acc).$
...	
4. Suchebene	
$p1, p2, p3, diph1.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2),$ $phon(Syl, P3, \_, T3), T1 = diphtong, syl(Syl, Acc).$
...	
$p1, p2, p3, lik3.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2),$ $phon(Syl, P3, \_, T3), T3 = liquid, syl(Syl, Acc).$
5. Suchebene	
$p1, p2, p3, diph1, diph2.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), phon(Syl, P3, \_, T3),$ $T1 = diphtong, T2 = diphtong, syl(Syl, Acc).$
$p1, p2, p3, diph1, diph3.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), phon(Syl, P3, \_, T3),$ $T1 = diphtong, T3 = diphtong, syl(Syl, Acc).$
...	
6. Suchebene	
...	
$p1, p2, p3, liq1, liq2, liq3.$	$phon(Syl, P1, P2, T1), phon(Syl, P2, P3, T2), phon(Syl, P3, \_, T3),$ $T1 = liquid, T2 = liquid, T3 = liquid, syl(Syl, Acc).$

Tafel 4.16 Resultat der Suche nach verallgemeinerten Sequenzen.

Pos.	Hypothese	Bewertung	Pos.	Hypothese	Bewertung
1.	$p1, p2, short2.$	0.023828	6.	$p1, p2, liq1.$	0.016486
2.	$p1, p2, p3, short2.$	0.023415	7.	$p1, p2, frik1.$	0.016052
3.	$p1, p2, short1.$	0.022489	8.	$p1, p2, p3, liq2.$	0.014619
4.	$p1, short1.$	0.022170	9.	$p1, liq1.$	0.012661
5.	$p1, frik1.$	0.021074	10.	$p1, p2, p3, frik1.$	0.012515

Das Ergebnis eines Suchlaufs mit diesem Suchraum zeigt Tafel 4.16. Wie in den vorhergehenden Beispielen zeichnet auch hier die Anwesenheit eines kurzen Vokals besonders interessante Hypothesen aus. Anders als in den vorhergehenden Beispielen gibt es hier jedoch mehrere Varianten der allgemeinen Eigenschaft „kurzer Vokal vorhanden“. Am besten bewertet ist die Hypothese $p1, p2, short2$, die bedeutet „ein kurzer Vokal und davor mindestens ein weiteres Phonem“. Darauf folgt die Hypothese $p1, p2, p3, short2$, also „ein kurzer Vokal und davor und dahinter je mindestens noch ein weiteres Phonem“. An dritter Stelle steht $p1, p2, short1$, das heißt „ein kurzer Vokal und dahinter mindestens ein weiteres Phonem“. Erst auf dem vierten Platz liegt die Hypothese, die in den vorhergehenden Beispielen am besten bewertet war, nämlich die allgemeinste Hypothese $p1, short1$ („kurzer Vokal vorhanden“).

Beispiel 47 *Gegeben sei eine Relation $p(P, Pv, Pn, G)$, die etwa einen Graph beschreibt als Menge von Knoten, wobei von jedem Knoten genau zwei Kanten zu anderen Knoten Pv und Pn ausgehen. Die beiden Knoten Pv und Pn können identisch sein. Das Attribut P ist der Primärschlüssel; seine Werte und die Werte von Pv und Pn sind Identifikatoren von Knoten. Das binäre Attribut G mit dem Wertebereich $\{1, 2\}$ gibt die Größe der Knoten an. Die Knoten bilden die Individuenpopulation. Zielgruppe seien die Knoten mit $G = 1$.*

Tafel 4.17 zeigt die Deklaration des Suchraums. Merkmale eines Knotens sind die Größen seiner Nachbarn $gv1, gv2, gn1$ und $gn2$. Die Verbinder pv und pn greifen auf die Tupel zu, die den linken beziehungsweise rechten Nachbarn eines Knotens repräsentieren. Ein weiteres Merkmal $pvpn$ eines Knotens ist, ob sein linker und sein rechter Nachbar identisch sind. Die Definition des Merkmals $pvpn$ identifiziert den linken und den rechten Nachbarn eines Knotens. Dieses Merkmal darf nur mit den Merkmalen $gv1$ und $gv2$, aber nicht mit den Merkmalen $gn1$ und $gn2$ kombiniert werden, weil die Hypothese $\{gv1, pvpn\}$ äquivalent ist zu $\{gn1, pvpn\}$ und die Hypothese $\{gv2, pvpn\}$ zu $\{gn2, pvpn\}$.

Die Deklarationen in Tafel 4.17 ergeben den Suchraum in Tafel 4.18. Zur zweiten Suchebene gehört die Hypothese $\{gv1, pvpn\}$, die Knoten beschreibt, deren linker und rechter Nachbar identisch ist und die Größe 1 hat. Der Suchraum enthält vier Verallgemeinerungen dieser Hypothese, nämlich $\{pvpn\}$, $\{gv1\}$, $\{gn1\}$ und $\{gv1, gn1\}$. Die Repräsentation der Hypothese als $\{gv1, pnpv\}$ spiegelt nur die Subsumtionsbeziehungen zu $\{pvpn\}$ und zu $\{gv1\}$ durch die Teilmengenbeziehung wieder.

Eine äquivalente Repräsentation, die alle existierenden Subsumtionsbeziehungen durch die Teilmengenbedingung sichtbar macht, ist $\{gv1, gn1, pnpv\}$. Diese Repräsentation hat jedoch die Teilmenge $\{gn1, pnpv\}$, die subsumtionsäquivalent ist zu $\{gv1, pnpv\}$ und deshalb mit einer excludes-Bedingung aus der Hypothesensprache ausgeschlossen wird. Dann ist auch die Hypothese $\{gv1, gn1, pnpv\}$ nicht mehr in der Hypothesensprache enthalten.

4.3.7 Fazit

Die Anwendungsbeispiele mit der Phonetik-Datenbank dienen lediglich zur Illustration des Sprachbias und lassen keine inhaltlich sinnvollen Resultate erwarten. Dies zeigt sich auch in den absolut gesehehen sehr niedrigen Interessantheitswerten der jeweils 10 besten

Tafel 4.17 Deklarationen zu Beispiel 47 „zwei alternative 1:1-Verbindungen“.

$$
\begin{array}{lllll}
link(p, & [rel(''p(P,Pv,Pn,G)'')]). & & & \\
link(pv, & [rel(\text{“}p(Pv,\_,\_,Gv)'')], & from([p])]). & & \\
link(pn, & [rel(\text{“}p(Pn,\_,\_,Gn)'')], & from([p])]). & & \\
 & & & & \\
lit(gv1, & [def([\text{“}Gv=1\text{“}]), & from([pv]), & excludes([gv2])]). \\
lit(gv2, & [def([\text{“}Gv=2\text{“}]), & from([pv]), & excludes([gv1])]). \\
lit(gn1, & [def([\text{“}Gn=1\text{“}]), & from([pn]), & excludes([gn2])]). \\
lit(gn2, & [def([\text{“}Gn=2\text{“}]), & from([pn]), & excludes([gn1])]). \\
lit(pvpn, & [def([\text{“}Pv=Pn\text{“}]), & from([p]), & excludes([gn1,gn2])]).
\end{array}
$$

Tafel 4.18 Der Suchraum zu zu Beispiel 47 „zwei alternative 1:1-Verbindungen“.

Hypothese	Logikrepräsentation
1. Suchebene	
$gv1.$	$Gv=1, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv).$
$gv2.$	$Gv=2, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv).$
$gn1.$	$Gn=1, p(P,Pv,Pn,G), p(Pn,\_,\_,Gn).$
$gn2.$	$Gn=2, p(P,Pv,Pn,G), p(Pn,\_,\_,Gn).$
$pvpn$	$Pv=Pn, p(P,Pv,Pn,G).$
2. Suchebene	
$gv1, gn1.$	$Gv=1, Gn=1, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv), p(Pn,\_,\_,Gn).$
$gv1, gn2.$	$Gv=1, Gn=2, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv), p(Pn,\_,\_,Gn).$
$gv1, pvpn.$	$Gv=1, Pv=Pn, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv).$
$gv2, gn1.$	$Gv=2, Gn=1, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv), p(Pn,\_,\_,Gn).$
$gv2, gn2.$	$Gv=2, Gn=2, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv), p(Pn,\_,\_,Gn).$
$gv2, pvpn.$	$Gv=2, Pv=Pn, p(P,Pv,Pn,G), p(Pv,\_,\_,Gv).$

Hypothesen. Daher ist es nicht sehr aufschlußreich, die Resultate dieser Anwendungen inhaltlich auszuwerten.

Die Anwendungsbeispiele zeigen, daß einfache und dennoch typische Hypothesenräume redundanzfrei erzeugt und durch die Teilmengenbeziehung strukturiert werden können. Im letzten Anwendungsbeispiel kann zwar der Hypothesenraum redundanzfrei erzeugt werden, jedoch lassen sich nicht alle Subsumtionsbeziehungen zur Suchraumbeschränkung nutzbar machen. Die Anwendungsbeispiele illustrieren außerdem, daß es schon für so kleine Datenbanken viele Alternativen gibt, einen Suchraum festzulegen, und daß sowohl die Deklarationen als auch die resultierenden Suchräume sehr umfangreich werden können.

Die Ergebnisse der verschiedenen Anwendungsbeispiele stimmen teilweise überein, verdeutlichen aber dennoch, daß die vier Anwendungsbeispiele tatsächlich unterschiedliche Fragestellungen bearbeiten, von denen jede für verschiedene Data Mining-Aufgaben am angemessensten sein kann.

4.4 Verwandte Arbeiten

Durch die Ausdrucksfähigkeit der verwendeten prädikatenlogischen Formalismen können die ILP-Hypothesenräume sehr umfangreich werden. Deshalb legt die ILP schon immer besonderen Wert auf die Entwicklung von Ansätzen zur Deklaration von Suchräumen und passenden Spezialisierungsoperatoren. In diesem Abschnitt werden Sprachbias und Suchmethoden von ILP-Systemen diskutiert, die ähnliche Aufgabenstellungen wie die hier definierte Teilgruppenanalyse bearbeiten. Darüberhinaus wird der Sprachbias eines Systems betrachtet, das eine andere Aufgabenstellung bearbeitet, dessen Sprachbias aber Gemeinsamkeiten mit dem hier entwickelten Sprachbias aufweist.

4.4.1 Regelentdecken mit Mobal/RDT DB

Das Regelentdeckungswerkzeug RDT (Rule Discovery Tool) [KW92] ist Bestandteil des Wissensmodellierungssystems Mobal [MWKE93] und dient zum Entdecken von Regeln in relationalen Datenbanken. Die Regeln haben die Form von Klauseln mit fest vorgegebenem Kopfliteral und beschreiben ein bestimmtes Konzept, für das positive und negative Beispiele vorgegeben sind. Mobal bearbeitet also eine individuenzentrierte Aufgabenstellung. Die Individuenpopulation besteht dabei aus den Instanziierungen der Variablen des vorgegebenen Kopfliterals q/n; die positiven und negativen Beispiele sind Grundfakten von q/n. Das Akzeptanzkriterium kann der Anwender den Zielen der Anwendung entsprechend definieren.

Regelmodelle. Die Hypothesensprache wird durch Regelmodelle festgelegt. Regelmodelle sind verallgemeinerte Klauseln, in denen Prädikatsymbole durch Prädikatvariablen ersetzt sind. Die Argumente der Prädikate, also Variablen und Konstanten, sind in den Regelmodellen buchstäblich vorgegeben. Aus den Regelmodellen erzeugt das System Hypothesen, indem es Prädikatvariablen mit Prädikatsymbolen instanziiert, die in der Datenbank vorkommen. Stelligkeit, Argumenttypen und Prädikatsorte der Prädikatsymbole können vorgegeben werden. Ein Prädikatsymbol wird dann nur mit damit übereinstimmenden Prädikaten ersetzt. Der Spezialisierungsoperator von RDT instanziiert die Regelmodelle in einer Reihenfolge, die eine Suche vom Allgemeinen zum Speziellen ergibt. Dazu werden die Regelmodelle in eine der θ-Subsumtion entsprechende partielle Ordnung gebracht. Auch der resultierende Suchraum ist nach der θ-Subsumtion geordnet. Um die Suche zu beschränken, speichert RDT die entdeckten kappbaren Hypothesen, und testet für jede neu generierte Hypothese, ob sie von einer kappbaren Hypothese θ-subsumiert ist. Wenn das zutrifft, wird die neue Hypothese aus dem Suchraum ausgeschlossen. Da RDT nur allgemeinste Regeln akzeptiert und Spezialisierungen akzeptierter Regeln ausschließt, testet es für jede neu generierte Hypothese außerdem, ob sie von einer bereits akzeptierten Hypothese θ-subsumiert ist, um sie gegebenenfalls aus dem Suchraum zu eliminieren. RDT führt also explizite θ-Subsumtions-Tests durch. Für Mobal/RDT wird berichtet, daß die Berechnung aller θ-Subsumtionsbeziehungen zwischen den Regeln zum Zweck der Suchraumbeschränkung so aufwendig ist, daß sie die Performanz des Systems wesentlich senkt statt sie zu erhöhen [Lin94].

Kopplung an ein RDBMS. Eine Erweiterung von Mobal/RDT ist RDT DB, das den RDT-Suchalgorithmus direkt an das kommerzielle relationale Datenbankmanagementsystem (RDBMS) Oracle anbindet [LM95, Lin94] und die Hypothesenevaluation durch SQL-Anfragen an das RDBMS realisiert. Dazu erzeugt RDT DB aus den Hypothesen automatisch geeignete SQL–Anfragen. Die Kopplung an das RDBMS ermöglicht es RDT DB, größere Datenbanken zu durchsuchen als das ursprüngliche RDT ohne RDBMS-Zugriff [LM95].

Um mit dem auf Regelmodellen basierenden Sprachbias eine Hypothesensprache für die Suche in einer relationalen Datenbank definieren zu können, ist es zweckmäßig, jedes Datenbankprädikat auf mehrere logische Prädikate abzubilden [Lin94]. Die Abbildung erleichtert es, Hypothesen zu beschreiben, die sich auf einzelne Attribute der Datenbankrelationen beziehen statt auf die Datenbankrelationen als Ganzes. Die abgeleiteten logischen Prädikate dienen nur zur Deklaration des Suchraums und zur verständlicheren Darstellung der Hypothesen. Die Datenbank selbst wird nicht transformiert. Eine mögliche Abbildung zeigt das folgende Beispiel.

Beispiel 48 *Gegeben sei eine Datenbank mit der Datenbank-Relation*

$$syl(S, Acc, Len, Height).$$

Dieses Prädikat wird in der Hypothesensprache nicht verwendet. Stattdessen wird es auf die folgenden Prädikate abgebildet, die dann zum Vokabular der Hypothesensprache gehören:

$$syl(S), acc\_syl(S, Acc), len\_syl(S, Len), height\_syl(S, Height).$$

4.4.2 Entdecken häufiger Muster mit WARMR

Das System WARMR läßt sich für verschiedene Varianten des Findens interessanter Sätze in relationalen Datenbanken einsetzen. Eine der Aufgabenstellungen, die WARMR löst, ist das Finden häufiger Anfragen (frequent query discovery). Das Finden häufiger Anfragen ist eine individuenzentrierte Aufgabenstellung. Die Individuenpopulation wird definiert als die Menge der Grundsubstitutionen θ eines vorgegebenen Schlüsselatoms KeyAtom, für die gilt $\mathcal{M}(D) \models \text{KeyAtom}\theta$. Das Schlüsselatom muß in allen betrachteten Hypothesen vorkommen. Diese Definition der individuenzentrierten Aufgabenstellung ist also sehr ähnlich zur Definition der Teilgruppenanalyse, die dieser Arbeit zugrundeliegt.

Kandidatenevaluation in WARMR. Der Suchalgorithmus von WARMR baut ebenfalls auf dem Apriori-Algorithmus auf [DT99, Deh98]. Während das hier vorgestellte System mit der Teilmengenbedingung ein Prinzip der Apriori- Kandidatenerzeugung auf den prädikatenlogischen Fall überträgt, liegt der Schwerpunkt bei WARMR darauf, die Ideen der Apriori-Kandidatenauswertung für den relationalen Fall zu übernehmen. Um sehr große Datenbanken durchsuchen zu können, die nicht vollständig in den Arbeitsspeicher des eingesetzten Rechners passen, kehrt der Apriori-Algorithmus das sonst übliche Vorgehen bei der Hypothesenauswertung um: statt eine Hypothesenliste sequentiell abzuarbeiten und dabei eine Hypothese nach der anderen auf der kompletten Datenbank zu evaluieren, lädt er Partitionen der Datenbank sukzessiv in den Arbeitsspeicher und evaluiert Mengen von Hypothesen gemeinsam auf der jeweils geladenen Partition. Aus den Ergebnissen der Auswertung auf den einzelnen Partitionen berechnet er die Bewertung jeder Hypothese. Dieses Vorgehen vermeidet sehr viel unnötigen Aufwand durch vielfach wiederholtes Laden der Datenbank. Für diese an der Datenbank statt an der Hypothesenliste orientierte Auswertungsstrategie des Apriori-Algorithmus ist die ebenenweise Suche besonders günstig, da die Hypothesen einer Suchebene zusammen eine geeignete Menge von Hypothesen für die gemeinsame Auswertung bilden, die es erlaubt, bei einer minimalen Anzahl von Datenbank-Durchläufen die Subsumtionsbeziehungen zwischen den Hypothesen zur Beschränkung der Suche auszunützen. WARMR führt eine ebenenweise Suche im relationalen Suchraum durch und wendet dabei die Auswertungsstrategie von Apriori an. Im multirelationalen Rahmen, wo die Relationen miteinander in Beziehung stehen, hängt ihr Erfolg davon ab, wie gut sich die Datenbank in voneinander möglichst unabhängige Partitionen teilen läßt.

WRMODE-Sprachdeklarationen. Die Deklaration des Hypothesenraums für die Suche nach häufigen Anfragen erfolgt durch WRMODE-Atome. Ein WRMODE-Atom ist eine abkürzende Schreibweise für ein Literal oder eine Menge von Literalen und hat die Form $p(A_1, \ldots, A_n)$. Ein A_i ist entweder eine Konstante oder hat die Form $+t_i$, $-t_i$ oder $\pm t_i$. Die t_i sind optional. Konstanten müssen in allen beschriebenen Literalen an der angegebenen Stelle vorkommen, die anderen Symbole werden durch Variablen ersetzt. Welche Variablen dabei möglich sind, hängt von der Hypothese h ab, an die das aus dem WRMODE-Atom abgeleitete Literal angefügt werden soll. Das Zeichen $+$ bedeutet, daß an der betreffenden Position im Literal eine alte Variable stehen muß. Das ist eine Variable, die bereits in anderen Literalen in der Hypothese h vorkommt. Das Zeichen $-$ bedeutet, daß an der jeweiligen Position eine neue Variable stehen muß, die in h noch nicht vorkommt. Bei $\pm$ sind alte und neue Variablen erlaubt. Das Symbol t_i bezeichnet einen Variablentyp. An einer mit $+t_i$ oder $\pm t_i$ markierten Position sind nur solche alte Variablen zugelassen, die in h an mit dem Typ t_i markierten Stellen stehen.

Die Suche erfolgt auch hier vom Allgemeinen zum Speziellen. Die allgemeinstmögliche Hypothese, mit der der Spezialisierungsprozeß startet, ist die Hypothese, die nur das Schlüsselatom enthält. Speziellere Hypothesen der $(i + 1)$-ten Suchebene entstehen durch Anfügen neuer Literale an die spezialisierbaren Hypothesen der i-ten Suchebene. Die angefügten Literale müssen mit einem deklarierten WRMODE-Atom konform sein. Zur Spezialisierung einer Hypothese h bestimmt der WRMODE-Spezialisierungsoperator alle Literale, die den Modus- und Typdeklarationen eines der deklarierten WRMODE-Atome genügen. Die Spezialisierungen von h ergeben sich dann durch Anhängen je eines dieser Literale. Auch der WRMODE-Spezialisierungsoperator bietet zusätzliche Kontrollmittel, die beispielsweise das Anfügen eines Literals von Vorhandensein oder Abwesenheit eines anderen Literals in der Hypothese abhängig machen. Diese Mittel ähneln den *req*- und *excl*-Bedingungen, sind in [Deh98] jedoch nicht näher ausgeführt.

Die Struktur des WARMR-Hypothesenraums beruht auf der θ-Subsumtion. Um den Suchraum zu beschränken, berechnet der WARMR-Suchalgorithmus die θ-Subsumtion zwischen den Hypothesen explizit. Dazu speichert er alle kappbaren Hypothesen aus früheren Suchebenen. Für jede neu erzeugte Hypothese wird geprüft, ob sie von einer kappbaren Hypothese aus einer früheren Suchebene θ-subsumiert ist. In diesem Fall wird sie ohne Datenbank-Auswertung verworfen. Um Redundanz im Hypothesenraum zu vermeiden, testet WARMR für jede neu erzeugte Hypothese, ob sie zu einer bereits erzeugten Hypothese der aktuellen Suchebene oder zu einer akzeptierten Hypothese aus einer früheren Suchebene subsumtions-äquivalent ist. Auch in diesem Fall wird die neue Hypothese eliminiert.

4.4.3 Multi-relationales Teilgruppenentdecken mit Midos

Das System Midos [Wro97] hat als erstes System die Aufgabenstellung des Findens interessanter Teilgruppen von aussagenlogischen auf relationale Datenbanken übertragen und die Verteilungsungewöhnlichkeit als Interessantheitsfunktion für den prädikatenlogischen Fall angepaßt. Midos ist in die kommerzielle Data Mining-Software Kepler [WWSE96] integriert. Sprachbias und Aufgabenstellung von Midos sind besonders auf relationale Datenbanken zugeschnitten. Die untersuchte Individuenpopulation wird identifiziert als die Werte der Schlüsselattribute der Relation $r_0(X_1, \ldots, X_M)$. Jede Teilgruppenbeschreibung muß das Atom $r_0(X_1, \ldots, X_M)$ enthalten, das also ein Populationsliteral darstellt. Die Abdeckung einer Teilgruppenbeschreibung h ist die Menge der Grundinstanzen σ der Schlüsselattribute, für die gilt $\mathcal{M} \models h\sigma$.

Der Fremdverbindungs-Bias. Zur Deklaration des Hypothesenraums bietet Midos den *Foreign Link*–Bias an. Dessen zentrales Ausdrucksmittel sind die Fremdverbindungen (*foreign links*). Der Ausdruck Fremdverbindung (*foreign link*) ist eine Verallgemeinerung des Begriffs Fremdschlüssel (*foreign key*). Schlüssel–Fremdschlüssel–Beziehungen sind beim

Datenbankentwurf vorgesehene Möglichkeiten, Relationen miteinander zu verknüpfen. Die Fremdverbindungen dienen ebenfalls dazu, die Verknüpfungen zwischen Relationen festzulegen, die in der deklarierten Hypothesensprache vorkommen sollen, sind aber nicht auf Schlüsselattribute eingeschränkt. Auch rekursive Fremdverbindungen sind möglich, bei denen die Fremdverbindung innerhalb einer Relation von einer Variable zur anderen weist. Der Benutzer legt eine Hypothesensprache für eine Anwendung fest, indem er eine Menge von Fremdverbindungen vorgibt. Daraus generiert das System automatisch die Hypothesensprache.

Der Spezialisierungsoperator in Midos. Midos verwendet eine logische Notation für Hypothesen. Die Relationen werden dazu auf Prädikate gleichen Namens und gleicher Stelligkeit abgebildet. Eine Hypothese ist bei Midos eine Konjunktion von positiven Literalen. Die Suche verläuft vom Allgemeinen zum Speziellen. Die allgemeinste Hypothese besteht aus dem Populationsliteral. Der Spezialisierungsoperator $\rho_{midos}(h)$ spezialisiert eine Hypothese $h = l_1, \ldots, l_n$ entweder, indem er ein einzelnes Literal l_i aus h spezialisiert, oder indem er neue Literale an die Hypothese anfügt.

Das Anfügen neuer Literale an eine Hypothese wird durch die deklarierten Fremdverbindungen gesteuert. Der Spezialisierungsoperator fügt neue Literale an eine Hypothese h an, wenn die Hypothese h ein Literal $p(V_1, \ldots, V_{a(p)})$ enthält, von dem eine Fremdverbindung $p[n] \rightarrow r[m]$ ausgeht. Dazu erzeugt er zunächst ein neues Literal $l' = r(U_1, \ldots, U_{a(r)})$ aus dem Symbol des Ziel-Prädikats der Fremdverbindung. Der Fremdverbindung $p[n] \rightarrow r[m]$ entsprechend steht an der m-ten Argumentposition des neuen Literals die n-te Variable des alten Literals $p(V_1, \ldots, V_{a(p)})$, also $V_n = U_m$. So sind alle Hypothesen individuenverbunden. Die anderen Variablen U_j im neuen Literal l' sind neu, das heißt sie kommen in der Hypothese h nicht vor. Wenn der Spezialisierungsoperator auf diese Weise ein neues Verbindungsliteral l' anlegt, erweitert er die Hypothese außerdem um any-Literale, die Möglichkeiten zur weiteren Verfeinerung der Hypothese anzeigen, ohne die Abdeckung der Hypothese zu verkleinern. Für jede neue Variable U im neuen Literal l' wird ein Literal $any(U)$ erzeugt, das in späteren Spezialisierungsschritten durch Literale ersetzt wird, die den Wertebereich der Variablen einschränken. Wenn der Wertebereich einer Variable V geordnet ist und die Form $\{v_1, v_2 \ldots, v_{m-1}, v_m\}$ aufweist, wird $any(V)$ durch ein Literal der Form $V \geq v_2$ oder $V \leq v_{m-1}$ ersetzt. Vergleichsliterale $V \geq v_i$ und $V \leq v_j$ werden später durch die spezielleren Literale $V \geq v_{i+1}$ beziehungsweise $V \leq v_{j+1}$ ersetzt. Für Variablen V mit baumartig strukturiertem Wertebereich werden Literale der Form $any(V)$ oder $V = a$ durch Literale $V = b_i$ ersetzt, wobei die b_i die allgemeinsten Elemente des Wertebereichs beziehungsweise die Nachfolger des Werts a sind. Wenn außer der zum Anfügen eines neuen Verbindungsliterals verwendeten Fremdverbindung $p[n] \rightarrow r[m]$ weitere Fremdverbindungen zwischen den Prädikaten p and r existieren, wird für jede dieser weiteren Fremdverbindungen ein zweistelliges any-Literal $any(V, U)$ generiert, das in folgenden Spezialisierungsschritten durch $V = U$ ersetzt werden kann. Der Spezialisierungsoperator führt die einzelnen Spezialisierungsschritte in einer bestimmten Reihenfolge aus, um zu verhindern, daß Hypothesen durch verschiedene Abfolgen von Spezialisierungsschritten mehrfach generiert werden.

4.4.4 Entdecken prädikatenlogischer Regeln mit Tertius

Tertius [FL01] löst ebenfalls eine eng verwandte Aufgabenstellung, nämlich das Entdecken von Regeln 1. Ordnung in individuenzentrierten Datenbanken oder in Datenbanken, die aus allgemeinen logischen Regeln bestehen. Wie WARMR läßt sich Tertius auf eine spezielle, individuenzentrierte Variante der Aufgabenstellung einschränken, die dem Finden interessanter Teilgruppen sehr ähnlich ist. Im individuenorientierten Kontext definiert Tertius die Individuen als Grundinstanzen von festen Individuenvariablen. Als wichtigsten Beitrag bei der Entwicklung des Systems Tertius betrachten die Autoren [FL01] die *confirmation*, eine

statistische Bewertungsfunktion für Regeln ohne vorher fixierten Klauselkopf oder vorher fixierte Indviduenpopulation. Für individuenzentrierte Anwendungen existiert eine Optimumschätzfunktion zur *confirmation*. Tertius sucht wahlweise die k Hypothesen mit der höchsten *confirmation* oder alle Hypothesen, deren *confirmation* einen vorab festgelegten Schwellwert übersteigt. Die Suchstrategie von Tertius ist die A*-Suche.

Der ISP-Sprachbias. Tertius bietet einen deklarativen Sprachbias, der besonders auf individuenzentrierte Aufgabenstellungen zugeschnitten ist. Der ISP-Sprachbias hat seinen Namen von den drei Arten von Prädikaten, die er unterscheidet: Individuenprädikate (I), Strukturprädikate (S) und Eigenschaftsprädikate (Properties) [LF00, FL01].

Individuenprädikate dienen dazu, die Individuen zu deklarieren, die in der jeweiligen Anwendung untersucht werden. Individuenprädikate sind Hilfsprädikate, die in der Datenbank-Repräsentation und den Hypothesen nicht in Erscheinung treten. Ihr Zweck ist lediglich, ein einheitliches Format für die Deklarationen aller drei Sprachbestandteile zu ermöglichen.

Strukturprädikate sind binäre Prädikate, die Verbindungen zwischen den Individuen in einer Datenbank beschreiben. In einer termbasierten Repräsentation verbinden sie einen komplexen Typ und seine Komponenten. In einer relationalen Datenbank verbinden sie Relationen miteinander. Strukturprädikate spezialisieren Hypothesen nicht, sondern dienen nur dazu, neue Variablen in eine Hypothese einzuführen. Strukturprädikate sind immer zweistellig. Jedes Literal mit einem Strukturprädikat, das an eine Hypothese angefügt wird, enthält immer eine alte Variable, die in der Hypothese bereits vorkommt, und eine neue Variable, die in der Hypothese noch nicht vorkommt. Zur Deklaration eines Strukturprädikats gehört auch die Angabe, ob es sich um eine $1:1$– oder $1:n$–Beziehung handelt.

Eigenschaftsprädikate führen keine neuen Variablen ein. In der Deklaration von Eigenschaftsprädikaten können Variablen mit dem Zeichen # markiert sein. Diese Variablen heißen Parameter und müssen immer mit einer Konstanten instanziiert werden.

Beispiel 49 *Eine Hypothesensprache für die Suche in der Phonetik-Datenbank läßt sich mit dem ISP-Sprachbias wie folgt deklarieren.*

$$
\begin{array}{ll}
Individual: & syl(syl). \\
Structural: & syl2phon(1:syl, phon). \\
Properties: & phontyp(phon, \#Typ).
\end{array}
$$

Der nichtredundante Spezialierungsoperator. Auch Tertius durchsucht den Hypothesenraum vom Allgemeinen zum Speziellen. Der Suchraum ist nach der θ-Subsumtion strukturiert. Der Spezialisierungsoperator kennt die drei elementaren Spezialisierungsoperationen Anfügen eines neuen Literals, Unifikation zweier Variablen und Instanziierung einer Variablen mit einer Konstanten aus ihrem Definitionsbereich. Wie bei Midos ist eine Reihenfolge fixiert, in der Spezialisierungsoperationen auf Hypothesen anzuwenden sind. Dadurch vermeidet es der Spezialisierungsoperator, redundante Hypothesen zu erzeugen.

4.4.5 Linus mit prädikatenlogischen Merkmalen

Linus [LD94] ist ein bekanntes System für prädiktive Anwendungen und gehört daher nicht zu den Systemen, die eine zum Finden interessanter Teilgruppen ähnliche Aufgabenstellung bearbeiten. Es ist jedoch interessant, die prädikatenlogischen Merkmale (P-Merkmale), um die Linus in [LF01] erweitert wurde, mit dem hier entwickelten Sprachbias zu vergleichen. P-Merkmale finden außerdem Verwendung in 1BC, einem ILP-System zum Bayes'schen Klassifikationslernen [FL99].

Der Linus-Grundalgorithmus

Linus lernt Klauselrümpfe zu einem vorgegebenen Klauselkopf. Der Grundalgorithmus besteht darin, das prädikatenlogische Problem in ein Attribut–Wert–Problem zu transformieren, mit einem aussagenlogischen Lernalgorithmus (wahlweise CN2, Assistent oder dem AQ-Vorgänger NewGem) zu lösen, und das Resultat wieder in prädikatenlogische Form zurück zu übertragen. Die Transformation erfolgt in zwei Schritten. Im ersten Schritt bestimmt Linus eine Liste von Literalen, die in den Klauselrümpfen vorkommen können. Im zweiten Schritt wird jedes dieser Literale in ein binäres propositionales Attribut überführt. Dazu wird jedes Literal mit den Belegungen der Kopfvariablen instanziiert und durch Anfragen an das Hintergrundwissen ausgewertet. Die Auswertung ergibt den Wahrheitswert des propositionalen Attributs für jede Belegung der Kopfvariablen.

Welche Literale erzeugt und in propositionale Attribute transformiert werden, wird durch Argumenttypen und Modusdeklarationen festgelegt. In früheren Version von Linus und seinem Nachfolger Dinus war die Hypothesensprache eingeschränkt auf bereichsbeschränkte beziehungsweise determinierte Klauseln. Bereichsbeschränkt sind Klauseln, in denen alle Rumpfvariablen auch im Klauselkopf vorkommen. In determinierten Klauseln ist die Belegung jeder Variable durch die Belegung der Kopfvariablen oder anderer, in der Klausel weiter links stehender Rumpfvariablen auf einen einzigen Wert festgelegt. Diese Einschränkungen der Hypothesensprache garantieren, daß sich für jede Belegung der Kopfvariablen ein eindeutiger Attributwert ergibt.

Prädikatenlogische Merkmale

Eine neue Erweiterung von Linus ist in der Lage, auch nicht-determinierte Klauseln zu lernen. Sie erfordert jedoch eine Einschränkung des Einsatzbereichs, nämlich auf individuenzentrierte Probleme. Das Vorgehen beruht auf der Konstruktion von sogenannten prädikatenlogischen Merkmalen (*first-order features*). Diese beschreiben Eigenschaften von Individuen und können auch nichtdeterminierte Variablen enthalten. Variablen in einer Klausel heißen global, wenn sie im Klauselkopf vorkommen, und lokal, wenn sie nur im Klauselrumpf, aber nicht im Klauselkopf enthalten sind. Formal ist ein prädikatenlogisches Merkmal (P-Merkmal) in einer Klausel eine Konjunktion von Rumpfliteralen, die globale und lokale Variablen enthalten, so daß keine der lokalen Variablen in anderen, nicht zum P-Merkmal gehörenden Literalen vorkommen [LF01]. Bei der Propositionalisierung werden die prädikatenlogischen Merkmale in die propositionalen Attribute transformiert, indem ihr Wahrheitswert für jedes Individuum berechnet wird.

Die P-Merkmale ermöglichen es, nichtdeterminierte Variablen im aussagenlogischen Lernalgorithmus zu verwenden, weil das P-Merkmal sozusagen die Nichtdeterminiertheit kapselt. Die nichtdeterminierte Variablenbelegung findet innerhalb des P-Merkmals statt und tritt in der Klausel außerhalb des P-Merkmals nicht in Erscheinung. Die zusätzliche Ausdrucksfähigkeit der Prädikatenlogik gegenüber der Aussagenlogik steckt innerhalb der P-Merkmale [LF01].

Für eine gegebene Anwendung sind meist sehr viele P-Merkmale bildbar. Die P-Merkmale, die für die Lösung einer Anwendung nützlich sind, müssen dem System von außen vorgegeben werden, manuell durch den Benutzer oder mit unabhängigen Methoden zum Entdecken interessanter Muster (teil-)automatisch ermittelt.

4.4.6 Vergleich und Diskussion

Beschränkung des Suchraums

Wie der hier entwickelte Suchalgorithmus durchsuchen auch die vier Systeme für verwandte Aufgabenstellungen, Mobal RDT, WARMR, Midos und Tertius, den Suchraum vom Allgemeinen zum Speziellen. Der Suchraum von RDT und von WARMR ist durch θ-Subsumtion geordnet. Diese Systeme berechnen θ-Subsumtionsbeziehungen zwischen

Hypothesen, um den Suchraum so weit wie möglich beschränken zu können. Bei RDT ist der Suchraum nicht in Ebenen eingeteilt. WARMR organisiert die Suche ebenenweise; die Ebeneneinteilung ist aber vorwiegend für die Kandidatenevaluation wichtig und betrifft weniger die Beschränkung des Suchraums. Das Vorgehen von WARMR bei der Beschränkung des Suchraums stimmt im wesentlichen mit dem von MOBAL/RDT überein. Die Berechnung der θ-Subsumtionsbeziehungen zwischen den Hypothesen verursacht großen Aufwand. und wird für WARMR als verbesserungswürdig eingestuft [Deh98]. Für Mobal/RDT haben sich die Subsumtionstests in einigen Anwendungen als zu aufwendig erwiesen, so daß das System ohne die damit erzielte Beschränkung des Suchraums effizienter war [Lin94].

Auch Midos und Tertius suchen vom Allgemeinen zum Speziellen. Ihre Spezialisierungsoperatoren vermeiden es, redundante Hypothesen zu erzeugen [Wro97, FL01]. Midos verzichtet darauf, neu generierte Hypothesen mit bereits evaluierten Hypothesen abzugleichen, um den Suchraum zu beschränken. Midos berechnet also keine θ-Subsumtionsbeziehungen zwischen den Hypothesen und führt keine ebenenweise Suche aus. Die Beschränkung des Suchraums beruht bei Midos ausschließlich auf dem Spezialisierungsoperator. Der Spezialisierungsoperator von Midos unterscheidet sich von den üblichen ILP–Spezialisierungsoperatoren dadurch, daß er Spezialisierung durch Abstieg in baumstrukturierten oder ordinalen Attributen integriert.

Der hier entwickelte Ansatz führt eine ebenenweise Suche vom Allgemeinen zum Speziellen durch. Er berechnet Subsumtionsbeziehungen zwischen den Hypothesen, um die Suche zu beschränken, jedoch berechnet er nicht die θ-Subsumtion, sondern die einfachere Teilmengenbeziehung zwischen den Hypothesen. Der Abstieg in baum-strukturierten oder ordinalen Attributen läßt sich auch in den hier entwickelten Suchalgorithmus integrieren (siehe Kapitel 5).

Spezialisierung mit zwei Arten von Literalen

RDT DB ist das erste ILP–System, das direkt an ein kommerzielles RDBMS angekoppelt wurde [Lin94]. Schon in dieser Arbeit wurde erkannt, daß es nützlich ist, die Spezialisierung von Hypothesen auf Attribute statt auf Prädikate auszurichten. In der Transformation von Datenbank- auf logische Prädikate, die für RDT DB entwickelt wurde, ist eine Unterscheidung in individuenbeschreibende Literale und spezialisierende Literale zu erkennen: So ist im Beispiel 48 das Literal $syl(S)$ individuenbeschreibend, während die Literale $acc\_syl(S, Acc)$, $len\_syl(S, Len)$, $height\_syl(S, Height)$ spezialisierend sind. Diese Transformation kann man als Vorstufe zur Unterscheidung zwischen Verbindern und Merkmalen des hier entwickelten Sprachbias betrachten. Die Unterscheidung ist auch beim Spezialisierungsoperator von Midos zu erkennen, der zwei Arten von Literalen generiert. Die Variabilisierungen von Datenbankprädikaten, die aus den Fremdverbindungs-Deklarationen abgeleitet werden, sind ähnlich zu den Verbindern, während die any-Literale den Merkmalen ähneln. Auch der ISP-Sprachbias von Tertius trennt mit den Struktur- und den Eigenschaftsprädikaten zwischen Prädikaten mit verbindender und spezialisierender Funktion. Da der Sprachbias von Mobal/RDT, Tertius, Midos und des hier entwickelten Ansatzes eine solche Unterscheidung zwischen verbindenden und spezialisierenden Literalen zeigen, scheint sie für die Deklaration von Hypothesenräumen für die bearbeitete Aufgabenstellung besonders günstig zu sein.

Deklaration der Hypothesensprache

Die vier Systeme verwenden unterschiedliche Methoden zur Deklaration der Hypothesensprache. In RDT beschreiben Regelmodelle die Hypothesensprache. Ein Regelmodell legt die Attribute der Literale für mehrere Hypothesenklauseln fest, die Prädikatsymbole werden geeignet instanziiert. Um Hypothesensprachen für relationale Datenbanken zu deklarieren, ist eine (implizite) Transformation der Datenrepräsentation nützlich.

WARMR verlangt Modus-Deklarationen für die Prädikate der Hypothesensprache; diese legen Einschränkungen für die Variablen der Literale fest, die der Spezialisierungsoperator mit dem betreffenden Prädikatsymbol erzeugen kann. Der Spezialisierungsoperator kann aus einer WRMODE-Literaldeklaration mehrere verschiedene Literale generieren. Laut [Deh98] erlaubt dieser Sprachbias eine besonders schnelle und bequeme Deklaration von Hypothesensprachen, bietet aber eher geringe Möglichkeiten, die Hypothesensprache genau und fein an die jeweilige Anwendung anzupassen.

Der ISP-Bias von Tertius ist auf individuenzentrierte Anwendungen zugeschnitten. Er verlangt ein spezielles Datenformat oder eine implizite Datentransformation, da die Strukturprädikate binär sind. Auch Tertius bietet einen komfortablen Sprachbias zur automatischen Erzeugung der Hypothesen.

Der Fremdverbindungs-Sprachbias von Midos ist besonders auf relationale Datenbanken ausgerichtet. Die Fremdverbindungen, die dabei zu deklarieren sind, legen fest, wie die Datenbankrelationen miteinander verbunden werden können. Aus diesen Deklarationen erzeugt der Spezialisierungsoperator automatisch die Literale, die die Hypothesen bilden. Dieser Sprachbias erlaubt also ebenfalls eine kompakte Beschreibung der Hypothesensprache, bietet aber geringere Möglichkeiten zur Feinabstimmung. So kann im Fremdverbindungs-Bias, anders als zum Beispiel im ISP-Bias, nicht vorgegeben werden, wie die Argumente der Literale (also die Attribute der Relationen) zu spezialisieren sind.

Der hier entwickelte Sprachbias verlangt, daß die Literale, aus denen die Hypothesen bestehen, buchstäblich vorab deklariert werden. Mit den Regelmodellen von RDT und RDT DB hat er gemein, daß die Argumente der Literale, also die Variablen und Konstanten, bei der Deklaration angegeben und festgelegt werden. Bei den übrigen Ansätzen (Fremdverbindungs-Bias, ISP und WRMODES) bestimmt das System automatisch, welche und wieviele Variablen und Variabilisierungen der Datenbankprädikate in den Hypothesen vorkommen.

Die Vorab-Deklaration der Literale und Variablen bedeutet jedoch keine echte Einschränkung, weil der Ansatz erlaubt, beliebig viele Literale zu deklarieren, von denen bei der Suche möglicherweise nur ein Teil verwendet wird. Im Unterschied zum ursprünglichen Apriori, in dem alle deklarierten Literale (die dort Artikel (*items*) heißen), mindestens einmal, nämlich auf der ersten Suchebene, ausgewertet werden, erlauben die *req*–Bedingungen im erweiterten Algorithmus die Deklaration von Literalen, die erst auf späteren Suchebenen (oder gar nicht) in Hypothesen aufgenommen und ausgewertet werden. Auch in modusbasierten Sprachbias–Formalismen wie WRMODE oder im Fremdverbindungsbias sind Längenbegrenzungen für Hypothesen gebräuchlich.

Die Systeme Mobal RDT, WARMR, Midos und Tertius bieten einen Sprachbias, der kompakte Deklarationen der Hypothesensprachen ermöglicht; die Erzeugung der Hypothesensprache ist stärker automatisiert und die Deklaration der Hypothesensprache benötigt weniger Aufwand seitens des Anwenders als der hier entwickelte Sprachbias. Andererseits scheint dieser eine flexiblere und feinere Festlegung von Hypothesenräumen zu erlauben als die weiter automatisierten, abstrakteren Ansätze. Auch der auf Modus–Deklarationen beruhende Sprachbias von WARMR bietet die Möglichkeit, die Hypothesensprache genauer zu modellieren durch erweiterte Deklarationen, die den Spezialisierungsoperator anweisen, Literale nur in Hypothesen aufzunehmen, in denen bestimmte andere Literale enthalten oder nicht enthalten sind [Deh98]. Diese zusätzlichen Ausdrucksmittel für die Feineinstellung der Hypothesensprache erhöhen ebenfalls den Aufwand bei der Deklaration.

P-Merkmale versus komplexe Merkmale

Als Propositionalisierungsverfahren unterscheidet sich Linus vom hier entwickelten Ansatz dadurch, daß bei der Propositionalisierung die P-Merkmale bereits vor dem Lernen ausgewertet werden, während sie bei nicht-propositionalisierenden Verfahren erst während des Lernens und nur soweit erforderlich ausgewertet werden. Die Propositionalisierung unter Verwendung von P-Merkmalen kann man als Verbesserung der Propositionalisierung

auf der Ebene von Hypothesen betrachten, die in Abschnitt 2.3.3 erwähnt wurde. Bei der Propositionalisierung auf der Ebene von Hypothesen wird jede Hypothese in ein aussagenlogisches Attribut überführt. Die P-Merkmale sind voneinander unabhängige Teile von Hypothesen, die zu aussagenlogischen Attributen werden. In Extremfällen degenerieren die P-Merkmale zu ganzen Hypothesen, und die Propositionalisierung unter Verwendung von P-Merkmalen degeneriert zur Propositionalisierung auf der Ebene von Hypothesen [LF01].

Die P-Merkmale von Lavrac und Flach sind ähnlich zu den komplexen Merkmalen des hier entwickelten Ansatzes in der Hinsicht, daß beide Ansätze mehrere Literale der Hypothesensprache zusammenfassen und aus der Sicht des Lernalgorithmus als ein Literal behandeln. Der Unterschied, daß bei Lavrac und Flach die strukturellen Literale in die P-Merkmale gehören, im hier entwickelten Ansatz die Verbindungsliterale dagegen gesondert behandelt werden, ist nicht prinzipieller Art, sondern, wie in Abschnitt 4.2.7 erläutert wurde, eher ein Unterschied der Notation. Ein wesentlicher Unterschied zwischen P-Merkmalen und den komplexen Merkmalen des hier entwickelten Ansatzes ist, daß die P-Merkmale als propositionale Attribute voneinander unabhängig sind und zwischen zwei P-Merkmalen keine Verbindungen über gemeinsame lokale Variablen bestehen sollen. Dagegen sind die komplexen Merkmale im hier entwickelten Ansatz nicht unabhängig voneinander, sondern können lokale Variablen teilen und stehen über *req*- und *excl*-Bedingungen miteinander in Beziehung, die zulässige Kombinationen von Merkmalen beschreiben und unzulässige Kombinationen verhindern. Dadurch ist es möglich, Subsumtionsbeziehungen zwischen Merkmalen für den Suchalgorithmus sichtbar zu machen und zur Beschränkung des Suchraums zu verwenden. In den P-Merkmalen nach [LF01, FL99] sind solche Subsumtionsbeziehungen verborgen.

Beispiel 50 *Angenommen, der Sprachbias erlaubt die P-Merkmale beziehungsweise Merkmale*

$$\begin{aligned} p_1 &= (phon(S, P_1), typ(P_1, frik)) \\ p_2 &= (phon(S, P_1), typ(P_1, frik), phon(S, P_2), typ(P_2 = frik), P_1 \neq P_2)\,. \end{aligned}$$

Das (P-)Merkmal p_1 subsumiert p_2. Anfügen von p_1 an eine Hypothese, die p_2 enthält, spezialisiert diese Hypothese nicht, sondern erzeugt eine subsumtionsäquivalente Hypothese.

Im hier entwickelten Ansatz kann eine entsprechende excl-Bedingung Hypothesen verbieten, die sowohl p_1 als auch p_2 enthalten. Alternativ kann eine req-Bedingung verlangen, daß p_2 nur zusammen mit p_1 in Hypothesen vorkommt. Filtert man die doppelten Vorkommen von Literale aus den Hypothesen mit p_1 und p_2, ergeben sich dieselben Hypothesen, die sich mit der Deklaration des kürzeren p_2' anstelle von p_2 ergeben:

$$p_2' = (phon(S, P_2), typ(P_2 = frik), P_1 \neq P_2)$$

Die excl- wie die req-Bedingung verhindern jeweils eine der beiden subsumtionsäquivalenten Hypothesen. Die req-Bedingung bietet den Vorteil, daß die Subsumtionsbeziehung zwischen p_1 und p_2 auf die Teilmengenbeziehung abgebildet wird ($\{p_2\}$ ist äquivalent zu $\{p_1, p_2\}$ und $\{p_1, p_2\}$ ist über die Teilmengenbeziehung als Spezialisierung von $\{p_1\}$ zu erkennen.)

Für die prädiktiven Aufgabenstellungen, für die die P-Merkmale im Rahmen von Linus verwendet werden, sind Redundanz und verborgene Subsumtionsbeziehungen ein geringeres Problem als für Aufgabenstellungen, die eine vollständige Suche verlangen, weil die Algorithmen für prädiktive Aufgabenstellungen üblicherweise auf Suchstrategien wie Hillclimbing oder Strahlensuche beruhen und sehr viel kleinere Suchräume durchlaufen. Im Gegenteil können die komplexen P-Merkmale beim prädiktiven Lernen vorteilhaft sein, weil sie eine vorausschauende Suche bewirken. Für die Effizienz einer vollständigen Suche ist es wichtiger, Redundanz zu vermeiden und Subsumtionsbeziehungen zwischen Merkmalen zur Beschränkung der Suche zu nutzen.

Kapitel 5

Allgemeinerbeziehungen zwischen Merkmalen

Um bei Data Mining-Anwendungen zu sinnvollen und nützlichen Entdeckungen zu kommen, ist es wichtig, Hintergrundwissen über den jeweiligen Anwendungsbereich einzubeziehen. In vielen Anwendungen liegt beispielsweise Hintergrundwissen in Form von Taxonomien vor. Die Taxonomien erlauben es, allgemeinere Muster zu bilden und zu entdecken, als sonst möglich wäre. In der dieser Arbeit zugrundeliegenden Formalisierung der Teilgruppenanalyse lassen sich Taxonomien als Systeme von Allgemeinerbeziehungen zwischen Merkmalen ausdrücken. Solche Allgemeinerbeziehungen bieten Ansatzpunkte zur Beschränkung des Suchraums. Dieses Kapitel beschreibt eine Methode, die Beschränkung des Suchraums anhand von Allgemeinerbeziehungen zwischen Merkmalen in den Algorithmus zur ebenenweisen Suche in relationalen Suchräumen zu integrieren.

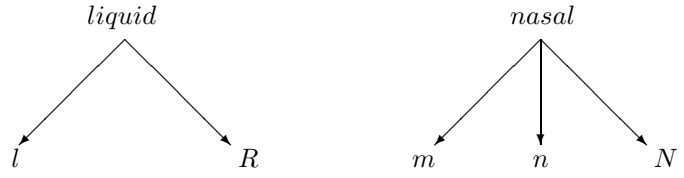
5.1 Einbeziehung von Taxonomien in den Algorithmus

5.1.1 Taxonomien und Allgemeinerbeziehungen zwischen Merkmalen

Das von Klösgen und Zytkow zusammengestellte Data Mining-Glossar [KZ] definiert eine Taxonomie τ als ein hierarchisches System von ausgewählten Teilmengen des Wertebereichs eines Attributs, die den gesamten Wertebereich des Attributs abdecken und meist in einer Baumstruktur arrangiert sind, wobei sich auf derselben Ebene der Baumstruktur angesiedelte Teilmengen nicht überschneiden. Attribute, auf denen eine Taxonomie definiert ist, bezeichnet man auch als strukturierte Attribute.

Beispiel 51 *Für die Phonetik-Daten existiert die in Tabelle 6.6 gezeigte Taxonomie auf Phonemtypen. In Abbildung 5.1 ist ein Fragment davon graphisch wiedergegeben.* lc_l *und* lc_R *sind Liquide, also Gleitlaute, und* nc_m, nc_n, nc_N *sind Nasallaute.*

Abbildung 5.1 Auszug aus der Taxonomie auf Phonemtypen in Tabelle 6.6.



Tafel 5.1 Deklaration von Merkmalen auf einem strukturierten Attribut.

$link(phon,$	$[def(''phon(\_, P, C, A, \_)'')]).$	
$lit(liquid,$	$[def([''liquid(A)'']),$	$from([phon]).$
$lit(nasal,$	$[def([''nasal(A)'']),$	$from([phon]).$
$lit(l,$	$[def([''A = l'']),$	$from([phon]).$
$lit('R',$	$[def([''A = 'R'']),$	$from([phon]).$
$lit('N',$	$[def([''A = 'N'']),$	$from([phon]).$
$lit(n,$	$[def([''A = n'']),$	$from([phon]).$
$lit(m,$	$[def([''A = m'']),$	$from([phon]).$

Im Rahmen der hier erarbeiteten Formalisierung der Teilgruppenanalyse kann eine Taxonomie als eine Menge von Allgemeinerbeziehungen zwischen Merkmalen formalisiert werden.

Beispiel 52 *Tafel 5.1 zeigt die Deklaration von Merkmalen, die den Werten des Attributs A in der Relation $phon$ entsprechen. Für jeden Wert aus dem Wertebereich des Attributs, l, R, N, n und m, und für die verallgemeinerten Werte $liquid$ und $nasal$ ist ein Merkmal deklariert. Auf dem Wertebereich des Attributs A sei die in Abbildung 5.1 gezeigte Taxonomie definiert. Dann ist das Merkmal $liquid$ allgemeiner als die Merkmale l und R, und das Merkmal $nasal$ ist allgemeiner als die Merkmale N, n und m. Wenn zwei Merkmale l_1 und l_2 in einer Allgemeiner-Relation $l_1 >_\tau l_2$ zueinander stehen, so ist die Menge der Individuen, die das Merkmal l_2 aufweisen, eine Teilmenge der Individuen, die das Merkmal l_1 aufweisen.*

Im folgenden bezeichnet der Begriff Taxonomie eine Menge von Allgemeinerbeziehungen zwischen Merkmalen wie folgt.

Definition 36 *Eine Taxonomie τ ist eine Menge von Allgemeinerbeziehungen $>_\tau$ zwischen Merkmalen, so daß jedes Merkmal l höchstens einen Vorgänger l' mit $l' >_\tau l$ hat, die Allgemeinerbeziehungen keine Zyklen bilden und jede Allgemeinerbeziehung $l_1 >_\tau l_2$ im gegebenen Anwendungsbereich mit der extensionalen Subsumtion übereinstimmt, das heißt, für alle Teilgruppenbeschreibungen h gilt:*

$$\textit{wenn } l_1 >_\tau l_2, \textit{ dann } cov(h \cup \{l_2\}) \subseteq cov(h \cup \{l_1\}).$$

Ein Merkmal l_i heißt allgemeiner bezüglich τ als ein Merkmal l_j, wenn $l_i >_\tau l_j$. Das allgemeinere Merkmal l_i heißt dann auch Vorgänger des spezielleren Merkmals l_j, und l_j heißt Nachfolger von l_i. Für Merkmale l_i, l_j mit $l_i >_\tau l_j$, heißt l_i direkter Vorgänger von l_j und l_j heißt direkter Nachfolger von l_i, wenn es keinen dazwischenliegenden Merkmal l_k mit $l_i >_\tau l_k >_\tau l_j$ gibt.

Diese Definition ist allgemeiner als die Definition von Klösgen und Zytkow, weil sie nicht verlangt, daß Merkmale auf derselben Hierarchie-Ebene disjunkt sind.

Allgemeinerbeziehungen zwischen Merkmalen können auch unabhängig von Taxonomien auf Attributwerten bestehen.

Beispiel 53 *Tafel 5.2 zeigt die Deklarationen von Merkmalen, die zwei numerische Attribute der untersuchten Individuen miteinander vergleichen, nämlich die Länge L und die Höhe H. Dabei ist das Merkmal „nicht-quadrat“ mit der Definition $H \neq L$ allgemeiner als die Merkmale „hochkant“ mit der Definition $H > L$ und „quer“ mit der Definition $H < L$.*

Tafel 5.2 Deklaration von Merkmalen mit Vergleichen zwischen Attributwerten.

$link(obj,$	$[def(''obj(O,H,L)'')]).$	
$lit(quadrat,$	$[def([''H = L'']),$	$from([obj]).$
$lit(nicht\_quadrat,$	$[def([''H \neq L)'']),$	$from([obj]).$
$lit(hochkant,$	$[def([''H > L'']),$	$from([obj]).$
$lit(quer,$	$[def([''A = 'H < L''']),$	$from([obj]).$

5.1.2 Die Genex-Repräsentation

Teilgruppenbeschreibungen können nicht nur durch Anfügen weiterer Merkmale spezialisiert werden, sondern auch dadurch, daß ein Merkmal durch einen seiner direkten Nachfolger ersetzt wird. Ein solcher Spezialisierungsschritt wird auch als „Abstieg in der Taxonomie“ bezeichnet. Allgemeinerbeziehungen zwischen einzelnen Merkmalen induzieren dadurch Allgemeinerbeziehungen zwischen Teilgruppenbeschreibungen.

Definition 37 *Eine Teilgruppenbeschreibung $h = \{l_1, \ldots, l_n\}$ ist allgemeiner als eine Teilgruppenbeschreibung $h' = \{l'_1, \ldots, l'_m\}$ bezüglich einer Taxonomie τ (geschrieben $h \succ_\tau h'$), wenn es für jedes $l \in h$ ein $l' \in h'$ gibt, so daß $l = l'$ oder $l \succ_\tau l'$.*

Wie die Teilmengenbeziehung $\subseteq$ kann auch die Allgemeinerbeziehung $\succ_\tau$ zwischen Teilgruppenbeschreibungen dazu dienen, den Suchraum einzuschränken. Aus der Definition 37 folgt unmittelbar, daß die Allgemeinerbeziehung bezüglich τ zwischen Teilgruppenbeschreibungen die Teilmengenbeziehung als Allgemeinerbeziehung einschließt, das heißt:

Proposition 6 *Für zwei Teilgruppenbeschreibungen h_1 und h_2 und eine Taxonomie τ gilt: wenn $h_1 \subseteq h_2$, dann $h_1 \succ_\tau h_2$.*

Die Genex-Repräsentation ist eine Darstellungsform für Teilgruppenbeschreibungen, die die Spezialisierung durch Abstieg in der Taxonomie in der Darstellung einer Teilgruppenbeschreibung sichtbar macht. Genex bedeutet „**Gen**eralisierung **ex**plizit“. Eine Hypothese ist in Genex-Repräsentation, wenn sie für jedes Merkmal, das sie enthält, auch alle bezüglich der Taxonomie allgemeineren Merkmale enthält.

Definition 38 *Eine Teilgruppenbeschreibung h ist in Genex-Repräsentation bezüglich einer Taxonomie τ, falls für alle $l \in h$ gilt: wenn $\exists_{l'} l' \succ_\tau l$, dann $l' \in h$.*

Für Teilgruppenbeschreibungen in Genex-Repräsentation gilt die zur Proposition 6 umgekehrte Implikation in Proposition 7.

Proposition 7 *Für zwei Teilgruppenbeschreibungen h und h' in Genex-Form und eine Taxonomie τ gilt: wenn $h' \succ_\tau h$, dann $h' \subseteq h$.*

Beweis 8 *Definition 38 besagt:*

$$h \text{ genex und } l \in h \text{ und } \exists_{l'} l' \succ_\tau l \Longrightarrow l' \in h$$

Damit und mit Definition 37 gilt:

$$\begin{aligned} h' \succ_\tau h &\Longrightarrow \forall_{l \in h'} \exists_{l' \in h} l = l' \text{ oder } l \succ_\tau l' \\ &\Longrightarrow \forall_{l \in h'} \exists_{l' \in h} l = l' \\ &\Longrightarrow h' \subseteq h \end{aligned}$$

♢

Die Genex-Repräsentation erlaubt, die Beschränkung des Suchraums anhand von Allgemeinerbeziehungen zwischen Merkmalen zu integrieren mit der Suchraumbeschränkung durch die Teilmengenbeziehung. Das bedeutet, daß die Teilmengenbeziehung und die Teilmengenbedingung genügen, um Subsumtionsbeziehungen zwischen Teilgruppenbeschreibungen, die auf Allgemeinerbeziehungen zwischen Merkmalen zurückgehen, zu erkennen und zur Beschränkung des Suchraums zu verwerten, sofern alle Teilgruppenbeschreibungen in Genex-Form sind.

Die Genex-Repräsentation einer Teilgruppenbeschreibung h kann redundant sein, da sie Merkmale enthalten kann, die ihre Abdeckung *cov*(h) nicht ändern, das heißt, es kann $l \in h$ geben, so daß *cov*$(h) =$ *cov*$(h \setminus \{l\})$. Die Genex-Repräsentation für Teilgruppenbeschreibungen ist aber eindeutig bestimmt, das heißt, es gibt keine syntaktischen Varianten der Genex-Repräsentation für dieselbe Teilgruppenbeschreibung. Bezüglich einer Taxonomie τ existiert für jede reduzierte Teilgruppenbeschreibung h genau eine Teilgruppenbeschreibung h_τ in Genex-Form. Wenn also alle Teilgruppenbeschreibungen in Genex-Form gebracht werden, erzeugt der Übergang von einer aus reduzierten Hypothesen bestehenden Hypothesensprache L zu einer Hypothesensprache L_τ in Genex-Repräsention weder Teilgruppenbeschreibungen, die zu anderen Teilgruppenbeschreibungen in der Hypothesensprache L_τ subsumtions-äquivalent sind, noch solche, für die es in L keine Entsprechung in reduzierter Form gibt.

5.1.3 Einbindung in den Such-Algorithmus

Wegen des Zusammenhangs in Proposition 7 leistet der im Kapitel 4 entwickelte Suchalgorithmus, der den Suchraum mittels der Teilmengenbeziehungen zwischen Teilgruppenbeschreibungen beschränkt, auch eine Beschränkung des Suchraums aufgrund der Beziehung $\succ_\tau$. Voraussetzung dafür ist, daß alle Teilgruppenbeschreibungen in Genex-Repräsentation vorliegen. Die Generierung aller Teilgruppenbeschreibungen in Genex-Repräsentation läßt sich durch geeignete Deklaration der Merkmale erreichen. Dazu wird die Deklaration jedes Merkmals l, das einen Vorgänger $l' \succ_\tau l$ in der Taxonomie τ besitzt, um eine *req*-Bedingung ergänzt, so daß das Merkmal l diesen Vorgänger l' erfordert.

Bedingung 1 (Genex-Form) *Alle Hypothesen werden in Genex-Repräsentation generiert, wenn für die Deklaration jedes Merkmals l gilt: wenn $l' \succ_\tau l$, dann $l' \in req(l)$.*

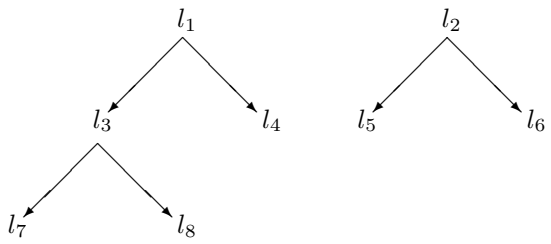
Damit die Hypothesensprache vollständig erzeugt wird, muß die Ordnung $>_l$, die eine Reihenfolge definiert, in der die Literale in einer Hypothese aufeinanderfolgen, so gewählt werden, daß die requires-Liste eines Literals l nur Literale l' mit $l >_l l'$ enthält (vergleiche Abschnitt 4.1.3).

Bedingung 2 *Die Ordnung $<_l$ muß so gewählt werden, daß gilt: wenn $l' \succ_t aul$, dann $l >_l l'$.*

Die *req*-Bedingungen bewirken, daß alle Teilgruppenbeschreibungen in *Genex*-Repräsentation generiert werden. Weil alle Vorgänger eines Merkmals in einer Hypothese enthalten sein müssen, werden die Hypothesen in der Reihenfolge zunehmender Spezifität bezüglich der Taxonomie erzeugt. Wenn eine Hypothese das Ausschlußkriterium erfüllt und gekappt wird, wird damit auch ihre weitere Spezialisierung durch Abstieg in der Taxonomie unterbunden. Das folgende Beispiel illustriert den Aufbau und die Beschränkung eines Suchraums unter Verwendung von Taxonomie-Information.

Beispiel 54 *Tafel 5.3 zeigt den Aufbau und die Beschränkung eines Suchraums, der sich mit acht Merkmalen ergibt, die in den in Abbildung 5.2 wiedergegebenen Allgemeinerbeziehungen zueinander stehen. Die Hypothesen sind unabhängig von einer konkreten Anwendung nur als Merkmalslisten ohne Logikrepräsentation dargestellt.*

Abbildung 5.2 Abstrakte Taxonomie zum Beispiel 54.



Für jede Allgemeinerbeziehung zwischen Merkmalen, die in Abbildung 5.2 durch einen Pfeil repräsentiert ist, gibt es entsprechende req-Bedingungen in den Merkmalsdeklarationen, also:

$$\begin{array}{lll} req(l_1) = \emptyset, & req(l_3) = \{l_1\}, & req(l_4) = \{l_1\}, \\ req(l_7) = \{l_3\}, & req(l_8) = \{l_3\}, & \\ req(l_2) = \emptyset, & req(l_5) = \{l_2\}, & req(l_6) = \{l_2\}. \end{array}$$

Es gibt keine weiteren req- oder excl-Bedingungen.

In der Tafel sind alle Hypothesen aufgeführt, die sich ergeben durch Anfügen eines Literals an eine der Hypothesen, die in der jeweiligen Vorgänger-Ebene generiert wurden. In runden Klammern stehen unzulässige Hypothesen, die nicht generiert werden, weil sie eine req-Bedingung verletzen.

Wenn die doppelt unterstrichene Hypothese $\{1, 3\}$ *das Beschränkungskriterium erfüllt und gekappt wird, verschwinden alle einfach unterstrichenen Hypothesen aus dem Suchraum.*

5.2 Anwendungsbeispiele

Das vorhergehende Beispiel 5.3 demonstriert den Aufbau und die Beschränkung eines Suchraums mit acht Merkmalen, die in einer Taxonomie angeordnet sind, unabhängig von einer konkreten Anwendung. Dieser Abschnitt zeigt zwei Anwendungsbeispiele für die Phonetik-Daten, die den Taxonomie-Ausschnitt in Abbildung 5.1 verwenden.

Eine Taxonomie von Merkmalen, in der mehrere Nachfolger eines gemeinsamen Vorgängers für ein Individuum zutreffen können, wird hier als kombinierende Taxonomie bezeichnet. Eine Taxonomie, bei der die Nachfolger gemeinsamer Vorgänger nicht miteinander vereinbar sind und sich gegenseitig ausschließen, wird hier als ausschließende Taxonomie bezeichnet.

Ein Suchraum für eine ausschließende Taxonomie wird deklariert, indem zusätzlich zu den in Bedingung 1 genannten *req*-Bedingungen noch die folgenden *excl*-Bedingungen spezifiziert werden, durch die sich Nachfolger gemeinsamer Vorgänger gegenseitig ausschließen:

Bedingung 3 (Ausschließende Taxonomie)

wenn $l' \succ_\tau l_1$ *und* $l' \succ_\tau l_2$, *dann* $l_1 \in excl(l_2)$ *und* $l_2 \in excl(l_1)$.

Das Beispiel 54 gilt demzufolge für eine kombinierende Taxonomie.

Beispiel 55 *Wenn die Individuenpopulation aus Silben besteht, induziert die Taxonomie in Abbildung 5.1 eine kombinierende Taxonomie auf den Silbenmerkmalen, da beispielsweise*

Tafel 5.3 Aufbau und Beschränkung eines Suchraums mit der Taxonomie in Abbildung 5.2.

Ebene	Hypothese
1	$\{l_1\}, \{l_2\}, (\{l_3\}, \{l_4\}, \{l_5\}, \{l_6\}, \{l_7\}, \{l_8\})$
2	$\{l_1, l_2\}, \underline{\{l_1, l_3\}}, \{l_1, l_4\}, (\{l_1, l_5\}, \{l_1, l_6\}, \{l_1, l_7\}, \{l_1, l_8\}),$ $(\{l_2, l_3\}, \{l_2, l_4\}), \{l_2, l_5\}, \{l_2, l_6\}, (\{l_2, l_7\}, \{l_2, l_8\})$
3	$\underline{\{l_1, l_2, l_3\}}, \{l_1, l_2, l_4\}, \{l_1, l_2, l_5\}, \{l_1, l_2, l_6\}, (\{l_1, l_2, l_7\}, \{l_1, l_2, l_8\}),$ $\underline{\{l_1, l_3, l_4\}}, (\{l_1, l_3, l_5\}, \{l_1, l_3, l_6\}), \underline{\{l_1, l_3, l_7\}}, \underline{\{l_1, l_3, l_8\}},$ $(\{l_1, l_4, l_5\}, \{l_1, l_4, l_6\}, \{l_1, l_4, l_7\}\{l_1, l_4, l_8\}),$ $\{l_2, l_5, l_6\}, (\{l_2, l_5, l_7\}, \{l_2, l_5, l_8\}),$ $(\{l_2, l_6, l_7\}, \{l_2, l_6, l_8\})$
4	$\underline{\{l_1, l_2, l_3, l_4\}}, \underline{\{l_1, l_2, l_3, l_5\}}, \underline{\{l_1, l_2, l_3, l_6\}}, \underline{\{l_1, l_2, l_3, l_7\}}, \underline{\{l_1, l_2, l_3, l_8\}},$ $\{l_1, l_2, l_4, l_5\}, \{l_1, l_2, l_4, l_6\}, (\{l_1, l_2, l_4, l_7\}, \{l_1, l_2, l_4, l_8\}),$ $\{l_1, l_2, l_5, l_6\}, (\{l_1, l_2, l_5, l_7\}, \{l_1, l_2, l_5, l_8\}),$ $(\{l_1, l_2, l_6, l_7\}, \{l_1, l_2, l_6, l_8\}),$ $(\{l_1, l_3, l_4, l_5\}, \{l_1, l_3, l_4, l_6\}), \underline{\{l_1, l_3, l_4, l_7\}}, \underline{\{l_1, l_3, l_4, l_8\}},$ $\underline{\{l_1, l_3, l_7, l_8\}},$ $(\{l_2, l_5, l_6, l_7\}, \{l_2, l_5, l_6, l_8\})$
5	$\underline{\{l_1, l_2, l_3, l_4, l_5\}}, \underline{\{l_1, l_2, l_3, l_4, l_6\}}, \underline{\{l_1, l_2, l_3, l_4, l_7\}}, \underline{\{l_1, l_2, l_3, l_4, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_5, l_6\}}, \underline{\{l_1, l_2, l_3, l_5, l_7\}}, \underline{\{l_1, l_2, l_3, l_5, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_6, l_7\}}, \underline{\{\{l_1, l_2, l_3, l_6, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_7, l_8\}},$ $\{l_1, l_2, l_4, l_5, l_6\}, (\{l_1, l_2, l_4, l_5, l_7\}, \{l_1, l_2, l_4, l_5, l_8\}),$ $(\{l_1, l_2, l_4, l_6, l_7\}, \{l_1, l_2, l_4, l_6, l_8\}),$ $\underline{\{l_1, l_3, l_4, l_7, l_8\}}$
6	$\underline{\{l_1, l_2, l_3, l_4, l_5, l_6\}}, \underline{\{l_1, l_2, l_3, l_4, l_5, l_7\}}, \underline{\{l_1, l_2, l_3, l_4, l_5, l_8\}}\ ,$ $\underline{\{l_1, l_2, l_3, l_4, l_6, l_7\}}, \underline{\{l_1, l_2, l_3, l_4, l_6, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_4, l_7, l_8\}},$ $\{l_1, l_2, l_4, l_5, l_6, l_7\}, \{l_1, l_2, l_4, l_5, l_6, l_8\},$ $\{l_1, l_2, l_4, l_5, l_7, l_8\},$ $\{l_1, l_2, l_4, l_6, l_7, l_8\}$
7	$\underline{\{l_1, l_2, l_3, l_4, l_5, l_6, l_7\}}, \underline{\{l_1, l_2, l_3, l_4, l_5, l_6, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_4, l_5, l_7, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_4, l_6, l_7, l_8\}},$ $\underline{\{l_1, l_2, l_3, l_5, l_6, l_7, l_8\}}$
8	$\underline{\{l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8\}}$

die Silbenmerkmale „enthält ein l“ und „enthält ein R“ zugleich für eine Silbe zutreffen können.

Wenn die Individuenpopulation jedoch aus Phonemen besteht, induziert die Taxonomie in Abbildung 5.1 eine ausschließende Taxonomie auf Phonemmerkmalen, da sich beispielsweise die Phonemmerkmale „ist ein l“ und „ist ein R“ offensichtlich gegenseitig ausschließen.

Tafel 5.4 Deklaration des Suchraums für eine kombinierende Taxonomie.

$link(syl,$	$[def(''syl(Syl,\_,Acc)'')]).$	
$link(p1,$	$[def(''phon(Syl,\_,\_,\_,T1)''),$	$from([syl])]).$
$link(p2,$	$[def(''phon(Syl,\_,\_,\_,T2)''),$	$from([syl])]).$
$link(p3,$	$[def(''phon(Syl,\_,\_,A3,\_)''),$	$from([syl])]).$
$link(p4,$	$[def(''phon(Syl,\_,\_,A4,\_)''),$	$from([syl])]).$
$link(p5,$	$[def(''phon(Syl,\_,\_,A5,\_)''),$	$from([syl])]).$
$link(p6,$	$[def(''phon(Syl,\_,\_,A6,\_)''),$	$from([syl])]).$
$link(p7,$	$[def(''phon(Syl,\_,\_,A7,\_)''),$	$from([syl])]).$
$lit(liquid,$	$[def([''T1 = liquid'']),$	$from([p1])]).$
$lit(nasal,$	$[def([''T2 = nasal'']),$	$from([p2])]).$
$lit(l,$	$[def([''A3 = l'']),$	$from([p3]),$
	$requires([liquid])]).$	
$lit('R',$	$[def([''A4 = `R'']),$	$from([p4]),$
	$requires([liquid])]).$	
$lit('N',$	$[def([''A5 = `N'']),$	$from([p5]),$
	$requires([nasal])]).$	
$lit(n,$	$[def([''A6 = n'']),$	$from([p6]),$
	$requires([nasal])]).$	
$lit(m,$	$[def([''A7 = m'']),$	$from([p7]),$
	$requires([nasal])]).$	

5.2.1 Kombinierende Taxonomie

Hier werden interessante Kombinationen von Phonemtypen in Silben gesucht. Da eine Silbe mehrere Konsonsanten enthalten kann, wird die Taxonomie kombinierend deklariert. Die Relation $phon(P, S, A, Typ)$ hat hier ein anderes Format als in den Beispielen in Abschnitt 4.3, weil sie neben dem Ausprägung jedes Phonems auch dessen übergeordneten Typ repräsentiert, also den Vorgänger des Phonemtyps in der Taxonomie. Das Attribut P beschreibt der Identifikator (Primärschlüssel) der Relation $phon$, S die Silbe, zu der das Phonem gehört, A die Ausprägung ($l,' R', n, m$ oder $'N'$) und T den Typ des Phonems, $liquid$ oder $nasal$. Wenn das Attribut A einen der Werte l oder $'R'$ aufweist, muß das Attribut T immer den Wert $liquid$ haben, und wenn A einen der Werte $n, m,' N'$ annimmt, muß das Attribut T den Wert $nasal$ aufweisen. Die Relation ist also nicht in Normalform. Dies ist eine Möglichkeit, übergeordnete Typen von Attributwerten zu repäsentieren; eine alternative Darstellungsform wird im nächsten Beispiel gewählt. Die Deklarationen für den Suchraum sind in Tafel 5.4 wiedergegeben. Den resultierenden Suchraum zeigt Tafel 5.5.

5.2.2 Ausschließende Taxonomie

In dieser Beispielanwendung wird nicht über Silben, sondern über Phoneme gelernt. Da jedes Phonem genau eine Ausprägung und einen Typ aufweist, müssen sich hier die Merkmale, die verschiedene Typen und Ausprägungen betreffen, gegenseitig ausschließen. Es liegt hier also eine ausschließende Taxonomie vor. Die Taxonomie ist hier repräsentiert in Form der Prädikate $liquid/1$ und $nasal/1$, die im Hintergrundwissen der Anwendung

Tafel 5.5 Der Suchraum zur Suche mit kombinierender Taxonomie.

Hypothese	Logikrepräsentation
1. Suchebene	
liquid.	$T1 = liquid$, $syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, \_, T1)$.
nasal.	$T2 = nasal$, $syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, \_, T2)$.
2. Suchebene	
liquid,nasal.	$T1 = liquid, T2 = nasal, syl(Syl, \_, Acc), phon(Syl, \_, \_, \_, T2)$, $phon(Syl, \_, \_, \_, T1)$.
liquid,l.	$T1 = liquid, A3 = l, syl(Syl, \_, Acc), phon(Syl, \_, \_, A3, \_)$, $phon(Syl, \_, \_, \_, T1)$.
liquid,R.	$T1 = liquid, A4 = \text{`}R\text{`}, syl(Syl, \_, Acc), phon(Syl, \_, \_, A4, \_)$, $phon(Syl, \_, \_, \_, T1)$.
nasal,N.	$T2 = nasal, A5 = \text{`}N\text{`}, syl(Syl, \_, Acc), phon(Syl, \_, \_, A5, \_)$, $phon(Syl, \_, \_, \_, T2)$.
nasal,n.	$T2 = nasal, A6 = n, syl(Syl, \_, Acc), phon(Syl, \_, \_, A6, \_)$, $phon(Syl, \_, \_, \_, T2)$.
nasal,m.	$T2 = nasal, A7 = m, syl(Syl, \_, Acc), phon(Syl, \_, \_, A7, \_)$, $phon(Syl, \_, \_, \_, T2)$.
3. Suchebene	
liquid,nasal,l.	$T1 = liquid, T2 = nasal, A3 = l, syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, A3, \_), phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1)$.
liquid,nasal,R.	$T1 = liquid, T2 = nasal, A4 = \text{`}R\text{`}, syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, A4, \_), phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1)$.
liquid,nasal,N.	$T1 = liquid, T2 = nasal, A5 = \text{`}N\text{`}, syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, A5, \_), phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1)$.
liquid,nasal,n.	$T1 = liquid, T2 = nasal, A6 = n, syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, A6, \_), phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1)$.
liquid,nasal,m.	$T1 = liquid, T2 = nasal, A7 = m, syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, A7, \_), phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1)$.
liquid,l,R.	$T1 = liquid, A3 = l, A4 = \text{`}R\text{`}, syl(Syl, \_, Acc)$, $phon(Syl, \_, \_, A4, \_)$,$phon(Syl, \_, \_, A3, \_), phon(Syl, \_, \_, \_, T1)$.
...	
7. Suchebene	
liquid,nasal,l,R,N,n,m.	$T1 = liquid, T2 = nasal, A3 = l, A4 = \text{`}R\text{`}, A5 = \text{`}N\text{`}$, $A6 = n, A7 = m, syl(Syl, \_, Acc), phon(Syl, \_, \_, A7, \_)$, $phon(Syl, \_, \_, A6, \_), phon(Syl, \_, \_, A5, \_), phon(Syl, \_, \_, A4, \_)$, $phon(Syl, \_, \_, A3, \_), phon(Syl, \_, \_, \_, T2), phon(Syl, \_, \_, \_, T1)$.

extensional definiert sind. Die Deklarationen für den Suchraum sind in Tafel 5.6 wiedergegeben. Den resultierenden Suchraum zeigt Tafel 5.7.

Tafel 5.6 Deklaration des Suchraums für die Suche interessanter Phoneme mit ausschließender Taxonomie.

$liquid(r).$ $liquid(\text{`}R\text{`}).$

$nasal('N').$ $nasal(n).$ $nasal(m).$

$link(phon,$ $[def(''phon(\_, P, C, A, \_)'')]).$

$lit(liquid,$ $[def(['' liquid(A)''])],$ $from([phon]),$
$excludes([nasal])]).$

$lit(nasal,$ $[def(['' nasal(A)''])],$ $from([phon]),$
$excludes([liquid])]).$

$lit(l,$ $[def(['' A = l''])],$ $from([phon]),$
$requires([liquid]),$ $excludes([R])]).$

$lit(R,$ $[def(['' A = 'R''])],$ $from([phon]),$
$requires([liquid]),$ $excludes([l])]).$

$lit(N,$ $[def(['' A = 'N''])],$ $from([phon]),$
$requires([nasal]),$ $excludes([n, m])]).$

$lit(n,$ $[def(['' A = n''])],$ $from([phon]),$
$requires([nasal]),$ $excludes([N, m])]).$

$lit(m,$ $[def(['' A = m''])],$ $from([phon]),$
$requires([nasal]),$ $excludes([n, N])]).$

Tafel 5.7 Der Suchraum zur Suche mit ausschließender Taxonomie.

Hypothese	Logik-Repräsentation
1. Suchebene	
liquid.	$liquid(A), phon(\_, P, C, A, \_).$
nasal.	$nasal(A), phon(\_, P, C, A, \_).$
2. Suchebene	
liquid,l.	$liquid(A), A = l, phon(\_, P, C, A, \_).$
liquid,R.	$liquid(A), A =' R', phon(\_, P, C, A, \_).$
nasal,N.	$nasal(A), A =' N', phon(\_, P, C, A, \_).$
nasal,n.	$nasal(A), A = n, phon(\_, P, C, A, \_).$
nasal,m.	$nasal(A), A = m, phon(\_, P, C, A, \_).$

5.2.3 Numerische Attribute

Attribute mit numerischem Wertebereich werden im Data Mining und im Maschinellen Lernen häufig diskretisiert. Dabei wird ihr Wertebereich in Intervalle eingeteilt und die Zugehörigkeit zu einem Intervall als Attributwert behandelt (zum Beispiel [SA95a, Klö99]). Viele Systeme diskretisieren die numerischen Attribute, indem sie vor der Anwendung des eigentlichen Entdeckungs- oder Lernverfahrens feste Intervallgrenzen berechnen und diese für den gesamten Entdeckungs- oder Lernprozeß beibehalten. Einen Überblick über ge-

Abbildung 5.3 Ein Beispiel für Allgemeinerbeziehungen zwischen Intervallen. Das numerische Attribut läuft von 0 bis 20.

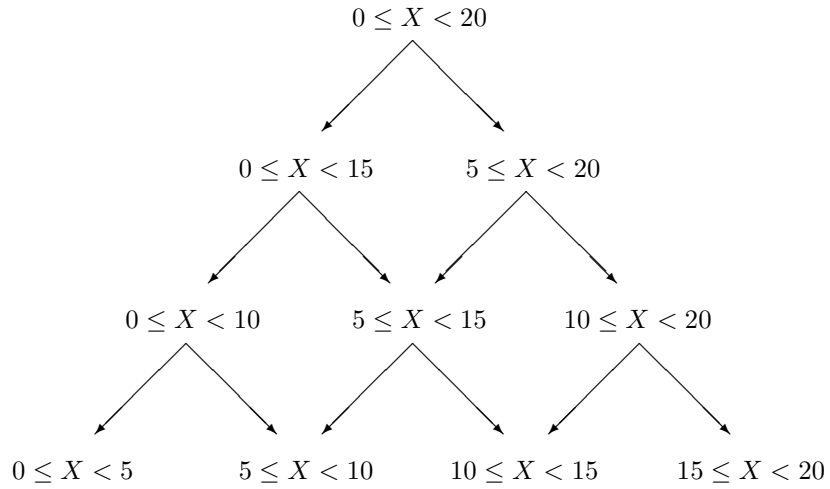
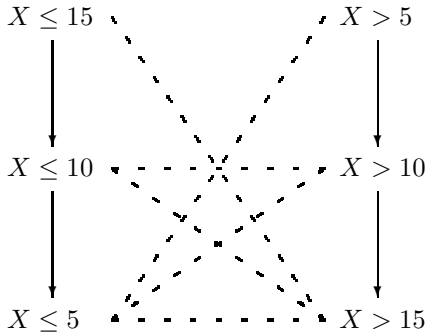


Abbildung 5.4 Allgemeiner- und Ausschlußbeziehungen zwischen Intervallgrenzen für numerische Attribute. Eine durchgezogene Linie symbolisiert eine Allgemeiner- und damit eine *req*-Beziehung zwischen den betreffenden Merkmalen; eine unterbrochene Linien steht für eine *excl*-Beziehung. Die Abbildung stellt eine alternative Repräsentation für die Taxonomie in Abbildung 5.3 dar.



bräuchliche Diskretisierungsmethoden geben [DKS95]. Die Intervalle lassen sich, wie in Abbildung 5.3 am Beispiel gezeigt, hierarchisch anordnen. Die Pfeile stellen dabei Allgemeinerbeziehungen zwischen den Intervallen dar. Die Intervalle schließen sich paarweise gegenseitig aus. Die Ausschlußbeziehungen sind in Abbildung 5.3 wegen der Übersichtlichkeit nicht eingezeichnet. Die Allgemeinerbeziehungen zwischen den Intervallen in Abbildung 5.3 weichen von einer Taxonomie nach Definition 36 darin ab, daß hier ein Intervall zwei direkte Vorgänger haben kann.

Ein Repräsentationswechsel ergibt eine Taxonomie, die der Definition 36 entspricht. Statt der Zugehörigkeit zu einem Intervall beschreibt ein Merkmal dabei einen Vergleich mit einer Intervallgrenze. Abbildung 5.4 zeigt diese alternative Repräsentation für die Hierarchie in Abbildung 5.3. Die durchgezogenen Pfeile in Abbildung 5.4 stellen Allgemeinerbeziehungen zwischen Merkmalen dar, die unterbrochenen Linien stehen für gegenseitigen Ausschluß zwischen den Merkmalen. Gemäß Bedingung 1 zur Erzeugung der Genex-Form werden die Allgemeinerbeziehungen auch hier in *req*-Bedingungen umgesetzt, der gegenseitige Ausschluß wird wieder durch *excl*-Bedingungen gemäß Bedingung 3 deklariert. Dann kombiniert der Spezialisierungsoperator des Suchalgorithmus die Intervallgrenzen

Tafel 5.8 Deklaration des Suchraums mit einem wie in Abbildung 5.4 diskretisierten numerischen Attribut.

$link(syl, \quad [def(''syl(Syl, L, Acc)'')]).$

$lit(l15, \quad [def(['' L \leq 15'']), \quad from([syl]), \quad excludes([g15])]).$

$lit(g5, \quad [def(['' L > 5'']), \quad from([syl]), \quad excludes([l5])]).$

$lit(l10, \quad [def(['' L \leq 10'']), \quad requires([liquid]), \quad from([syl]), \quad excludes([g15, g10])]).$

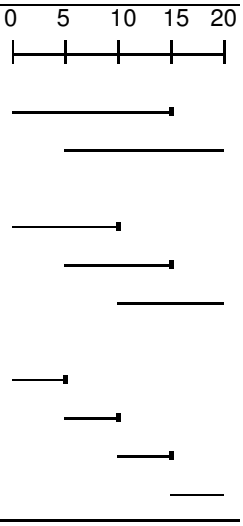
$lit(g10, \quad [def(['' L > 10'']), \quad requires([g5]), \quad from([syl]), \quad excludes([l10, l5])]).$

$lit(l5, \quad [def(['' L \leq 5'']), \quad requires([nasal]), \quad from([syl]), \quad excludes([g15, g10, g5])]).$

$lit(g15, \quad [def(['' L > 15'']), \quad requires([g10]), \quad from([syl]), \quad excludes([l5, l10, l15])]).$

Tafel 5.9 Der Suchraum zur Suche mit Intervallgrenzen.

Hypothese	Definition
1. Suchebene	
$\{l15\}$	$syl(Syl, X, Acc), L \leq 15$
$\{g5\}$	$syl(Syl, L, Acc), L > 5$
2. Suchebene	
$\{l15, l10\}$	$syl(Syl, L, Acc), L \leq 15, L \leq 10$
$\{l15, g5\}$	$syl(Syl, L, Acc), L \leq 15, L > 5$
$\{g5, g10\}$	$syl(Syl, L, Acc), L > 5, L > 10$
3. Suchebene	
$\{l15, l10, l15\}$	$syl(Syl, L, Acc), L \leq 15, L \leq 10, L \leq 5$
$\{l15, l10, g5\}$	$syl(Syl, L, Acc), L \leq 15, L \leq 10, L > 5$
$\{l15, g5, g10\}$	$syl(Syl, L, Acc), L \leq 15, L > 5, L > 10$
$\{g5, g10, g15\}$	$syl(Syl, L, Acc), L > 5, L > 10, L > 15$



zu den Intervallen, die in Abbildung 5.3 dargestellt sind.

Die Individuenpopulation dieses Anwendungsbeispiels besteht aus Silben. Das Attribut L soll die (normierte) Dauer einer Silbe angeben und ist wie in Abbildung 5.4 diskretisiert. Für ein kontinuierliches numerisches Attribut L mit Wertebereich $[1, 20]$ realisieren die Merkmalsdeklarationen eine Suche durch einen Raum von Intervallen abnehmender Länge. Tafel 5.8 zeigt die Deklaration dieses Suchraums, Tafel 5.9 illustriert den resultierenden Suchraum.

5.3 Verwandte Arbeiten

Da der beschriebene Algorithmus auf der Apriori-Hypothesengenerierung aufbaut, ist es aufschlußreich, ihn mit anderen Algorithmen zu vergleichen, die als Erweiterungen des Apriori-Algorithmus für allgemeinere Aufgabenstellungen entwickelt wurden. Assoziationsregeln, die auf den Artikeln definierte Taxonomien einbeziehen, bezeichnet man als generalisierte Assoziationsregeln und Assoziationsregeln über numerische Attribute als quantitative Assoziationsregeln [SA95a, SA96].

Der folgende Überblick vergleicht Hypothesenerzeugung und Suchraumbeschränkung einiger Apriori-basierter Algorithmen zum Finden generalisierter und quantitativer Assoziationsregeln, und analysiert, wie diese Algorithmen den Suchraum strukturieren und die Struktur des Suchraums zur Beschränkung des Suchraums nutzen. Für einige Algorithmen werden grobe Abschätzungen für die Anzahl der im schlechtesten Fall nötigen Datenbank-Durchläufe abgeleitet. Diese Worst-Case-Abschätzungen geben einen Hinweis auf die Feinheit der Ebeneneinteilung, die der jeweilige Algorithmus erzeugt. Der Überblick erhebt keinen Anspruch auf Vollständigkeit.

5.3.1 Algorithmen zum Finden generalisierter Assoziationsregeln

Basic, Cumulate und Stratify

Srikant und Agrawal haben in [SA95a] das Finden generalisierter Asssoziationsregeln als neues Data Mining-Problem eingeführt und eine Reihe von Algorithmen zu seiner Bearbeitung entwickelt und experimentell verglichen. Die Algorithmen sind Varianten des ursprünglichen Apriori-Algorithmus.

Basic. Die einfachste Variante, Basic, transformiert die Datenbank, indem sie jede Transaktion um die generalisierten Artikel erweitert. Das heißt, daß sie für jeden Artikel in einer Transaktion seine Vorgänger aus der Taxonomie bestimmt und der Transaktion zufügt. Der Apriori-Algorithmus läuft dann ohne Änderungen auf der erweiterten Datenbank und behandelt dabei alle Artikel gleich, ohne ihre Taxonomie-Ebene zu berücksichtigen. Damit nützt der Algorithmus die Taxonomie-Information überhaupt nicht zum Beschränken des Suchraums, sondern erzeugt und evaluiert eine große Anzahl von Artikelmengen, die redundant dargestellt sind in der Hinsicht, daß sie Artikel und deren Verallgemeinerungen zugleich enthalten. Der Algorithmus erzeugt und verarbeitet auch alle (häufigen) Teilmengen dieser redundant repräsentierten Artikelmengen, also auch diejenigen, die zwar die verallgemeinerten, aber nicht alle spezielleren Artikel enthalten. Damit sind nicht nur einzelne Artikelmengen redundant repräsentiert, sondern auch der Hypothesenraum enthält Redundanz in Form von Artikelmengen, die unterschiedlich dargestellt sind, aber äquivalente Abdeckungen haben.

Wenn n die Anzahl der Artikel der untersten Taxonomie-Ebene bezeichnet und g die Anzahl generalisierter Artikel, kann der Basic-Algorithmus im schlechtesten Fall Artikelmengen der Länge $n + g$ erzeugen und evaluieren. Diese maximale Länge entspricht auch der im schlechtesten Fall benötigten Anzahl von Datenbank-Durchläufen.

Cumulate. Cumulate ist eine Verbesserung von Basic, die die Redundanz reduziert, indem sie die redundant repräsentierten Artikelmengen ausfiltert; das sind Artikelmengen, die Artikel und ihre Vorgänger gemeinsam enthalten. Dadurch wird die schlechtestmögliche maximale Länge von Artikelmengen und die schlechtestmögliche Anzahl von Datenbank-Durchläufen auf die Anzahl der Artikel der untersten Taxonomie-Ebene n begrenzt. Wie Basic nützt Cumulate die Taxonomie-Information aber nicht zur Beschränkung des Suchraums.

Abbildung 5.5 Beispieltaxonomie zur Diskussion des Stratify-Algorithmus.

Stratify. Die nächste Algorithmus-Erweiterung, Stratify, verwendet die durch die Taxonomie bereitgestellte Information zur Beschränkung des Suchraums. Dazu bestimmt Stratify auf jeder Suchebene k zunächst alle Artikelmengen der Länge k, wobei er genau wie Cumulate redundant repräsentierte Artikelmengen ausschließt. Dies ergibt wie im ursprünglichen Apriori-Algorithmus alle zur Suchebene k gehörenden Hypothesen. Diese werden nun jedoch nicht in einem gemeinsamen Datenbank-Durchlauf ausgewertet, sondern nach ihrer Tiefe bezüglich der Taxonomie sortiert.

Srikant und Agrawal definieren die Tiefe einer Artikelmenge bezüglich einer Taxonomie relativ zu einer Menge K von Artikelmengen, die alle die gleiche Länge k haben. Eine Artikelmenge I hat die Tiefe 0, wenn K keine Vorgänger-Artikelmenge von I enthält. Wenn K Vorgänger von I enthält, ist die Tiefe einer Artikelmenge I in K gleich 1 + die maximale Tiefe der in K enthaltenen Vorgänger von I.

Beginnend mit den Artikelmengen minimaler Tiefe 0, evaluiert Stratify nur Artikelmengen derselben Tiefe in einem gemeinsamen Datenbank-Durchlauf und kappt nicht-häufige Artikelmengen, so daß deren Spezialisierungen bezüglich der Taxonomie in den folgenden Datenbank-Durchläufen nicht ausgewertet werden.

Der reine Stratify-Algorithmus nützt also die Taxonomie-Informationen bestmöglich zur Reduktion des Suchraums, benötigt dazu jedoch für jede Suchebene viele zusätzliche Datenbank-Durchläufe. Besonders ungünstig sind für Stratify degenerierte Taxonomien, in denen ein Knoten höchstens einen Nachfolger hat. Beispielsweise werden für die in Abbildung 5.5 wiedergegebene Taxonomie auf der 3. Suchebene im schlechtesten Fall 9 Datenbank-Durchläufe nötig [Web98b]. Um den Mehraufwand für wiederholte Datenbank-Durchläufe zu begrenzen, faßt Stratify Artikelmengen verschiedener Tiefe dann zu einem gemeinsamen Datenbank-Durchlauf zusammen, wenn ihre Anzahl eine bestimmte Grenze unterschreitet.

Weitere Verbesserungen von Stratify sind Estimate and EstMerge, die Sampling zur Reduktion des Suchaufwands einsetzen [SA95a]. Sie werden hier nicht analysiert.

MLT

Die Algorithmen der MLT–Familie [HF95a] wurden für eine etwas andere Aufgabenstellung entwickelt als die Apriori-Algorithmen. Die MLT–Algorithmen suchen Assoziationsregeln, in denen nur Artikel derselben Tiefe vorkommen, und die Support-Grenzwerte für verschiedene Tiefen sind unterschiedlich. Die Tiefe eines Artikels, der in der Taxonomie keine Vorgänger besitzt, ist 1. Die Tiefe der übrigen Artikel ist 1 plus die Tiefe ihres direkten Vorgängers.

Entsprechend der Aufgabenstellung sortiert MLT die Artikel nach ihrer Tiefe bezüglich der Taxonomie und bestimmt zuerst die häufigen Artikelmengen aus Artikeln der Tiefe 1, dann der Tiefe 2 und so fort. Die häufigen Artikelmengen jeder Tiefe werden mit der Apriori-Hypothesengenerierung ermittelt; dabei werden die Nachfolger-Artikel nichthäufiger Artikel der vorhergehenden Tiefe von der Suche ausgeschlossen. Der Algorithmus ist also so in Phasen organisiert, daß er in der i-ten Phase häufige Artikelmengen der Tiefe i

ermittelt, wobei Artikelmengen der Tiefe i nur Artikel der Tiefe i enthalten. Innerhalb jeder Phase führt der Algorithmus eine Apriori-artige ebenenweise Suche durch und verwendet die Teilmengenbedingung, um die Suche zu begrenzen.

Der Algorithmus läßt sich für die Suche nach generalisierten Assoziationsregeln nach der Definition von [SA95a] adaptieren, die Assoziationen aus Artikeln unterschiedlicher Tiefe zuläßt. Der adaptierte Algorithmus bestimmt in der i-ten Phase die häufigen Artikelmengen der Tiefe i, wobei jetzt die Tiefe einer Artikelmenge gleich der maximalen Tiefe ihrer Artikel ist. Dieses Vorgehen ist gewissermaßen komplementär zur Strategie von Stratify. Während Stratify zuerst Artikelmengen der Länge k (in Reihenfolge steigender k) erzeugt und diese zu Gruppen derselben Tiefe zusammengefaßt evaluiert, erzeugt MLT zuerst Artikelmengen derselben Tiefe (ebenfalls in der Reihenfolge zunehmender Tiefe) und faßt sie zur Evaluation zu Gruppen derselben Länge zusammen.

Die von den MLT–Algorithmen implizit verwendete Definition der Tiefe einer Artikelmenge ist schwächer als die Definition, die der Stratify-Algorithmus annmimmt. Entsprechend können die MLT–Algorithmen die Taxonomie-Information nur teilweise zur Suchraumbeschränkung nutzen. Wenn zum Beispiel für drei Artikel i_1, i_2, i_3 gilt, daß i_1 ein Vorgänger von i_3 ist und i_2 und i_3 beide die Tiefe 2 haben, so haben nach der MLT–Definition die Artikelmengen $\{i_1, i_2\}$ und $\{i_2, i_3\}$ beide die Tiefe 2 und werden im selben Datenbankdurchlauf ausgewertet. Die Allgemeinerbeziehung zwischen $\{i_1, i_2\}$ und $\{i_2, i_3\}$, die sich aus der Allgemeinerbeziehung zwischen i_1 und i_3 ergibt, kann nicht zur Suchraumbeschränkung genutzt werden.

Prutax

Der Algorithmus Prutax [HMWG98] dient ebenfalls zum Finden häufiger Assoziationen mit generalisierten Artikeln in Transaktionsdatenbanken. Anders als die vorher beschriebenen Algorithmen verwendet PRUTAX als Suchstrategie keine Breitensuche, sondern eine Tiefensuche. Jede Artikelmenge wird unmittelbar nach ihrer Erzeugung auf der Datenbank ausgewertet. Aus den spezialisierbaren Artikelmengen erzeugt der Spezialisierungsoperator neue Hypothesen. Die Reihenfolge, in der die Artikelmengen erzeugt werden, ist so gewählt, daß eine Artikelmenge erst generiert wird, wenn alle ihre Teilmengen und alle ihre Vorgänger-Artikelmengen bezüglich der Taxonomie bereits erzeugt und ausgewertet sind. Prutax kann so trotz Tiefensuche die Suche mithilfe der Teilmengenbeziehung zwischen Artikelmengen begrenzen und auch die Taxonomie-Information vollständig zur Suchbegrenzung verwerten.

Um Unterschied zu den vorher beschriebenen Algorithmen führt PRUTAX jedoch keine ebenenweise Suche durch, bei der mehrere Hypothesen in einem gemeinsamen Datenbank-Durchlauf ausgewertet werden, sondern evaluiert die Hypothesen einzeln.

5.3.2 Algorithmen zum Finden quantitativer Assoziationsregeln

QAR

Srikants und Agrawals Algorithmus QAR ist der erste Algorithmus, der für das Finden quantitativer Assoziationsregeln entwickelt wurde [SA96]. QAR bestimmt in einem ersten Datenbank-Durchlauf alle häufigen Intervalle beliebiger Allgemeinheit für die quantitativen Attribute. In der zweiten Phase ermittelt er mit dem Apriori-Grundalgorithmus alle häufigen Artikelmengen. Dabei werden die häufigen Intervalle behandelt wie nichtnumerische Attribute mit dem Unterschied, daß QAR die Erzeugung von Artikelmengen unterdrückt, die mehr als ein Intervall für dasselbe numerische Attribut enthalten. Weitere Beziehungen zwischen den Intervallen verwendet der Algorithmus nicht.

Die maximal benötigte Anzahl von Datenbank-Durchläufen im schlechtesten Fall, also wenn der gesamte Suchraum durchlaufen werden muß, entspricht der Anzahl der Attribute.

Diese Anzahl an Durchläufen würde der Algorithmus auch benötigen, wenn die numerischen Attribute einfache Boole'sche Attribute wären; nur die Anzahl der Hypothesen pro Suchebene ist größer, weil mehrere Intervalle pro numerischem Attribut auszuwerten sind.

Q2

Der Algorithmus Q2 [BW98] ist anwendbar für Transaktionsdatenbanken mit diskreten numerischen (oder sonstigen ordinalen) Attributen. Er besteht aus zwei Phasen. In der ersten Phase ignoriert der Algorithmus die numerische Information, indem er die quantifizierten Artikel gleich wie nicht quantifizierte Boole'sche Artikel behandelt, und bestimmt mit dem Apriori-Grundalgorithmus alle häufigen Artikelmengen. Erst in der zweiten Phase berücksichtigt der Algorithmus auch die quantitative Information und erzeugt Spezialisierungen aus den häufigen Artikelmengen, indem er jedes numerische Attribut ersetzt durch jedes mögliche Intervall, das es für dieses numerische Attribut geben kann. Wenn beispielsweise in einer häufigen Artikelmenge $\{l_1, l_2, n_3\}$ n_3 ein ganzzahliges numerisches Attribut mit maximalem Wert 3 ist, erzeugt der Algorithmus die Spezialisierungen $\{l_1, l_2, n_3 \in [1,2]\}, \{l_1, l_2, n_3 \in [2,3]\}, \{l_1, l_2, n_3 \in [1]\}$, $\{l_1, l_2, n_3 \in [2]\}$ und $\{l_1, l_2, n_3 \in [3]\}$. Alle spezialisierten Artikelmengen werden anschließend in einem gemeinsamen Datenbank-Durchlauf ausgewertet.

Damit nützt Q2 einen Teil der Strukturinformation zur Beschränkung des Suchraums, indem er die numerischen Attribute nur in solchen Artikelmengen spezialisiert, die er mit nicht-eingeschränkten numerischen Attribut in der ersten Phase als häufig erkannt hat. Oder, anders gesagt, er verwendet, daß das allgemeinste nichteingeschränkte Intervall eine Verallgemeinerung aller eingeschränkten Intervalle darstellt, aber nicht die Allgemeinerbeziehungen zwischen den kleineren Intervallen. Q2 benötigt also einen Datenbank-Durchlauf, um die numerische Information auszuwerten, zusätzlich zu den Datenbank-Durchläufen, die der Apriori-Grundalgorithmus für die entsprechende Datenmenge ohne numerische Information benötigt.

5.4 Diskussion

Der Genex-Ansatz stellt eine Erweiterung des Apriori-Algorithmus für die Anwendung auf Datenbanken mit strukturierten oder ordinalen Attributen dar. Er strebt an, das Prinzip der ebenenweisen Suche beizubehalten und den Suchraum bei einer minimalen Anzahl zusätzlicher Suchebenen bestmöglich zu beschränken. Tatsächlich gelingt es mit der Genex-Repräsentation, alle deklarierten Allgemeinerbeziehungen zur Beschränkung des Suchraums zu nutzen. Von den oben beschriebenen Algorithmen gilt dies noch für Stratify und Prutax. Stratify benötigt aber, wenn er alle Allgemeinerbeziehungen beachtet, mehr Datenbank-Durchläufe, während Prutax auf die ebenenweise Suche verzichtet, die für bestimmte Datenbank-Layouts sehr günstig ist. Die übrigen der diskutierten Algorithmen verwenden die Allgemeinerbeziehungen nicht oder nur teilweise zur Suchbeschränkung.

Die Genex-Repräsentation bringt mit sich, daß die speziellste Hypothese im Hypothesenraum alle Merkmale enthalten kann, falls keine *excl*-Bedingungen vorliegen. Der Algorithmus benötigt dann im schlechtesten Fall (das ist beim vollständigen Durchsuchen des gesamten Suchraums) so viele Durchläufe wie Merkmale (Knoten und Blätter der Taxonomie) deklariert sind.

Die Genex-Repräsentation ist entgegengesetzt zum Vorgehen der oben diskutierten Apriori-Erweiterungen zum Finden generalisierter oder quantitativer Assoziationen. Während die meisten dieser Algorithmen Hypothesen verbieten, die Artikel (beziehungsweise Merkmale) und zugleich deren Vorgänger enthalten, ist dies charakteristisch für die Genex-Repräsentation. Die daraus resultierende Redundanz in der Repräsentation der einzelnen Hypothesen kann für manche Algorithmen und Anwendungskontexte nachteilig sein, denn bei der Auswertung der Hypothesen auf den Daten verursachen die redundanten Artikel

beziehungsweise Merkmale zusätzlichen Aufwand. Dessen Ausmaß hängt von den verwendeten Datenstrukturen ab. Zwar lassen sich die redundanten Artikel vor der Auswertung aus den Hypothesen eliminieren, aber auch die dadurch verursachten Kosten können zu hoch werden, zum Beispiel, wenn sehr viele Artikel und Assoziationen zwischen ihnen existieren.

Ein Vorteil der Genex-Repräsentation ist, daß sie über die Erweiterung des Grundalgorithmus um *req*- und *excl*-Bedingungen keine weiteren Modifikationen des Suchalgorithmus erfordert. Wie das Beispiel 5.2.1 zeigt, ist der Ansatz zur Behandlung von Allgemeinerbeziehungen zwischen Merkmalen und Hierarchien auf diskreten numerischen Attributen auch für multi-relationale Daten einsetzbar. Als ILP-Verfeinerungsoperator hat der Ansatz mit Midos gemein, die Behandlung strukturierter und diskreter numerischer Attribute in den Verfeinerungsoperator zu integrieren. Der hier entwickelte Ansatz behält dabei die ebenenweise Suche und die Beschränkung des Suchraums mittels der Teilmengenbeziehung bei.

Die *req*-Bedingungen, die die aus einer Taxonomie stammenden Allgemeinerbeziehungen in den Merkmalslsdeklarationen ausdrücken, sowie, im Falle einer ausschließenden Taxonomie, die entsprechenden *excl*-Bedingungen sind einfach automatisch zu generieren. Sie verursachen daher nur wenig zusätzlichen Aufwand bei der Deklaration des Suchraums.

Der hier vorgeschlagene Ansatz zur Behandlung von Taxonomien nützt also die Allgemeinerbeziehungen gut zur Beschränkung der Suche. Dies ist vorteilhaft in Anwendungen, in denen die Auswertung der Hypothesen auf der Datenbank teuer ist, beispielsweise, wenn sie durch Anfragen an relationale Datenbanken oder Logik-Interpreter erfolgt. Der Ansatz führt eine ebenenweise Suche durch und erfordert dabei wenige zusätzliche Datenbank-Durchläufe, um die Taxonomie auszuwerten. Dies ist zum Beispiel günstig für Daten, die in der in Abschnitt 2.3.3 erläuterten Interpretationen-Repräsentation vorliegen.

Kapitel 6

Wirksamkeit der Methoden zur Suchraumbeschränkung

Die Teilgruppenanalyse, die in Tafel 2.2 formalisiert ist, erfordert eine vollständige Suche im Hypothesenraum. Um diese auch für umfangreiche und komplexe Hypothesensprachen durchführen zu können, muß der Suchraum soweit wie möglich beschränkt werden. Die Teilmengenbedingung, Subsumtionsbeziehungen zwischen Merkmalen und Optimumschätzfunktionen erlauben eine sichere Beschränkung des Suchraums, bei der keine akzeptablen Hypothesen übersehen werden. Eine weitere Möglichkeit zur Verkleinerung des Suchraums, die sich jedoch auf die Menge der akzeptierten Hypothesen auswirkt, besteht darin, akzeptierte Hypothesen zu kappen und so die von ihnen subsumierten Hypothesen von der weiteren Suche auszuschließen. Die Wirksamkeit dieser Ansätze zur Aufwandsreduktion wird in diesem Kapitel anhand von Experimenten evaluiert und verglichen. Für die Experimente stehen zwei Datenbanken zur Verfügung. Es werden drei verschiedene Aufgabenstellungen definiert, die unterschiedliche Möglichkeiten zur Suchraumbeschränkung bieten. Für jede Datenbank und Aufgabenstellung werden Suchläufe durchgeführt, die die jeweils vorhandenen Möglichkeiten zur Suchraumbeschränkung in unterschiedlichem Maße nutzen.

6.1 Datenbanken und Suchräume

6.1.1 Die Schachdatenbank KRK

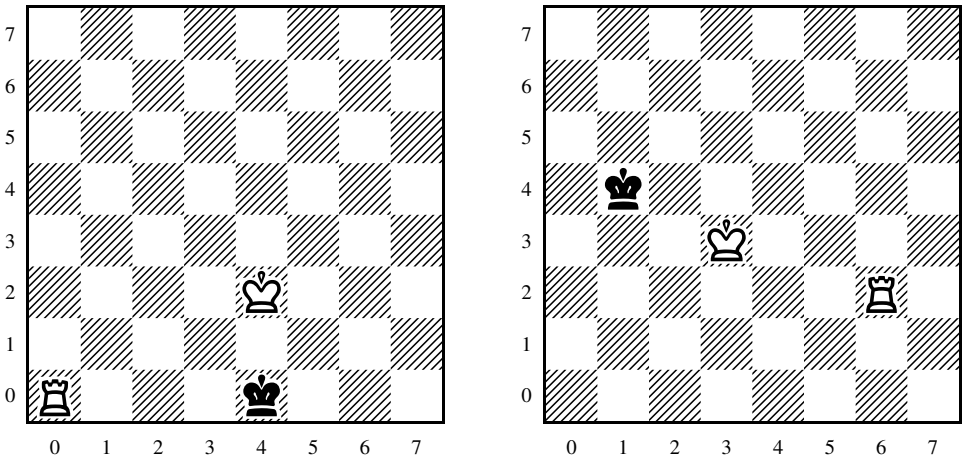
Daten

Die Schachdatenbank KRK[1] repräsentiert Spielstellungen des Schachendspiels, bei dem sich der weiße König (WK), der schwarze König (BK) und der weiße Turm (WR) im Spiel befinden. Weiß ist am Zug. Die Datenbank besteht aus 20 000 Tupeln einer Relation krk über dem Schema $krk(Id, C, \mathit{WKC}, \mathit{WKR}, \mathit{WRC}, \mathit{WRR}, \mathit{BKC}, \mathit{BKR})$. Die Attribute *WKC*, *WRC* und *BKC* geben die Spalte an, in der sich die betreffende Figur (WK, WR beziehungsweise BK) befindet. Die Attribute *WKR*, *WRR* und *BKR* beschreiben entsprechend die Reihen, in denen die Figuren stehen. Die Spalten wie die Reihen sind numeriert von 0 bis 7. Jedes der Attribute *WKC*, *WRC*, *BKC*, *WKR*, *WRR* und *BKC* hat also den Wertebereich $\{0, 1, 2, \ldots, 7\}$. Die Werte des Attributs Id sind Identifikatoren für die Spielstellungen. Das Attribut C ist binär und nimmt den Wert 1 an für unzulässige oder unmögliche Spielstellungen und den Wert 0 für zulässige Spielstellungen.

Beispiel 56 *In Abbildung 6.1 sind zwei Spielstellungen zu sehen. In der linken Spielstellung* $krk(id1, 1, 4, 2, 0, 0, 4, 0)$ *steht der schwarze König im Schach. Eine solche Spielstel-*

[1] http://www-users.cs.york.ac.uk/~stephen/chess.html

Abbildung 6.1 Beispiele für Schach-Spielstellungen: $krk(id1, 1, 4, 2, 0, 0, 4, 0)$ und $krk(id2, 0, 3, 3, 6, 2, 1, 4)$.



lung kann während eines Spiels nicht vorkommen, wenn Weiß am Zug ist. Deshalb ist diese Spielstellung unzulässig. Die Spielstellung $krk(id2, 0, 3, 3, 6, 2, 1, 4)$ *ist zulässig.*

Population und Zielgruppe

Die KRK-Datenbank wurde ursprünglich erstellt als Beispielanwendung für das Klassifikatorenlernen mit Klassifikationsaufgabe, unzulässige Spielstellungen von zulässigen Spielstellungen zu unterscheiden. Die Population besteht hier aus Spielstellungen. Für die folgenden Versuchsreihen bilden die unzulässigen Spielstellungen die Zielgruppe.

Suchraum

Üblicherweise wird für das Schach-Problem eine Sprache mit Hypothesen festgelegt, die Beziehungen zwischen Positionen der Spielfiguren auf dem Schachbrett darstellen (und nicht etwa absolute Positionen wie „schwarzer König steht in Spalte 1“). Die möglichen Beziehungen sind $=$, $\neq$, $>$, $<$ und adj (für engl. $adjacent$, benachbart). Definitionen dieser Beziehungen bilden das Hintergrundwissen der Anwendung. Da die Spalten- und die Reihenpositionen als Zahlen repräsentiert sind, läßt sich $adjac$ mit Prolog für Spalten und Reihen einheitlich definieren als

$$adj(Pos_1, Pos_2) \quad :- \quad N\ is\ Pos_1 - Pos_2, (N = 1; N = -1; N = 0).$$

Die Definition des Prädikats $adj/2$ kann direkt in die Definition der betreffenden Merkmale eingetragen werden. Für jedes drei Figurenpaare WK/BK, WK/WR und BK/WR werden Merkmale für jede der Beziehungen $=$, $\neq$, $>$, $<$ und adj zwischen den Spalten- und zwischen den Reihenpositionen deklariert. Dies ergibt insgesamt 30 Merkmale. Der einzige Verbinder für diese Anwendung ist krk.

Strukturierung des Suchraums ohne Taxonomie

Um widersprüchliche Hypothesen zu verhindern, schließen sich die Merkmale für die Beziehungen zwischen den Spaltenpositionen jedes Figurenpaares gegenseitig aus; genauso die Merkmale für die Beziehungen zwischen den Zeilenpositionen. Die Deklarationen für das Figurenpaar WK/WR sind in Tafel 6.1 zusammengestellt. Eine Hypothese kann maximal sechs Literale enthalten, und zwar für jedes der drei Figurenpaare eines oder keines

Tafel 6.1 Auszug aus den Deklarationen des Suchraums für die KRK-Datenbank ohne Taxonomie.

$count(pos, \quad [att(''Id''), \quad from([krk])]).$
$class(class, \quad [classes([legal -'' C = 0'', illegal -'' C = 1'']),$
$\quad targetclass(illegal), \quad from([krk])]).$

$link(krk, \quad [rel(''krk\_pos(Id, C, \mathbf{WKC}, \mathbf{WKR}, \mathbf{WRC}, \mathbf{WRR}, \mathbf{BKC}, \mathbf{BKR})'')]).$

$lit(wkc\_eq\_wrc, \quad [def([''\mathbf{WKC} = \mathbf{WRC}'']), \quad from([krk]),$
$\quad excludes([wkc\_neq\_wrc, wkc\_l\_wrc, wkc\_g\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_neq\_wrc, \quad [def([''\mathbf{WKC} \backslash == \mathbf{WRC}'']), \quad from([krk]),$
$\quad excludes([wkc\_eq\_wrc, wkc\_l\_wrc, wkc\_g\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_l\_wrc, \quad [def([''\mathbf{WKC} < \mathbf{WRC}'']), \quad from([krk]),$
$\quad excludes([wkc\_eq\_wrc, wkc\_neq\_wrc, wkc\_g\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_g\_wrc, \quad [def([''\mathbf{WKC} > \mathbf{WRC}'']), \quad from([krk]),$
$\quad excludes([wkc\_eq\_wrc, wkc\_neq\_wrc, wkc\_l\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_ad\_wrc, \quad [def([''adj(\mathbf{WKC}, \mathbf{WRC})'']), \quad from([krk]),$
$\quad excludes([wkc\_eq\_wrc, wkc\_neq\_wrc, wkc\_l\_wrc, wkc\_g\_wrc])]).$

$lit(wkr\_eq\_wrr, \quad [def([''\mathbf{WKR} = \mathbf{WRR}'']), \quad from([krk]),$
$\quad excludes([wkr\_neq\_wrr, wkr\_l\_wrr, wkr\_g\_wrr, wkr\_ad\_wrr])]).$
$lit(wkr\_neq\_wrr, \quad [def([''\mathbf{WKR} \backslash == \mathbf{WRR}'']), \quad from([krk]),$
$\quad excludes([wkr\_eq\_wrr, wkr\_l\_wrr, wkr\_g\_wrr, wkr\_ad\_wrr])]).$
$lit(wkr\_l\_wrr, \quad [def([''\mathbf{WKR} < \mathbf{WRR}'']), \quad from([krk]),$
$\quad excludes([wkr\_eq\_wrr, wkr\_neq\_wrr, wkr\_g\_wrr, wkr\_ad\_wrr])]).$
$lit(wkr\_g\_wrr, \quad [def([''\mathbf{WKR} > \mathbf{WRR}'']), \quad from([krk]),$
$\quad excludes([wkr\_eq\_wrr, wkr\_neq\_wrr, wkr\_l\_wrr, wkr\_ad\_wrr])]).$
$lit(wkr\_ad\_wrr, \quad [def([''adj(\mathbf{WKR}, \mathbf{WRR})'']), \quad from([krk]),$
$\quad excludes([wkr\_eq\_wrr, wkr\_neq\_wrr, wkr\_l\_wrr, wkr\_g\_wrr])]).$

aus den 5 Literalen, die die Beziehung zwischen den Zeilenpositionen des Figurenpaars beschreiben, sowie eines oder keines aus den 5 Literalen, die die Beziehung zwischen den Spaltenpositionen beschreiben. Der Suchraum enthält also $6^6 = 46656$ Hypothesen.

Strukturierung des Suchraums mit Taxonomie

Der Suchraum für das KRK-Problem kann weiter strukturiert werden durch folgende Subsumtionsbeziehungen zwischen den Merkmalen:

$adj(Pos_1, Pos_2)$ subsumiert $Pos_1 = Pos_2$
$Pos_1 \neq Pos_2$ subsumiert $Pos_1 < Pos_2$
$Pos_1 \neq Pos_2$ subsumiert $Pos_1 > Pos_2$

Die Taxonomie ist ausschließend. Eine Hypothese kann hier wegen der Genex-Repräsentation aus bis zu 12 Literalen bestehen. Die Merkmalsdeklarationen für das Figurenpaar

Tafel 6.2 Auszug aus den Deklarationen des Suchraums für die KRK-Datenbank mit Taxonomie.

$lit(wkc\_ad\_wrc,$	$[def([''adj(WKC, WRC)'']),$	$from([krk]),$
	$excludes([wkc\_neq\_wrc])]).$	
$lit(wkc\_neq\_wrc,$	$[def([''WKC \backslash == WRC'']),$	$from([krk]),$
	$excludes([wkc\_ad\_wrc])]).$	
$lit(wkc\_eq\_wrc,$	$[def([''WKC = WRC'']),$	$from([krk]),$
		$requires([wkc\_ad\_wrc])]).$
$lit(wkc\_l\_wrc,$	$[def([''WKC < WRC'']),$	$from([krk]),$
	$excludes([wkc\_g\_wrc]),$	$requires([wkc\_neq\_wrc])]).$
$lit(wkc\_g\_wrc,$	$[def([''WKC > WRC'']),$	$from([krk]),$
	$excludes([wkc\_l\_wrc]),$	$requires([wkc\_neq\_wrc])]).$
$lit(wkr\_ad\_wrr,$	$[def([''adj(WKR, WRR)'']),$	$from([krk]),$
	$excludes([wkr\_neq\_wrr])]).$	
$lit(wkr\_neq\_wrr,$	$[def([''WKR \backslash == WRR'']),$	$from([krk]),$
	$excludes([wkr\_ad\_wrr])]).$	
$lit(wkr\_eq\_wrr,$	$[def([''WKR = WRR'']),$	$from([krk]),$
		$requires([wkr\_ad\_wrr])]).$
$lit(wkr\_l\_wrr,$	$[def([''WKR < WRR'']),$	$from([krk]),$
	$excludes([wkr\_g\_wrr]),$	$requires([wkr\_neq\_wrr])]).$
$lit(wkr\_g\_wrr,$	$[def([''WKR > WRR'']),$	$from([krk]),$
	$excludes([wkr\_l\_wrr]),$	$requires([wkr\_neq\_wrr])]).$

WK/WR sind in Tafel 6.2 zu sehen.

6.1.2 Die Phonetik-Datenbank

In Abschnitt 2.3.3 wurde dargelegt, daß ein Problem dann nicht mehr sinnvoll mit aussagenlogischen Methoden zu bearbeiten ist, wenn in der Datenbank nichtdeterministische oder rekursive Verbindungen vorkommen. Die Phonetik-Datenbank weist solche Vebindungen auf, ist also eine echte prädikatenlogische Datenbank.

Daten

Die Phonetik-Datenbank ist eine wissenschaftliche Datenbank mit deutschen phonetischen Daten. Die Daten stammen aus Sprachaufnahmen, die für phonetische Untersuchungen gesammelt, analysiert und aufbereitet wurden [Rap98]. Die Datenbank besteht aus den drei Relationen *syl* mit 32 Attributen und 18109 Tupeln, *phon* mit 10 Attributen und 48093 Tupeln und *word* mit 7 Attributen und 7867 Tupeln. Die Tafeln 6.3 bis 6.5 geben einen Überblick über die Relationen der Phonetik-Datenbank.

Population und Zielgruppe

In den folgenden Experimenten bilden die Silben, identifiziert über den Schlüssel *Syl* der Relation *syl*, die Population der untersuchten Individuen. Die Silben mit Satzakzent bilden

Tafel 6.3 Attribute der Relation syl aus der Phonetik-Datenbank. Die mit $*$ markierten Attribute werden in den Experimenten verwendet.

Syl	$*$	Identifikator; Primärschlüssel
$Fnr, File$		technisch; Herkunft der Daten
$Accent$	$*$	1, wenn Satzakzent vorhanden, sonst 0
$Boundary$		Phrasengrenze
$Word$	$*$	Identifikator des Worts, zu dem die Silbe gehört
$Previous$		Syl der vorhergenden Silbe
$Next$		Syl der folgenden Silbe
$Lenexpected$	$*$	Summe der erwarteten Phonemlängen der Silbe
$Lenmeasure$	$*$	Gemessene Silbenlänge
$Lenrelative$		Quotient der vorhergehenden beiden Merkmale
$Wordstress$		Lexikoneintrag: 1 für wortbetonte Silben, sonst 0
$Syls2wordend$		Anzahl der Silben zum Wortende
$Syls2nextp$		Anzahl der Silben bis zur nächsten Pause
$Dist2nextp$	$*$	Abstand zur nächsten Pause in Sekunden
$Lenofnextp$		Länge der nächsten Pause in Sekunden
$Tonaldiff$		reell; Parameter α aus der F_0-Parametrisierung
$Tonalsteep$		reell; Parameter β aus der F_0-Parametrisierung
$Tonalalign$	$*$	reell; Parameter γ aus der F_0-Parametrisierung
$Peakheight$	$*$	reell; Parameter δ aus der F_0-Parametrisierung
$Peaksteep$		reell; Parameter ϵ aus der F_0-Parametrisierung
$Peakalign$		reell; Parameter ζ aus der F_0-Parametrisierung
$Level$		reell; Parameter η aus der F_0-Parametrisierung
rms_0_500		reell; Intensität zwischen 0 und 500 Hz
rms_500_1k		reell; Intensität zwischen 500 und 1000 Hz
rms_1k_2k		reell; Intensität zwischen 1000 und 2000 Hz
rms_2k_4k		reell; Intensität zwischen 2000 und 4000 Hz
rms_4k_8k		reell; Intensität zwischen 4000 und 8000 Hz
rms_0_8k		reell; Intensität zwischen 0 und 8000 Hz
rms_2k_8k		reell; Intensität zwischen 2000 und 8000 Hz

Tafel 6.4 Attribute der Relation $phon$ aus der Phonetik-Datenbank. Die mit $*$ markierten Attribute werden in den Experimenten verwendet.

$Phon$	$*$	Identifikator; Primärschlüssel
$Fnr, File$		technisch; Herkunft der Daten
Syl	$*$	Silbe, aus der das Phonem stammt
$Previous$		$Phon$ des vorhergehenden Phonems
$Next$		$Phon$ des nächsten Phonems
Onc		Silbenstruktur (Onset, Nucleus oder Coda)
$Phoneme$	$*$	Phonem; Werte siehe Tafel 6.6
$Vowcon$		Vokal oder Konsonant; Werte siehe Tafel 6.6
$Type$	$*$	Typ des Phonems; Werte siehe Tafel 6.6

Tafel 6.5 Attribute der Relation $word$ aus der Phonetik-Datenbank. Die mit $*$ markierten Attribute werden in den Experimenten verwendet.

$Word$	$*$	Identifikator; Primärschlüssel
$Fnr, File$		technisch; Herkunft der Daten
$Previous$		$Word$ des vorhergehenden Worts
$Next$		$Word$ des nächsten Worts
$Inlex$		binär; Lexikoneintrag vorhanden oder nicht
Tag	$*$	syntaktische Funktion des Worts; Werte siehe Tafel 6.7

Tafel 6.6 Phoneme und Phonemtypen.

Vowcon	Type	Phoneme	Erklärung
consonant	*frikativ*	*fc_C, fc_f, fc_h, fc_j, fc_s, fc_S, fc_v, fc_x, fc_z, fc_Z*	Reiblaute
	liquid	*lc_l, lc_R*	Gleitlaute
	nasal	*nc_m, nc_n, nc_N*	Nasale
	plosiv	*pc_b, pc_d, pc_g, pc_k, pc_p, pc_t*	Plosive
vowel	*diphtong*	*dv_aI, dv_aU, dv_OY*	Diphtonge
	short	*kv_9, kv_a, kv_E, kv_I, kv_O, kv_U, kv_Y*	kurze Vokale
	long	*lv_2colon, lv_acolon, lv_ecolon, lv_Ecolon, lv_icolon, lv_ocolon, lv_ucolon, lv_ycolon*	lange Vokale
	schwa		Schwa-Laut

die Zielgruppe. Die Deklaration von Population und Zielgruppe für die Experimente mit der Phonetik-Datenbank ist in Tafel 6.8 wiedergegeben.

Die Relationen syl, $phon$ und $word$ sind über Fremdschlüssel miteinander und rekursiv verbunden. Das Attribut Syl der Relation $phon$ ist der Schlüssel der Relation syl (vergleiche Tafeln 6.3 und 6.4). Die Beziehung zwischen den beiden Relationen ist eine $1 : m$–Beziehung, denn es kann zu jedem Wert von Syl mehrere $Phon$ geben. Das Attribut $Word$ der Relation *syl* ist der Schlüssel der Relation *Word* (siehe Tafel 6.5) und ordnet jede Silbe einem Wort zu. Auch diese Beziehung ist eine $1 : m$–Beziehung, da ein Wort mehrere Silben enthalten kann. Diese Beziehungen werden in den folgenden Experimenten verwendet. Die rekursiven Beziehungen der Relationen, die für jede Silbe, jedes Phonem und jedes Wort den Vorgänger und den Nachfolger angeben, werden dagegen nicht verwendet. Tafel 6.8 zeigt die Deklarationen von Verbindern für die Phonetik-Datenbank.

Der Suchraum

Die Datenbank erlaubt, sehr viele und komplexe Eigenschaften von Silben zu definieren. Um die Laufzeiten zu begrenzen, wird in den folgenden Experimenten nur eine Auswahl davon verwendet. Aus den Attributen der Relation $word$ werden Merkmale definiert, die das Tag des Worts abfragen, zu dem die Silbe gehört. Die Tags (Markierungen) geben die Funktion eines Wort im Satzzusammenhang an. Sie stammen aus dem Stuttgart-Tübingen-Tag-Set [STST99, Rap98]. Die möglichen Werte sind in Tafel 6.7 aufgelistet. Einschließlich der verallgemeinerten Tags p, v, n etc. werden hier 54 Tags unterschieden. Entsprechend gibt es 54 Merkmale, die sich auf Tags beziehen. Aus den Attributen der Relation $phon$ werden Merkmale abgeleitet, die Phoneme und Phonemtypen einer Silbe beschreiben. Tafel 6.6 zeigt die Wertebereiche der Attribute $Phoneme$, $Type$ und $Vowcon$ der Re-

Tafel 6.7 Tags und die Taxonomie auf Tags.

		Tag	Erklärung
	adj	*adja*	attributives Adjektiv
		adjd	adverbiales oder prädikatives Adjektiv
		adv	*Adverb*
	ap	*appr*	Präposition; Zirkumposition links
		apprart	Präposition mit Artikel
		art	bestimmter oder unbestimmter Artikel
		card	Kardinalzahl
	ko	*koui*	unterordnende Konjunktion mit „zu“ und Infinitiv
		kous	unterordnende Konjunktion mit Satz
		kon	nebenordnende Konjunktion
		kokom	Vergleichskonjunktion
	n	*nn*	normales Nomen
		ne	Eigennamen
p	*pd*	*pds*	substituierendes Demonstrativpronomen
		pdat	attribuierendes Demonstrativpronomen
	pi	*pis*	substituierendes Indefinitpronomen
		piat	attribuierendes Indefinitpronomen ohne Determiner
		pidat	attribuierendes Indefinitpronomen mit Determiner
		pper	irreflexives Personalpronomen
		pposat	attribuierendes Possessivpronomen
	prel	*prels*	substituierendes Relativpronomen
		prelat	attribuierendes Relativpronomen
		prf	reflexives Personalpronomen
	pw	*pws*	substituierendes Interrogativpronomen
		pwav	adverbiales Interrogativ- oder Relativpronomen
		pav	Pronomialadverb
	ptk	*ptkzu*	„zu“ vor Infinitiv
		ptkneg	Negationspartikel
		ptkvz	abgetrennter Verbzusatz
		ptka	Partikel bei Adjektiv oder Adverb
		trunc	Kompositions-Erstglied
v	*vv*	*vvfin*	finites Verb, voll
		vvinf	Infinitiv, voll
		vvizu	Infinitiv mit „zu“, voll
		vvpp	Partizip Perfekt, voll
	va	*vafin*	finites Verb, aux
		vainf	Infinitiv, aux
		vapp	Partizip Perfekt, aux
	vm	*vmfin*	finites Verb, modal
		vminf	Infinitiv, modal

Tafel 6.8 Deklaration von Verbindern, Population und Zielgruppe zur Phonetik-Datenbank.

$count(sylids, \quad [att(''Syl''), \quad from([syl])]).$
$class(accent, \quad [targetclass('acc'), \quad from([syl]),$
$\quad classes([\quad acc -'' Accent =' acc' '',$
$\quad no\_acc -'' Accent =' 0' ''])]).$

$link(syl, \quad [rel(''syl(Syl, Accent, \ldots, Peak\_height)'')]).$
$link(w, \quad [rel(''word(Word, \_, \_, \_, \_, Tag)''),$
$\quad from([syl])]).$
$link(avg, \quad [rel(''avg(P\_Avg, T\_Avg, D\_Avg)'')]).$
$link(t1, \quad [rel(''phon(Syl, \_, \_, \_, \_, \_, \_, \_, T1)''),$
$\quad from([syl])]).$
$link(t2, \quad [rel(''phon(Syl, \_, \_, \_, \_, \_, \_, \_, T2)''),$
$\quad from([syl])]).$
$\ldots$
$link(t8, \quad [rel(''phon(Syl, \_, \_, \_, \_, \_, \_, \_, T8)''),$
$\quad from([syl])]).$
$link(p1, \quad [rel(''phon(Syl, \_, \_, \_, \_, \_, \_, Phoneme1, \_, \_)''),$
$\quad from([syl])]).$
$link(p2, \quad [rel(''phon(Syl, \_, \_, \_, \_, \_, \_, Phoneme2, \_, \_)''),$
$\quad from([syl])]).$
$\ldots$
$link(p39, \quad [rel(''phon(Syl, \_, \_, \_, \_, \_, \_, Phoneme39, \_, \_)''),$
$\quad from([syl])]).$

lation $phon$. Für jedes Phonem und jeden Phonemtyp ist ein Merkmal definiert. Dies ergibt 47 Merkmale, die die Phoneme einer Silbe betreffen. Dazu kommen noch acht Merkmale, die gemessene Eigenschaften von Silben abfragen, die in der Relation syl gespeichert sind. Sechs dieser Merkmale vergleichen den Wert einer Silbe mit dem entsprechenden Durchschnittswert über alle Silben. Diese Durchschnittswerte sind als Hintergrundinformation in der Relation avg gespeichert, die aus einem einzigen Tupel $avg(P\_Avg, D\_Avg, T\_Avg)$ besteht.

Strukturierung des Suchraums ohne Taxonomie

Bei der Deklaration eines Suchraums für die Phonetik-Datenbank kann man das Anwendungswissen verwenden, daß jede Silbe zwar mehrere Konsonanten enthalten kann, jedoch nur einen Vokal. Demzufolge können alle Merkmale, die den Vokal einer Silbe beschreiben, als sich gegenseitig ausschließend deklariert werden. Da jede Silbe zu genau einem Wort gehört, schließen sich auch die Merkmale, die Tags betreffen, gegenseitig aus. Tafel 6.9 zeigt einen Ausschnitt aus den Deklarationen von Merkmalen für die Phonetik-Datenbank.

Tafel 6.9 Merkmale im Suchraum ohne Taxonomie zur Phonetik-Datenbank.

$lit(long\_syl,$	$[def([''Len\_expected \geq Len\_measure'']),$	$from([syl]),$
	$excludes([short\_syl])]).$	
$lit(short\_syl,$	$[def([''Len\_expected < Len\_measure'']),$	$from([syl]),$
	$excludes([long\_syl])]).$	
$lit(high\_sylpeak,$	$[def([''Peak\_height \geq P\_Avg'']),$	$from([syl, avg]),$
	$excludes([low\_sylpeak])]).$	
$lit(low\_sylpeak,$	$[def([''Peak\_height < P\_Avg'']),$	$from([syl, avg]),$
	$excludes([high\_sylpeak])]).$	
$lit(high\_tonalalign,$	$[def([''Tonal\_align \geq T\_Avg'']),$	$from([syl, avg]),$
	$excludes([low\_tonalalign])]).$	
$lit(low\_tonalalign,$	$[def([''Tonal\_align < T\_Avg'']),$	$from([syl, avg]),$
	$excludes([high\_tonalalign])]).$	
$lit(short\_dist2nextp,$	$[def([''Dist2nextp < D\_Avg'']),$	$from([syl, avg]),$
	$excludes([long\_dist2nextp])]).$	
$lit(long\_dist2nextp,$	$[def([''Dist2nextp \geq D\_Avg'']),$	$from([syl, avg]),$
	$excludes([short\_dist2nextp])]).$	
$lit(diphtong,$	$[def([''T1 = diphtong'']),$	$from([t1]),$
	$excludes([übrigeVokale])]).$	
$lit(frikativ,$	$[def([''T2 = frikativ'']),$	$from([t2]),$
	$excludes([übrigeVokale])]).$	
...		
$lit(dv\_aI,$	$[def([''Phoneme1 = dv\_aI'']),$	$from([p1]),$
	$excludes([übrigeVokale])]).$	
$lit(dv\_aU,$	$[def([''Phoneme2 = dv\_aU'']),$	$from([p2]),$
	$excludes([übrigeVokale])]).$	
$lit(dv\_OY,$	$[def([''Phoneme3 = dv\_OY'']),$	$from([p3]),$
	$excludes([übrigeVokale])]).$	
...		
$lit(pc\_k,$	$[def([''Phoneme37 = pc\_k'']),$	$from([p37])]).$
$lit(pc\_p$	$[def([''Phoneme38 = pc\_p'']),$	$from([p38])]).$
$lit(pc\_t,$	$[def([''Phoneme39 = pc\_t'']),$	$from([p39])]).$
...		
$lit(adj,$	$[def([''(Tag == adja; Tag == adjd)'']),$	$from([w]),$
	$excludes([übrigeTags])]).$	
$lit(adja,$	$[def([''Tag = adja'']),$	$from([w]),$
	$excludes([übrigeTags])]).$	
$lit(adjd,$	$[def([''Tag = adjd'']),$	$from([w]),$
	$excludes([übrigeTags])]).$	

Strukturierung des Suchraums mit Taxonomien

Der Suchraum kann weiter strukturiert werden mit den Taxonomien auf Phonemen und Tags, die in den Tafeln 6.6 und 6.7 dargestellt sind. Für die Konsonanten unter den Phonemen wird eine kombinierende, für Tags und Vokale eine ausschließende Taxonomie deklariert. Dazu werden die Merkmalsdeklarationen um *req*- und *excl*-Bedingungen ergänzt wie in Kapitel 5 beschrieben.

6.2 Aufbau der Experimente

6.2.1 Das verwendete System

Die Suchläufe wurden mit einer prototypischen Implementation des entwickelten Ansatzes durchgeführt. Das System besteht aus einer Client- und aus einer Server-Komponente. Der Client enthält den eigentlichen Suchalgorithmus, der in Tafel 4.2 und Abschnitt 4.1 beschrieben ist. Der Client liest eine Textdatei mit Deklarationen des Suchraums ein, deren Format in Abschnitt 4.2.2 beschrieben ist. Der Client generiert die Hypothesen und die Anfragen, die zur Auswertung der Hypothesen dienen, wie in Abschnitt 4.2.5 beschrieben. Die einzelnen Anfragen schickt der Client über eine TCP/IP-Verbindung an den Server, der die Anfragen an das jeweilige Datenbankmanagementsystem übergibt und ihre Resultate an den Client zurückliefert.

Der Client ist implementiert in C und Prolog, der Server in C. Zur Bearbeitung von Prolog-Anfragen benutzt der Server einen Quintus Prolog-Interpreter; zur Bearbeitung von SQL-Anfragen greift er auf ein IBM DB2 Datenbankmanagementsystem zu. Bei den Experimenten liefen der Client und der Server auf je einer Sun Ultra-1 mit 448 MByte Speicher.

6.2.2 Aufgabenstellungen

Suche nach Assoziationen: assoc

Die Aufgabenstellung *assoc* sucht Assoziationen von Merkmalen, die für die Zielgruppe typisch sind. Sie verwendet die Interessantheitskriterien Bestätigung und Genauigkeit. Akzeptiert werden alle Hypothesen, deren Bestätigung und Genauigkeit vorgegebene Schwellwerte erreichen oder überschreiten. Je höher der Schwellwert σ_a für die Bestätigung liegt, desto stärker kann die Suche beschränkt werden. Der Schwellwert wird in den verschiedenen Testläufen auf die Werte $\sigma_a = 0.1, 0.2, 0.3, 0.4$ festgelegt. Der Schwellwert für die Genauigkeit beeinflußt den Umfang der Lösungsmenge, jedoch nicht den Umfang des Suchraums. Er hat in allen Testläufen den Wert 0.8. Optimumschätzfunktionen existieren für diese Aufgabenstellung nicht. Akzeptierte Hypothesen werden nicht von der weiteren Suche ausgeschlossen.

Suche nach ungewöhnlichen Teilgruppen: midos

Die Aufgabenstellung midos findet Hypothesen mit hoher Verteilungsungewöhnlichkeit. Gesucht werden die 10 bezüglich der Verteilungsungewöhnlichkeit interessantesten Hypothesen, deren Häufigkeit eine untere Schwelle erreicht oder überschreitet. Zur Verteilungsungewöhnlichkeit existieren Optimumschätzfunktionen. Außer diesen beeinflußt der Häufigkeitsschwellwert das Ausmaß der Suchraumbeschränkung. In den verschiedenen Suchläufen wird der Häufigkeitsschwellwert auf $\sigma_m = 0.0001, 0.1, 0.2, 0.3$ festgelegt. Die Spezialisierungen akzeptierter Hypothesen verbleiben im Suchraum.

Suche nach Implikationen: impint

Die Interessantheitsfunktion der Aufgabenstellung impint ist die Folgerungsstärke. Akzeptiert werden allgemeinste Hypothesen, deren Folgerungsstärke und Häufigkeit vorgegebene

Schwellwerte erreichen. Spezialisierungen akzeptierter Hypothesen werden aus dem Suchraum ausgeschlossen. Der Häufigkeitsschwellwert ist für alle Testläufe auf $\sigma_i = 0.005$ fixiert. Der Parameter α, der gleich 1 minus der Folgerungstärke ist, gibt das für Akzeptanz maximal erlaubte Irrtumsrisiko an. Er variiert in den Testläufen über die Werte $\alpha = 0.0001, 0.005, 0.01, 0.05$. In dieser Aufgabenstellung erlauben der Häufigkeitsschwellwert und eine Optimumschätzfunktion zur Folgerungsstärke die Beschränkung des Suchraums. Die Optimumschätzfunktion greift desto besser, je schwieriger das Akzeptanzkriterium zu erfüllen ist. Zugleich führt das Kappen akzeptierter Hypothesen dazu, daß der Suchraum desto stärker beschränkt wird, je leichter das Akzeptanzkriterium zu erfüllen ist.

6.2.3 Versuchsreihen

Eine erste Reihe von Suchläufen (all) nutzt alle Möglichkeiten zur Beschränkung des Suchraums, also die vorhandenen Optimumschätzfunktionen, die Teilmengenbedingung und Subsumtionsbeziehungen zwischen Merkmalen (Taxonomien). Die Versuchsreihe all dient als Referenz, mit der die anderen Versuchsreihen verglichen werden. Für alle Suchläufe wird die Anzahl der Hypothesen bestimmt, die ohne Verwendung der Teilmengenbedingung auszuwerten wäre. Die Reihe dieser Anzahlen aus der ersten Versuchsreihe all wird in der Diskussion der Ergebnisse als ohne_tm aufgeführt. Die zweite Versuchsreihe ohne_tax verwendet Optimumschätzfunktionen und die Teilmengenbedingung, durchsucht aber einen Suchraum ohne Taxonomie-Information. Die dritte Versuchsreihe wird zweifach ausgeführt (ohne_oe und schwach_oe). Dabei werden Taxonomie-Information und Teilmengenbedingung verwendet, aber keine beziehungsweise nur abgeschwächte Optimumschätzfunktionen eingesetzt. Eine letzte Versuchsreihe ohne_aka besteht aus Suchläufen in einer modifizierten Aufgabenstellung impint, bei der die von akzeptierten Hypothesen subsumierten Hypothese im Suchraum verbleiben. Die ersten beiden Versuchsreihen all/ohne_tm und ohne_tax werden für alle drei Aufgabenstellungen assoc, midos und impint durchgeführt. Die dritte Versuchsreihe ohne_oe/schwach_oe findet nur für die Aufgabenstellungen midos und impint statt, da für die Aufgabenstellung assoc keine Optimumschätzfunktion vorliegt. Die vierte Versuchsreihe ohne_aka betrifft nur die Aufgabenstellung impint. In allen Suchläufen ist die maximale Suchtiefe auf 13 Suchebenen begrenzt.

Insgesamt werden für jede der beiden Datenbanken und mit den genannten Parametereinstellungen die Anzahl der Hypothesen ermittelt, die unter den folgenden Bedingungungen per Datenbankanfrage auszuwerten sind:

> all: mit Teilmengenbedingung, Taxonomie, Optimumschätzfunktionen und Akzeptierten-Ausschluß, also mit allen Beschränkungsmöglichkeiten, für die Aufgabenstellungen assoc, midos und impint;
>
> ohne_tm: ohne Teilmengenbedingung, mit Taxonomie, Optimumschätzfunktionen und Akzeptierten-Ausschluß, für die Aufgabenstellungen assoc, midos und impint;
>
> ohne_tax: mit Teilmengenbedingung, ohne Taxonomie, mit Optimumschätzfunktionen und mit Akzeptierten-Ausschluß, also unter Verzicht auf Taxonomie, für die Aufgabenstellungen assoc, midos und impint;
>
> schwach_oe/ohne_oe: mit Teilmengenbedingung und mit Taxonomie, aber ohne oder mit schwachen Optimumschätzfunktionen, mit Akzeptierten-Ausschluß, also unter Verzicht auf Optimumschätzfunktionen; nur für die Aufgabenstellungen midos und impint, da es für die assoc-Aufgabe keine Optimumschätzfunktionen gibt;
>
> ohne_aka: mit Teilmengenbedingung, mit Taxonomie und mit Optimumschätzfunktionen, ohne Akzeptierten–Ausschluß; nur in der Aufgabenstellung impint, da für die Aufgabenstellungen assoc und midos der Ausschluß akzeptierter Regeln nicht vorgesehen ist.

Die Tafeln 6.1 und 6.2 fassen die Ergebnisse der Versuchsreihen für die Phonetik- beziehungsweise die KRK-Datenbank zusammen, die mit Prolog als Anfragesprache und dem Prolog-Interpreter zur Auswertung der Anfragen erzielt wurden. Sie zeigen für jeden Suchlauf:

Kandidaten/otm: die Anzahl der Datenbank-Anfragen, die ohne Überprüfen der Teilmengenbedingung nötig wären. Diese Anfragen wurden nicht tatsächlich ausgeführt, sondern lediglich gezählt. Ausgeführt wurden nur die Datenbank-Anfragen, die nach der Überprüfung der Teilmengenbedingung noch erforderlich waren;

Kandidaten/tm: die Anzahl der Datenbank-Anfragen, die nach Überprüfung der Teilmengenbedingung noch erforderlich waren und ausgeführt wurden;

Zeit/otm: die Zeit in Sekunden, die für das Erzeugen der Hypothesen ohne Überprüfung der Teilmengenbedingung benötigt wurde;

Zeit/tm: die Anzahl der Sekunden, die für die anschließende Überprüfung der Teilmengenbedingung für alle erzeugten Hypothesen aufgewendet wurde;

Zeit/∅ db: die durchschnittliche Dauer einer Datenbank–Anfrage. Da der Rechner, der die Datenbank-Anfragen ausführte, gleichzeitig noch andere Aufgaben erledigte, können diese Werte nur einen ungefähren Eindruck vermitteln. Tendenziell dauern Anfragen, die für viele Individuen gelingen, länger als solche, die nur kleine Ergebnismengen liefern;

Zeit/gesamt: die gesamte Laufzeit (nicht CPU-Zeit) eines Suchlaufs in Sekunden.

Mit einer Ausnahme liefen alle Suchläufe, bis sie den gesamten Suchraum durchwandert hatten und keine weiteren Hypothesen mehr generiert werden konnten. Der mit * markierte Suchlauf wurde nach der angegebenen Zeit ($\approx$ 81000 Sekunden = 22.5 Stunden) abgebrochen, nachdem er die fünfte Suchebene durchsucht hatte. Bis dahin hatte er 77849 Hypothesen erzeugt und ausgewertet.

Zum Vergleich wurden die Suchläufe, die alle Möglichkeiten zur Beschränkung des Suchraums einsetzen, mit SQL als Anfragesprache und IBM DB2 als Datenbankmanagement durchgeführt. Die Werte, die diese Suchläufe ergeben haben, sind in den Tabellen 6.2 und 6.4 wiedergegeben. In den einander entsprechenden Prolog- und den KRK-Experimenten werden identische Hypothesenräume durchsucht. Auch die Parameter-Einstellungen in korrepondierenden Suchläufen stimmen überein. Deshalb erzeugten einander entsprechende Suchläufe dieselben Kandidatenzahlen vor und nach der Teilmengenbeschneidung (Spalten otm und tm). Die Tabellen 6.1 und 6.3 enthalten deshalb in allen Spalten außer den beiden letzten dieselben Werte; dasselbe gilt für die Tabellen 6.2 und 6.4. Die Deklarationen für die SQL-Hypothesensprachen sind in Anhang A aufgeführt.

Die Auswertung von Anfragen auf der KRK-Datenbank dauerte mit dem Prolog-Interpreter und mit der SQL-Datenbank im Durchschnitt ungefähr gleich lang (Spalten ∅ db in Tabellen 6.2 und 6.4), so daß auch ähnliche Gesamtlaufzeiten resultierten (Spalten gesamt in Tabellen 6.2 und 6.4). Dagegen zeigen die Tabellen 6.1 und 6.3 zur Phonetik-Datenbank jedoch sehr unterschiedliche Werte für die durchschnittliche Dauer von Datenbankanfragen (Spalten ∅ db in Tabellen 6.1 und 6.3) und unterschiedliche Gesamtlaufzeiten (Spalten gesamt in Tabellen 6.1 und 6.3). Darauf wird in Abschnitt 6.3.2 noch näher eingegangen. Die Auswertung der SQL-Anfragen auf der Phonetik-Datenbank dauerte um ein Vielfaches länger als die Auswertung der entsprechenden Prolog-Anfragen, obwohl der Prolog-Interpreter und die SLQ-Datenbank auf demselben Rechner und mit ähnlicher Rechner-Auslastung liefen.

Tabelle 6.1 Ergebnisse für die Phonetik-Datenbank.

		Kandidaten		Zeit in sec.			
	Par.	otm	tm	otm	tm	∅ db	gesamt
mit Taxonomie, Optimumschätzfunktionen und Akzeptierten-Ausschluß							
assoc	0.100000	2412	1174	0	1	0.965928	1173
	0.200000	724	322	0	0	0.987578	333
	0.300000	307	192	0	0	0.973958	199
	0.400000	159	105	0	0	1.000000	117
midos	0.000100	1487	883	0	0	0.937712	857
	0.100000	727	346	0	0	0.979769	358
	0.200000	318	200	0	0	0.970000	206
	0.300000	208	144	0	0	0.965278	155
impint	0.000100	3035	1529	1	2	0.905821	1452
	0.005000	1943	950	0	0	0.932632	923
	0.010000	1283	684	0	0	0.972222	696
	0.050000	1417	744	0	2	0.998656	769
ohne Taxonomie, mit Optimumschätzfunktionen und Akzeptierten-Ausschluß							
assoc	0.100000	4543	1426	0	2	0.921459	1375
	0.200000	1577	424	0	1	0.900943	408
	0.300000	782	243	0	0	0.876543	232
	0.400000	428	171	0	0	0.853801	160
midos	0.000100	5018	1805	1	1	0.762327	1428
	0.100000	1871	513	0	0	0.892788	480
	0.200000	1061	303	0	0	0.871287	280
	0.300000	631	209	0	0	0.861244	197
impint	0.000100	11512	3281	1	4	0.810728	2795
	0.005000	8068	1962	0	2	0.849643	1734
	0.010000	6634	1672	1	2	0.834330	1467
	0.050000	5376	1573	2	2	0.854418	1414
mit Taxonomie, ohne Optimumschätzfunktionen mit Akzeptierten-Ausschluß							
midos	$*$ 0.000100	-	77849	-	-	-	81000
	0.100000	1583	839	0	0	0.970203	856
	0.200000	454	275	0	0	0.996364	291
	0.300000	216	148	1	0	0.993243	161
impint	0.050000	5721	2932	0	5	0.915757	2807
	0.000100	13184	7091	1	27	0.888027	6646
	0.005000	8132	4008	0	10	0.889222	3748
	0.010000	5796	2963	0	6	0.908876	2834
mit Taxonomie und Optimumschätzfunktionen, ohne Akzeptierten-Ausschluß							
impint	0.000100	35048	19470	3	195	0.894504	18661
	0.005000	42900	24330	3	318	0.897000	23630
	0.010000	44478	25430	3	348	0.891624	24673
	0.050000	48737	28638	4	447	0.987394	30660

Tabelle 6.2 Ergebnisse für die KRK-Datenbank.

		Kandidaten		Zeit in sec.			
	Par.	otm	tm	otm	tm	∅ db	gesamt
mit Taxonomie, Optimumschätzfunktionen und Akzeptierten-Ausschluß							
assoc	0.100000	5340	3157	0	7	0.668990	2154
	0.200000	1575	862	0	0	0.799304	700
	0.300000	784	378	0	0	0.875661	333
	0.400000	300	173	0	0	0.947977	169
midos	0.000100	868	551	0	0	0.840290	471
	0.100000	827	479	0	0	0.853862	416
	0.200000	562	278	0	0	0.946043	266
	0.300000	297	161	0	0	0.981366	160
impint	0.000100	8710	5224	1	23	0.606049	3312
	0.005000	8178	4998	0	26	0.639056	3333
	0.010000	7304	4562	0	18	0.716791	3398
	0.050000	6559	4132	1	13	0.711278	3049
ohne Taxonomie, mit Optimumschätzfunktionen und Akzeptierten-Ausschluß							
assoc	0.100000	7485	3421	0	9	0.639871	2257
	0.200000	2440	913	1	2	0.785323	732
	0.300000	1355	465	0	0	0.804301	381
	0.400000	525	203	0	0	0.911330	188
midos	0.000100	1725	629	0	0	0.682035	433
	0.100000	1475	573	0	0	0.713787	414
	0.200000	1090	380	0	0	0.881579	340
	0.300000	865	314	0	0	0.863057	276
impint	0.000100	11890	5401	0	24	0.603592	3382
	0.005000	11000	5114	1	21	0.630622	3363
	0.010000	9645	4636	1	17	0.682485	3251
	0.050000	8700	4231	1	15	0.731033	3182
mit Taxonomie, ohne Optimumschätzfunktionen mit Akzeptierten-Ausschluß							
midos	0.000100	41587	37254	5	1186	0.382321	17728
	0.100000	5959	3565	0	9	0.719495	2639
	0.200000	2022	906	0	1	0.847682	782
	0.300000	1009	571	0	1	0.868652	504
impint	0.005000	9406	6627	0	35	0.605402	4194
	0.010000	8170	5759	0	27	0.677722	4058
	0.050000	7269	5124	2	22	0.687158	3659
	0.000100	10774	7525	2	45	0.556678	4427
mit Taxonomie und Optimumschätzfunktionen, ohne Akzeptierten-Ausschluß							
impint	0.000100	29719	23744	4	454	0.463106	12611
	0.005000	30473	24391	1	486	0.452749	12768
	0.010000	30607	24466	5	496	0.453854	12834
	0.050000	31041	24865	3	510	0.448944	13001

Tabelle 6.3 Ergebnisse mit der SQL-Version für die Phonetik-Datenbank.

	Par.	Kandidaten otm	Kandidaten tm	Zeit in sec. otm	Zeit in sec. tm	Zeit in sec. ∅ db	Zeit in sec. gesamt
mit Taxonomie, Optimumschätzfunktionen und Akzeptierten-Ausschluß							
assoc	0.100000	2412	1174	0	0	2.964225	3518
	0.200000	724	322	0	0	2.692547	878
	0.300000	307	192	0	0	2.302083	448
	0.400000	159	105	0	0	2.104762	226
midos	0.000100	1487	883	0	0	2.383919	2131
	0.100000	727	346	0	0	2.621387	919
	0.200000	318	200	0	0	2.420000	490
	0.300000	208	144	0	0	2.291667	334
impint	0.000100	3035	1529	0	0	2.075212	3223
	0.005000	1943	950	0	1	2.013684	1935
	0.010000	1283	684	0	1	1.988304	1379
	0.050000	1417	744	0	1	1.981183	1504

Tabelle 6.4 Ergebnisse mit der SQL-Version für die KRK-Datenbank.

	Par.	Kandidaten otm	Kandidaten tm	Zeit in sec. otm	Zeit in sec. tm	Zeit in sec. ∅ db	Zeit in sec. gesamt
mit Taxonomie, Optimumschätzfunktionen und Akzeptierten-Ausschluß							
assoc	0.100000	5340	3157	0	7	0.655686	2142
	0.200000	1575	862	0	0	0.800464	695
	0.300000	784	378	1	0	0.878307	336
	0.400000	300	173	0	0	0.971098	170
midos	0.000100	868	551	0	0	0.818512	454
	0.100000	827	479	0	0	0.839248	407
	0.200000	562	278	0	0	0.920863	258
	0.300000	297	161	0	0	0.968944	161
impint	0.000100	8710	5224	0	24	0.576953	3161
	0.005000	8178	4998	1	23	0.588435	3059
	0.010000	7304	4562	2	17	0.604121	2856
	0.050000	6559	4132	0	16	0.607212	2613

6.3 Ergebnisse

Um den Nutzen der einzelnen Methoden zur Suchraumbeschränkung zu illustrieren, stellen die folgenden Abbildungen den Umfang des Suchraums, der sich bei Verzicht auf eine Beschränkungsmethode ergibt, dem Umfang des Suchraums gegenüber, der bei Einsatz aller Beschränkungsmöglichkeiten durchsucht wird. Der Umfang des Suchraums ist dabei die Anzahl der Hypothesen, die im betreffenden Suchlauf per Datenbankanfrage ausgewertet werden; diese werden als „Kandidaten“ bezeichnet. Die Kurve für den Referenzsuchlauf, in dem alle Beschränkungsmöglichkeiten verwendet wurden, ist in den Graphiken mit gleicher Datenbank und Aufgabenstellung bis auf den unterschiedlichen Maßstab identisch. Die Graphiken stellen den Umfang des Suchraums in je einer der Aufgabenstellungen as-

soc, midos, impint in je einer der Datenbanken Phonetik oder KRK für die verschiedenen Parametereinstellungen dar. Für die Aufgabenstellungen assoc und midos variiert der Parameter Mindesthäufigkeit (σ_a beziehungsweise σ_m). In der Aufgabenstellung impint ändert sich der Parameter α, $1 - \alpha$ ist die für Akzeptanz mindestens verlangte Folgerungsstärke einer Hypothese. Die Parametereinstellungen sind auf der x-Achse abgetragen, die Anzahl der nötigen Datenbankanfragen an der y-Achse, die mit „Kandidaten“ beschriftet ist.

In den Graphiken zu den Aufgabenstellungen assoc und midos steigt die verlangte Mindesthäufigkeit von links nach rechts. Also ist das Akzeptanzkriterium von links nach rechts immer schwerer zu erfüllen und der Umfang des Suchraums verringert sich daher von links nach rechts.

Die Aufgabenstellung impint ist so festgelegt, daß akzeptierte Hypothesen in der weiteren Suche nicht mehr spezialisiert, sondern aus dem Suchraum ausgeschlossen werden. Damit wirkt das Akzeptanzkriterium zugleich als ein Ausschlußkriterium. Das heißt, je leichter das Akzeptanzkriterium zu erfüllen ist, desto mehr Hypothesen lassen sich durch den Akzeptierten–Ausschluß aus dem Suchraum ausschließen. Die anderen Methoden zur Beschränkung des Suchraums, Taxonomien, Mindesthäufigkeitsforderungen und Optimumschätzfunktionen, sind desto wirksamer, je strenger das Akzeptanzkriterium ist. Der Umfang des Suchraums muß sich in dieser Aufgabenstellung also nicht monoton zu α verhalten.

6.3.1 Nutzen der Teilmengenbedingung

Abbildung 6.2 stellt die Anzahl an Hypothesen, die in den verschiedenen Versuchsläufen ohne Überprüfung der Teilmengenbedingung auszuwerten wären (ohne_tm), denen gegenüber, deren Auswertung nach der Überprüfung der Teilmengenbedingung noch erforderlich ist (all). Sie illustriert also, wieviele Datenbank-Anfragen die Überprüfung der Teilmengenbedingung einspart. Die Illustrationen gelten für die Versuchsreihe all, in der alle vorhandenen Möglichkeiten zur Suchraumbeschränkung eingesetzt wurden. Den Tafeln 6.1 und 6.2 kann man entnehmen, daß der Zeitbedarf für die Elimination von Hypothesen durch Überprüfung der Teilmengenbedingung nur einen geringen Anteil an der Gesamtlaufzeit eines Suchlaufs ausmacht. Insgesamt zeigt sich, daß die Überprüfung der Teilmengenbedingung bei geringen Kosten gute Einsparungsmöglichkeiten bietet. Die Wirkung ist recht zuverlässig, das heißt, die Einsparung von Datenbankanfragen tritt recht gleichmäßig in allen Suchläufen auf und beträgt meist ungefähr ein Viertel bis die Hälfte der Datenbankauswertungen.

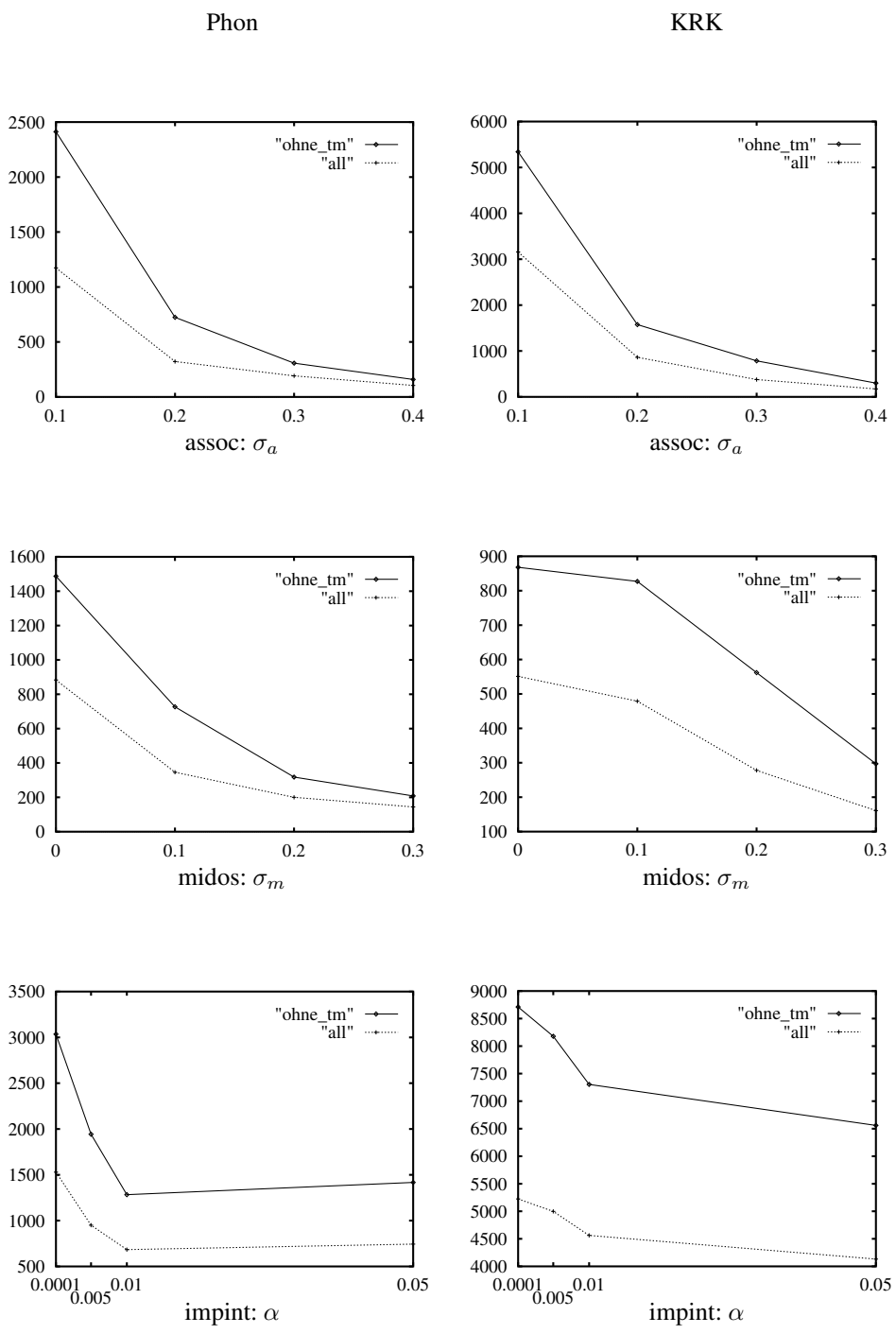
6.3.2 Nutzen von Subsumtionsbeziehungen zwischen Merkmalen

KRK

Die Beitrag der Taxonomien und Subsumtionsbeziehungen zur Reduktion des Suchaufwands erweist sich in den verschiedenen Suchläufen als sehr uneinheitlich. Im Hypothesenraum zur KRK-Datenbank gibt es nur wenige Subsumtionsbeziehungen zwischen Merkmalen; insgesamt sind 5*6 = 30 Subsumtionsbeziehungen gegeben. Diese bewirken entsprechend eher geringe Reduktionen des Suchraums. Für die KRK-Datenbank ergaben sich die größten Einsparungen durch Subsumtionsinformationen in den midos–Läufen (siehe Abbildung 6.3). Der Grund dafür ist möglicherweise, daß die midos–Versuche durch k-besten–Suche und Optimumschätzfunktionen die Subsumtionsbeziehungen besonders gut zur Suchraumbeschränkung verwerten können. Die Abbildung 6.3 zeigt auch, daß die Kandidatenzahlen in den assoc- und impint-Versuchen mit der KRK-Datenbank durch die Subsumtionsinformationen nur sehr wenig reduziert werden.

Auffällig ist hier, daß die Berücksichtigung der Subsumtionsbeziehungen zwischen Merkmalen die Laufzeiten der Suchläufe kaum verbessert und zum Teil sogar verschlechtert, auch wenn weniger Datenbank-Anfragen ausgeführt werden (vergleiche Tafel 6.2).

Abbildung 6.2 Anzahl der Hypothesenauswertungen mit Überprüfung der Teilmengenbedingung (all) versus ohne Überprüfung der Teilmengenbedingung (ohne_tm).



Bei den midos–Läufen mit der KRK-Datenbank liegt das unter anderem an der Dynamik des Akzeptanzkriteriums, das sich durch die k-besten–Suche im Verlauf der Suche ständig verschärft. Dadurch spielt hier die Reihenfolge eine Rolle, in der Hypothesen ausgewertet werden. Auch wenn der gleiche Suchraum durchsucht wird, verändert die Taxonomie-Information die Auswertungsreihenfolge, weil allgemeinere Hypothesen vor den spezielleren Hypothesen evaluiert werden müssen und die Subsumtionsbeziehungen zwischen Merkmalen andere Subsumtionsbeziehungen zwischen Hypothesen induzieren. In den KRK-Versuchen ist die Auswertung der adjacent-Merkmale besonders teuer, weil sie Disjunktionen beinhalten. Da die adjacent-Merkmale andere Merkmale subsumieren, führt die Berücksichtigung der Taxonomie-Information dazu, daß sie früh im Verlauf der Suche ausgewertet werden. Dann ist das Akzeptanzkriterium noch schwach und kann daher weniger Hypothesen aussondern. Im Suchlauf ohne Taxonomie können dagegen unter Umständen die spezielleren Hypothesen schon früher evaluiert werden und dazu beitragen, das Akzeptanzkriterium zu verschärfen, so daß die teuer auszuwertenden Hypothesen und damit ihre gleichermaßen teuren Spezialisierungen gekappt werden können. In den midos–Suchläufen mit $\sigma_m = 0.0001$ wirkt sich das beispielsweise so aus, daß im Suchlauf mit Taxonomie die teure Hypothese $h =$

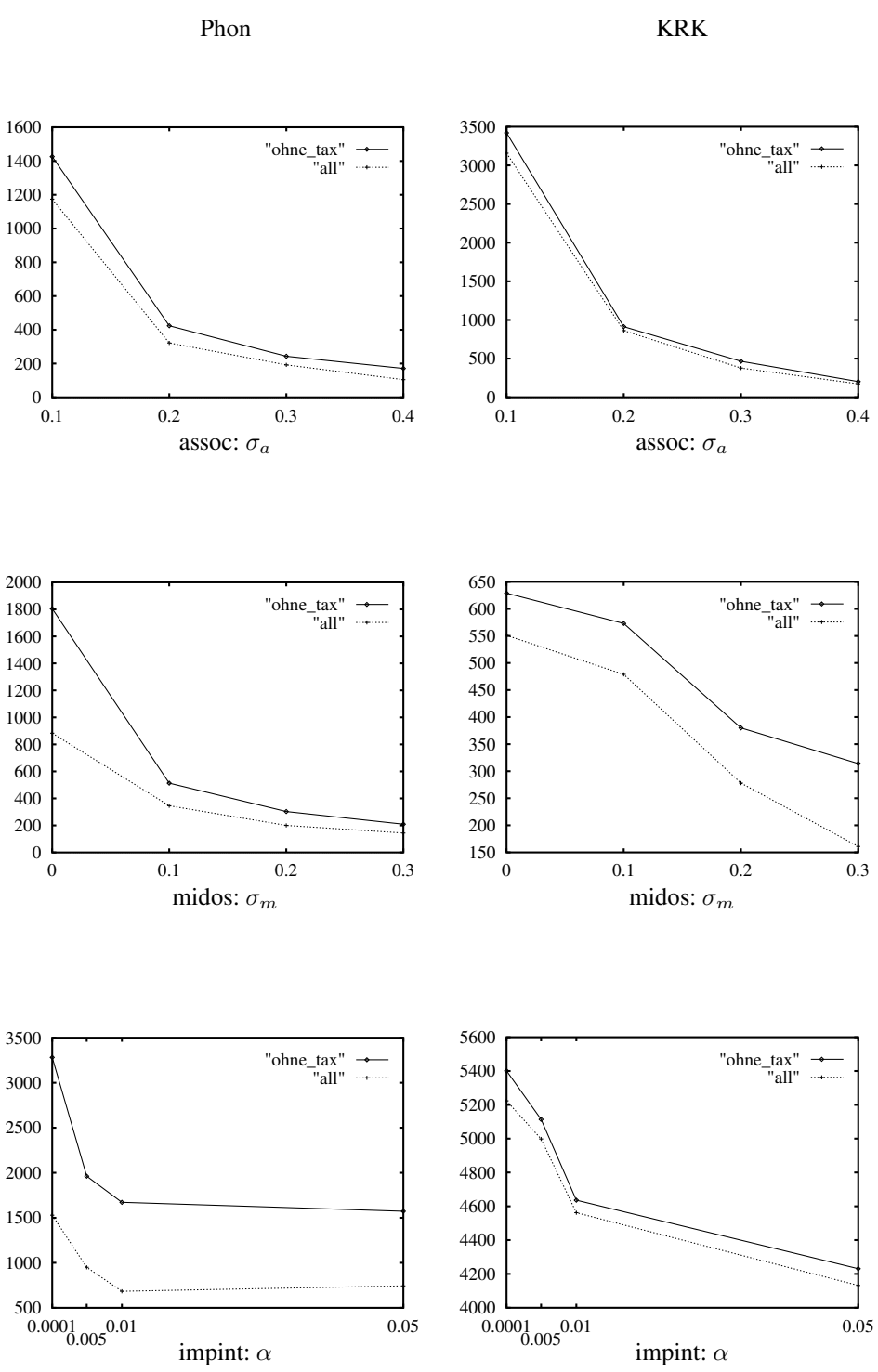
$$krk\_pos(Id, Class, \textit{WKC}, \textit{WKR}, \textit{WRC}, \textit{WRR}, \textit{BKC}, \textit{BKR}),$$
$$N1 \text{ is } \textit{WKC} - \textit{WRC}, (N1 = 1; N1 = -1; N1 = 0),$$
$$N2 \text{ is } \textit{WKR} - \textit{WRR}, (N2 = 1; N2 = -1; N2 = 0)$$

als 14. ausgewertet wird und nicht gekappt werden kann. Im weiteren Verlauf der Suche werden 10 ebenfalls teure Spezialisierungen dieser Hypothese erzeugt und ausgewertet. Dagegen wird die Hypothese h im Suchlauf ohne_tax als 135. ausgewertet und kann dann gekappt werden. Im Vergleich zum all-Suchlauf erspart dies 10 teure Datenbankanfragen. Obwohl also insgesamt im all-Suchlauf deutlich weniger Hypothesen auszuwerten sind, ist die Auswertung dieser Hypothesen durchschnittlich teurer als die Auswertung der Hypothesen im Fall ohne_tax (vergleiche Tafel 6.2), so daß der all-Suchlauf insgesamt länger dauert als der Suchlauf ohne_tax.

Phonetik

Für die Hypothesensprache für die Phonetik-Datenbank liegt mehr Taxonomie-Information vor. Die Taxonomien sind stärker verzweigt mit mehr Ebenen und ermöglichen dadurch auch größere Einsparungen als in der KRK-Datenbank. Auffällig ist der starke Effekt der Taxonomie in der impint-Versuchsreihe mit der Phonetik-Datenbank, der wohl darauf zurückzuführen ist, daß hier der Ausschluß akzeptierter Hypothesen starke Suchraum-Einschränkungen ermöglicht hat. Tafel 6.10 zeigt die Anzahl akzeptierter Hypothesen in den impint-Versuchen mit der Phonetik-Datenbank und zum Vergleich auch mit der KRK-Datenbank. Man entnimmt ihr einen deutlichen Unterschied zwischen der Anzahl akzeptierter Hypothesen mit Taxonomie und der Anzahl akzeptierter Hypothesen ohne Taxonomie für die Phonetik-Datenbank. Die Ergebnismengen einander entsprechender Suchläufe mit und ohne Taxonomie unterscheiden sich voneinander, wenn akzeptierte Hypothesen gekappt werden, weil aus einer Taxonomie auch bei ansonsten identischer Hypothesensprache zusätzliche Subsumtionsbeziehungen zwischen Hypothesen folgen, die den Ausschluß der subsumierten Hypothesen aus dem Suchraum mit sich bringen können, die ohne Kenntnis der Taxonomie im Suchraum verbleiben würden.

Abbildung 6.3 Anzahl der Hypothesenauswertungen mit Taxonomie (all) versus ohne Taxonomie (ohne_tax).



Tafel 6.10 Anzahl akzeptierter Hypothesen in den impint-Versuchen mit versus ohne Taxonomie.

	Phon		KRK	
α	all	ohne_tax	all	ohne_tax
0.000100	15	42	6	11
0.005000	17	43	8	12
0.010000	16	41	8	12
0.050000	19	49	9	15

6.3.3 Nutzen der Optimumschätzfunktionen

Für die Verteilungsungewöhnlichkeit, (die Interessantheitsfunktion der Aufgabenstellung midos) sind die in Abschnitt 3.3.2 eingeführten Optimumschätzfunktionen da_{oe1} und da_{oe2} verfügbar. Eine abgeschwächte Version von da_{oe1} ist da_{oe3}. Für die Interessantheitsfunktion der Aufgabenstellung impint, die Folgerungsstärke, gibt es die Optimumschätzfunktion II_{oeN}, die, wenn wie hier eine Mindesthäufigkeit σ_i verlangt ist, verbessert werden kann zu $II_{oeN,\sigma}$. In den Referenzsuchläufen all wurden in der Aufgabenstellung midos die Optimumschätzfunktionen da_{oe1} und da_{oe2} eingesetzt. In den Suchhläufen schwach_oe wird da_{oe1} durch die schwächere da_{oe3} ersetzt und da_{oe2} beibehalten. In den Suchläufen ohne_oe wurde keine Optimumschätzfunktion ausgewertet. Suchraumbeschränkungen anhand der Mindesthäufigkeit fanden jedoch in allen Suchläufen statt.

In den Suchläufen zur Aufgabenstellung impint wird sonst immer die in Abschnitt 3.4.4 definierte Optimumschätzfunktion $II_{oeN,\sigma}$ verwendet. In den Suchläufen schwach_oe ist $II_{oeN,\sigma}$ durch II_{oeN} ersetzt; in den Suchläufen ohne_oe wird keine Optimumschätzfunktion berechnet. Suchraumbeschränkung mithilfe der Mindesthäufigkeit σ_i findet immer statt. In den hier durchgeführten Experimenten ist die Mindesthäufigkeit von $\sigma_i = 0.005$ (das entspricht 100 Fällen) so groß, daß die Folgerungsstärke für alle akzeptablen Hypothesen nach dem Normalverteilungszweig der Folgerungsstärke berechnet wird. Die Optimumschätzfunktion $II_{oeN,\sigma}$ und II_{oeN} bilden damit sichere Beschränkungskriterien.

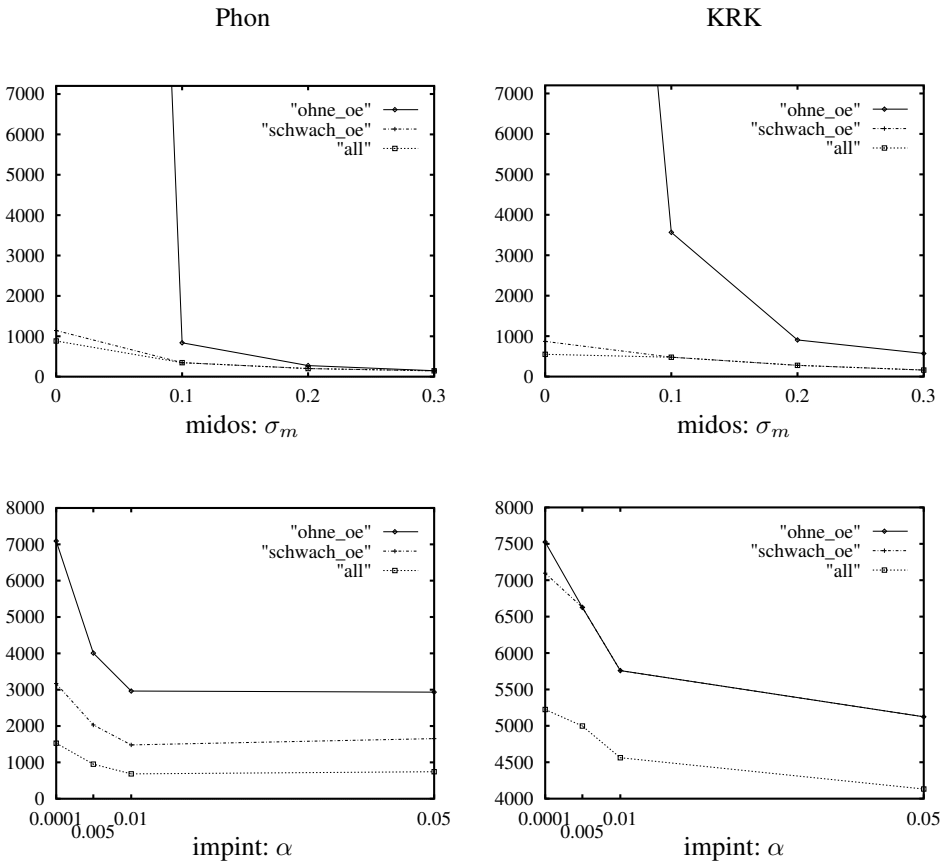
Die Graphiken in Abbildung 6.4 veranschaulichen den Einfluß der Optimumschätzfunktionen auf die Anzahl von auszuwertenden Hypothesen. Die Kurven mit der Markierung ohne_oe gehören zu den Versuchen ohne Optimumschätzfunktionen; die Kurven mit der Markierung all gehören zu den Versuchen mit Optimumschätzfunktionen, und die mit schwach_oe markierten Kurven gehören zu den Versuchen mit abgeschwächten Optimumschätzfunktionen. Die Abbildungen zeigen, daß die starken Versionen der Optimumschätzfunktionen gut und zuverlässig wirken, das heißt, sie erbringen für beide Datenbanken in beiden Aufgabenstellungen impint und midos mit beinahe allen Parametereinstellungen gute Einsparungen. In der Aufgabenstellung impint wirkt die abgeschwächte Optimumschätzfunktion wesentlich schwächer als die starke Version. In der midos–Aufgabe bringt die verbesserte Optimumschätzfunktion jedoch nur in den Läufen mit geringer Mindesthäufigkeit einen Vorteil gegenüber der schwächeren Optimumschätzfunktion. Das Verhalten der Optimumschätzfunktionen wird kurz genauer betrachtet.

Midos

Tafel 6.11 vergleicht die Wirksamkeit der Optimumschätzfunktionen da_{oe1}, da_{oe2} und da_{oe3} und der Mindesthäufigkeit. Sie zeigt

- insgesamt (ms/oe1/oe2/oe3): die Anzahl der Fälle, in denen das Mindesthäufigkeitskriterium ms beziehungsweise eine der Optimumschätzfunktionen da_{oe3}, da_{oe2} oder da_{oe1} während eines Suchlaufs angesprochen, das heißt, die Kappbarkeit einer Hypothese signalisiert hat;

Abbildung 6.4 Anzahl der Hypothesenauswertungen mit Optimumschätzfunktion (all), mit schwacher Optimumschätzfunktion (schwach_oe) und ohne Optimumschätzfunktion (ohne_oe).



- einzig (oe1/oe2): die Anzahl der Fälle, in denen die Optimumschätzfunktion da_{oe1} beziehungsweise die Optimumschätzfunktion da_{oe2} als einzige angesprochen hat;

- Σ: die Gesamtzahl der Hypothesen, für die eines oder mehrere der Beschränkungskriterien da_{oe1}, da_{oe2} und Mindesthäufigkeit, angesprochen haben.

Für die Aufgabenstellung midos sind die Optimumschätzfunktionen besonders wirkungsvoll in den Suchläufen mit sehr niedriger Mindesthäufigkeit, wo demzufolge die Mindesthäufigkeit nur sehr wenige Möglichkeiten zur Suchraumbeschränkung bietet. Der Effekt der Optimumschätzfunktionen wird geringer, wenn höhere Mindesthäufigkeiten gefordert sind, und verschwindet fast ganz im Suchlauf mit maximalem $\sigma_m = 0.3$ angewandt auf die Phonetik-Datenbank.

Die Tafel 6.11 zeigt, daß die Optimumschätzfunktion da_{oe1} besonders gut wirkt, wenn eine niedrige Mindesthäufigkeit geringe Möglichkeiten zur Suchraumbeschränkung bietet. Die Optimumschätzfunktion da_{oe2} erkennt besonders dann viele Hypothesen als einziges Kriterium als kappbar, wenn das Mindesthäufigkeitskriterium stark ist, und ergänzt daher das Mindesthäufigkeitskriterium und die Optimumschätzfunktion da_{oe1} gut. Die Tafel zeigt auch, daß die Optimumschätzfunktion da_{oe1} für sich betrachtet hier sehr viel stärker ist als da_{oe3}. Der Abbildung 6.4 entnimmt man aber, daß dieser Vorteil auf die Zahl der ausgewerteten Hypothesen kaum durchschlägt. Dies zeigt, daß die übrigen beiden Beschränkungskriterien Mindesthäufigkeit und da_{oe2} die Schwächen von da_{oe3} gegenüber da_{oe1} gut aus-

Tafel 6.11 Vergleich der Optimumschätzfunktionen für die Verteilungsungewöhnlichkeit in den Suchläufen zur Aufgabenstellung midos.

	insgesamt				einzig			Σ
σ_m	ms	oe1	oe2	oe3	ms	oe1	oe2	
Phon								
0.000100	3	666	559	200	0	107	0	666
0.100000	142	150	252	11	9	0	78	261
0.200000	132	30	139	4	24	0	31	163
0.300000	123	0	90	0	39	0	6	129
KRK								
0.000100	0	339	170	69	0	169	0	339
0.100000	38	267	274	37	0	6	13	280
0.200000	68	55	151	0	9	0	76	160
0.300000	54	3	103	0	8	0	57	111

Tafel 6.12 Nützlichkeit der Optimumschätzfunktion zur Folgerungsstärke in den Suchläufen zur Aufgabenstellung impint.

	all		schwach_oe	
α	ms	oe	ms	oe
Phonetik				
0.000100	100	563	676	429
0.005000	78	314	421	195
0.010000	39	202	264	101
0.050000	43	195	317	95
KRK				
0.000100	8	293	8	84
0.005000	8	232	8	0
0.010000	8	185	8	0
0.050000	8	164	8	0

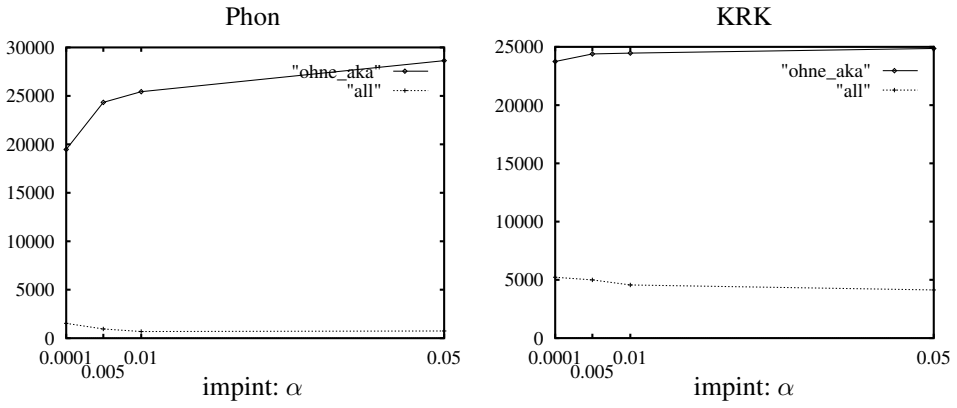
gleichen konnten. Die Schwäche von da_{oe3} gegenüber da_{oe1} wirkt sich nur für die Hypothesen aus, für die da_{oe1} als einziges Kriterium eine Beschränkungsmöglichkeit geboten hat.

Impint

Tafel 6.12 zeigt die Anzahl der Hypothesen, die anhand des Mindesthäufigkeitskriteriums gekappt wurden (ms), und die Anzahl der Hypothesen, die mit den Optimumschätzfunktionen gekappt wurden, ohne daß das Mindesthäufigkeitskriterium das Kappen erlaubt hat. Der Tafel entnimmt man, daß in den Suchläufen mit der Phonetik-Datenbank das Mindesthäufigkeitskriterium bei schwacher Optimumschätzfunktion sehr viel mehr Hypothesen gekappt hat als wenn die starke Optimumschätzfunktion eingesetzt wurde.

In den Experimenten mit der KRK-Datenbank variiert die Zahl der durch das Mindesthäufigkeitskriterium gekappten Hypothesen zwischen den Läufen mit starker und mit schwacher Optimumschätzfunktion gar nicht. Hier konnte also das Mindesthäufigkeitskriterium die Schwäche der Optimumschätzfunktion nicht ausgleichen. Daß die Anzahl der Hypothesen, für die das Mindesthäufigkeitskriterium angesprochen hat, über alle Parame-

Abbildung 6.5 Anzahl der Hypothesenauswertungen mit Ausschluß akzeptierter Hypothesen (all) versus ohne Ausschluß akzeptierter Hypothesen (ohne_aka).



tereinstellungen hinweg konstant ist, ist deshalb plausibel, weil in den impint-Versuchen (anders als in den assoc- und midos–Suchläufen) die Mindesthäufigkeit über alle Läufe konstant ist und nur der Schwellwert für die Folgerungsstärke sich ändert.

6.3.4 Effekt des Kappens akzeptierter Hypothesen

Abbildung 6.5 stellt die Anzahl ausgewerteter Hypothesen in der Aufgabenstellung impint mit versus ohne Kappen akzeptierter Hypothesen einander gegenüber. Die mit ohne_aka bezeichnete Kurve beschreibt die Suchläufe ohne Ausschluß akzeptierter Hypothesen (aka steht für „Akzeptierten-Ausschluß“); die mit all bezeichnete Kurve beschreibt die Suchläufe mit Ausschluß akzeptierter Hypothesen. Die Graphiken zeigen, daß der Ausschluß akzeptierter Hypothesen den Suchaufwand stark reduziert. In diesen beiden Experimenten bewirkt der Verzicht auf ausführlichere Ergebnisse, den der Ausschluß akzeptierter Hypothesen mit sich bringt, auch eine wesentliche Reduktion des Suchaufwands. Insbesondere die Suchläufe auf der Phonetik-Datenbank dauern ohne Ausschluß akzeptierter Hypothesen sehr lange.

In allen hier durchgeführten Versuchen mit der KRK-Datenbank überwiegt der Einfluß des Ausschlusses akzeptierter Hypothesen den Einfluß der übrigen Methoden zur Suchraumbeschränkung, das heißt, der Gesamt-Suchaufwand verringert sich monoton mit wachsendem α. In den Versuchen mit der Phonetik-Datenbank überwiegt im Bereich zwischen $\alpha = 0.001, \ldots, 0.01$ ebenfalls der Einfluß der Ausschlusses akzeptierter Hypothesen (das heißt, der Suchaufwand verringert sich zunächst monoton mit wachsendem α). Zu $\alpha = 0.05$ steigt der Suchaufwand jedoch wieder an. Das bedeutet, daß hier der Effekt der übrigen Beschränkungsmethoden den Effekt des Ausschlusses akzeptierter Hypothesen überwiegt. Die entsprechenden Kurven in den Abbildungen 6.5 haben deshalb bei $\alpha = 0.01$ einen Knick.

Tafel 6.13 listet die Anzahl akzeptierter Hypothesen aus den verschiedenen Suchläufen auf. Ohne Ausschluß akzeptierter Hypothesen tritt hier auch das bekannte Problem auf, daß der Entdeckungs-Algorithmus unüberschaubar große Resultatmengen liefert.

6.4 Zusammenfassung

Die Experimente haben erbracht, daß bei einer Suche vom Allgemeinen zum Speziellen der Suchraum deutlich besser beschränkt werden kann, wenn alle Subsumtionsbeziehungen zwischen neu generierten und bereits evaluierten Hypothesen ausgewertet werden, als

Tafel 6.13 Anzahl akzeptierter Hypothesen in den impint-Suchläufen mit (all) versus ohne (ohne_aka) Ausschluß akzeptierter Hypothesen.

	Phon		KRK	
$alpha$	all	ohne_aka	all	ohne_aka
0.000100	15	4010	6	10593
0.005000	17	7606	8	12270
0.010000	16	8514	8	12632
0.050000	19	11234	9	13590

wenn der Suchraum lediglich dadurch beschränkt wird, daß Hypothesen, die bei der Anwendung als kappbar erkannt wurden, nicht weiter spezialisiert werden. Die vollständige Überprüfung der Subsumtionsbeziehungen mit den bereits evaluierten Hypothesen hat in allen durchgeführten Suchläufen zuverlässig große Kosteneinsparungen erbracht. Sie wird mit dem hier entwickelten Suchverfahren durch Überprüfung der Teilmengenbedingung realisiert. Die Suchraumbeschränkung mittels der Teilmengenbedingung war in den Experimenten für die mehrrelationale Datenbank gleichermaßen erfolgreich und mit ähnlich geringem zusätzlichen Aufwand verbunden wie für die ein-relationale Datenbank. Das heißt, daß die Überprüfung der Teilmengenbedingung nur wenig Zeit erforderte im Verhältnis zu der Zeit, die die eingesparten Datenbankanfragen gekostet hätten. Die erzielte Einsparung wurde nicht experimentell ermittelt; sie läßt sich abschätzen als die mittlere Dauer einer Datenbank-Anfrage mal die Anzahl der eingesparten Datenbank-Anfragen.

Die Taxonomien waren dagegen insgesamt weniger effektvoll zur Suchraumbeschränkung, und ihr Beitrag schwankte stark zwischen den verschiedenen Anwendungen. Offensichtlich hängt die Nützlichkeit einer Taxonomie zur Suchraumbeschränkung sehr von anwendungsabhängigen Gegebenheiten ab wie Tiefe und Verzweigungsgrad der Taxonomie. Taxonomien können nützlich sein, andererseits kann ihre Verarbeitung auch Zeit kosten, im Unterschied zu Optimumschätzfunktionen oder zum Ausschluß akzeptierter Hypothesen, die kaum Aufwand verursachen. Interessant waren in diesen Experimenten die Auswirkungen, die eine veränderte Auswertungsreihenfolge der Hypothesen auf die Gesamtlaufzeit ausübte. Hier bietet sich möglicherweise ein Ansatzpunkt zur Weiterentwicklung des Suchalgorithmus.

Die Optimumschätzfunktionen haben sich als sehr nützliches Mittel erwiesen, um den Suchaufwand zu reduzieren. Da die Berechnung der Optimumschätzwerte lediglich einfache arithmetische Operationen erfordert, verursacht ihr Einsatz nur geringe Kosten, führt aber besonders in Anwendungen mit wenig restriktivem Akzeptanzkriterium zu einer wesentlichen Beschleunigung der Suche. Es hat sich gezeigt, daß (abhängig von den Gegebenheiten der jeweiligen spezifischen Aufgabenstellung) schon einfache Verbesserungen an den Optimumschätzfunktionen zu deutlichen Verbesserungen der Suche führen können. Weiterhin wurde deutlich, daß es sehr vorteilhaft ist, mehrere Optimumschätzfunktionen und daraus abgeleitete Beschränkungskriterien miteinander zu kombinieren, um eine gleichmäßige Kostenreduktion über verschiedene Anwendungen hinweg zu erreichen.

Ausschluß akzeptierter Hypothesen bedeutet hier, daß akzeptierte Hypothesen nicht weiter spezialisiert, sondern alle von ihnen subsumierten Hypothesen aus dem Suchraum ausgeschlossen werden. Dadurch wird die Suche schneller, aber die Lösungsmenge kleiner. In den hier durchgeführten Experimenten war der Ausschluß akzeptierter Hypothesen sehr wirksam, um den Suchraum zu beschränken. Dem Nachteil, daß dadurch möglicherweise weitere interessante Hypothesen übersehen wurden, stand somit tatsächlich eine deutliche Verbesserung der Suchaufwands gegenüber. In Anwendungen mit einem wenig trennenden Akzeptanzkriterium, als das sich die Folgerungsstärke hier erwiesen hat, trägt der Ausschluß akzeptierter Hypothesen auch dazu bei, die Anzahl akzeptierter Hypothesen in vernünftigen Grenzen zu halten.

Das hier entwickelte Suchverfahren bietet die Möglichkeit, die Hypothesensprache einzuengen durch die Deklaration von *req*- und *excl*-Bedingungen. Auch diese Bedingungen drücken Hintergrundwissen darüber aus, wie sinnvolle Hypothesen für die gegebene Suchaufgabe aussehen. Die Auswirkung dieser Bedingungen auf den Suchaufwand wurde nicht experimentell untersucht. Anders als die untersuchten Methoden zur Suchraumbeschränkung verringern sie den Umfang einer Hypothesensprache bereits unabhängig von den Inhalten einer zu untersuchenden Datenbank. Diese Verringerung läßt sich ohne Zugriff auf eine bestimmte Datenbank berechnen. Während eines Suchlaufs bieten die *req*-Bedingungen darüberhinaus, ähnlich wie Taxonomien, zusätzliche Möglichkeiten zur Suchraumbeschränkung.

Auch durch die Wahl strengerer Akzeptanzkriterien läßt sich der Suchaufwand vermindern. Der Effekt strengerer Akzeptanzkriterien auf den Suchaufwand wird innerhalb der Versuchsreihen evaluiert, in denen das Akzeptanzkriterium durch Parametereinstellung variiert.

Kapitel 7

Interessante Teilgruppen in einer Finanz-Datenbank

Anläßlich der PKDD99 organisierten P. Berka und M. Sochorová einen sogenannten Discovery Challenge, bei dem eine Datenbank mit Daten einer tschechischen Bank einem Data Mining-Prozeß unterzogen werden sollte [BZ99]. Der Discovery Challenge bot die Gelegenheit, den entwickelten Ansatz praxisnah anzuwenden und mit anderen Ansätzen zu vergleichen.

7.1 Die Data Mining-Anwendung

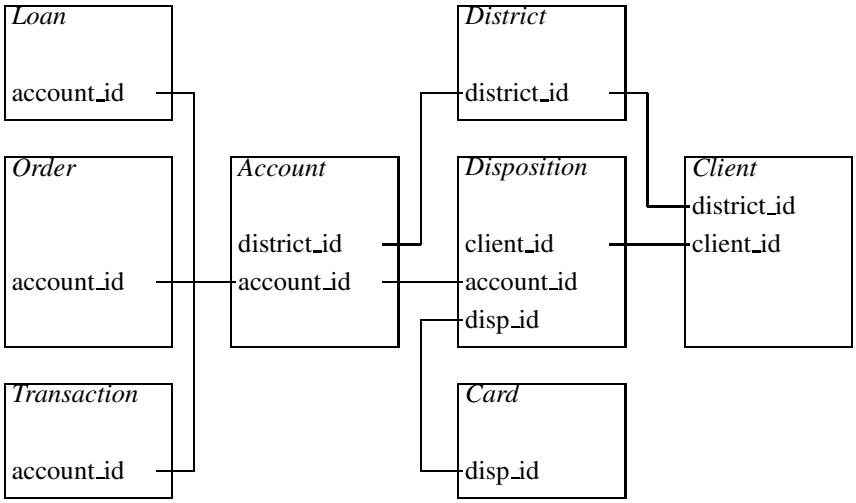
Die Organisatoren des Discovery Challenge haben keine spezifischen Data Mining-Ziele vorgegeben; die Definition geeigneter Ziele war vielmehr Bestandteil der Data Mining-Aufgabe. Das bereitgestellte Datenmaterial bietet vor allem zwei Data Mining-Ziele an, die auch in der im folgenden beschriebenen Anwendung bearbeitet wurden. Das erste Data Mining-Ziel ist, Indikatoren für gut verlaufende Kredite aufzudecken, im Vergleich zu Krediten, in deren Verlauf Probleme auftreten. Solche Indikatoren sind nützlich, um die Kreditvergabe-Strategien der Bank zu evaluieren und gegebenfalls zu überarbeiten. Ein zweites interessantes Data Mining-Ziel ist, typische Eigenschaften von Kreditkarteninhabern im Vergleich zu Konto-Inhabern ohne Kreditkarten zu entdecken, die der Bank helfen sollen, neue Kreditkarteninhaber zu gewinnen.

7.1.1 Die Finanz-Datenbank

Die Finanzdatenbank [Ber99a] besteht aus acht Relationen. Abbildung 7.1 gibt einen Überblick über die Relationen der Datenbank und die Beziehungen zwischen ihnen. Die Relation **account** umfaßt 4500 Objekte und beschreibt statische Eigenschaften von Konten, nämlich das Datum der Eröffnung, den Bezirk, in dem die Bankfiliale liegt, und die Häufigkeit, mit der Kontoauszüge erstellt werden. Mögliche Werte dieses letzten Attributs sind monatlich, wöchentlich oder unmittelbar nach jeder Kontobewegung. Die Relation **client** enthält 5369 Datensätze mit Informationen über Bankkunden, nämlich Geburtstag, Geschlecht und Wohnbezirk. Die Relation **disposition** mit 5369 Objekten stellt die Verbindung zwischen Kunden und Konten her und gibt an, ob ein Kunde als Inhaber oder als zusätzlicher Zugriffsberechtigter über das Konto verfügt. Die Relation **(permanent) order** beschreibt 6471 Daueraufträge. Sie gibt Kontonummer und Bank des Zahlungsempfängers, die Höhe der Überweisungen und möglicherweise den Zweck der Überweisung (Versicherung, Haushalt, Leasing oder Kreditraten) an.

Die Relation **transaction** beinhaltet 1 056 320 Beschreibungen von Transaktionen. Angegeben sind Kontonummer, Datum, der involvierte Geldbetrag, der Kontostand nach der

Abbildung 7.1 Datenbank-Schema: Relationen der Finanzdatenbank und Beziehungen zwischen ihnen



Transaktion sowie das jeweils zweite beteiligte Konto und die dazugehörige Bank. Weitere Angaben sind der Typ, der Modus und der Zweck der Transaktion. Der Typ gibt an, ob es sich um eine Gutschrift oder Abbuchung handelt. Mögliche Werte des Modus sind Abhebung mit Kreditkarte, Barabhebung, Bareinzahlung, Übertragung von oder zu einer anderen Bank. Als Zweck können Versicherungsbeitrag, Rechnung, Zinsgutschrift, Überziehungszins, Haushalt, Pensionszahlung oder Kreditrate vorkommen.

Die 682 Datensätze der Relation **loan** geben Eigenschaften von Krediten an. Hier sind das betroffene Konto, das Datum des Kreditbeginns, Zweck des Kredits, Kreditsumme und -dauer, Höhe der monatlichen Zahlungen und der Status des Kredits gespeichert. Der Kreditstatus kann die Werte a, b, c oder d annehmen. Der Wert a bedeutet, daß der Kredit ohne Probleme abgeschlossen wurde, b gibt an, daß der Kreditvertrag ohne vollständige Kreditrückzahlung abgelaufen ist. Bei c und d läuft der Kredit noch. Nur bei d sind bereits Zahlungsschwierigkeiten aufgetreten.

Die Relation **credit card** beschreibt 892 Kreditkarten. Zu jeder Kreditkarte gibt es einen Verweis auf die Relation **disposition**, die die Beziehung zum zugehörigen Konto und den Nutzungsrechten herstellt, außerdem das Ausgabedatum und den Typ der Karte. Mögliche Typen sind junior, classic und gold.

Demographisches Hintergrundwissen stellt die Relation **district** mit 77 Einträgen bereit. Sie beinhaltet für jeden Bezirk den Namen und die Region, sowie Einwohnerzahl, Häufigkeiten der Orte verschiedener Größe, Anzahl der Städte, Anteil städtischer Einwohner, Durchschnittseinkommen, Arbeitslosenrate von 1995 und 1996, den Anteil der Unternehmer pro 1000 Einwohner und schließlich die Anzahl krimineller Delikte von 1995 und 1996.

7.1.2 Wahl eines Interessantheitskriteriums

Das angestrebte Ergebnis der Anwendung ist nicht, Klassifikatoren mit hoher Vorhersagegenauigkeit herzuleiten, sondern verständliche und allgemeine Regelmäßigkeiten, die typisches Kundenverhalten beschreiben. Diese Aufgabenstellung kann auch als Entdecken von interessanten Teilgruppen aufgefaßt werden.

Die Aufgabenstellung erfordert eine vollständige Suche im Raum potentiell interessanter Muster. Wenn jedoch eine große Zahl von Mustern aufgezählt und evaluiert werden, erweisen sich meist sehr viele Assoziationen als signifikant, besonders in großen Daten-

beständen. Deshalb ist die Wahl eines guten Auswahlkriteriums in solchen Aufgabenstellungen besonders wichtig. Auch fehlt hier, im Unterschied zu Klassifikationsaufgaben ein objektives und allgemein anerkanntes Erfolgskriterium, wie es die Vorhersagegenauigkeit auf Testmengen für Klassifikationsaufgaben darstellt. Daher ist die Evaluation gefundener Muster notwendigerweise subjektiv und erfordert Kenntnisse des Anwendungsbereichs und eine Beurteilung durch Anwendungsexperten. Das Interessantheitskriterium trifft eine Vorauswahl unter den potenziell interessanten Mustern, die dann von Experten interpretiert und weiter selektiert werden müssen.

Von den Interessantheitskriterien, die in Kapitel 3 diskutiert wurden, kommen hier besonders die Folgerungsstärke und die Verteilungsungewöhnlichkeit in Frage. Die Kombination von Bestätigung und Genauigkeit (confidence und support) ist für diese Anwendung weniger geeignet, da sie die Vorkommenshäufigkeit und die Genauigkeit von Mustern getrennt bewertet, was in der vorliegenden Aufgabenstellung nicht den Intentionen entspricht.

Sowohl die Folgerungsstärke wie auch die Verteilungsungewöhnlichkeit können für eine Suche mit Schwellwert- oder mit k-besten-Akzeptanz verwendet und mit einer Mindesthäufigkeitsforderung kombiniert werden. Eine weitere Wahlmöglichkeit liegt darin, als interessant akzeptierte Regeln weiter zu spezialisieren oder aber aus dem Suchprozeß auszuschließen. Für die vorliegende Anwendung wurde aufgrund einiger Versuchsläufe die Folgerungsstärke als Interessantheitskriterium ausgewählt. Der Vorteil der Folgerungsstärke gegenüber der Verteilungsungewöhnlichkeit ist, daß sie als statistisches Maß eher eine absolute Aussage über die Interessantheit einer Teilgruppe trifft, während sich die Verteilungsungewöhnlichkeit die Interessantheit einer Teilgruppe eher relativ zur Qualität der anderen im Datensatz auffindbaren Regeln bewertet.

Die Folgerungsstärke hat die Eigenschaft, auch für sehr spezielle Regeln mit geringer Abdeckung hohe Werte zu ergeben, wenn diese Regeln sehr genau sind. In der vorliegenden Aufgabenstellung sind allgemeinere Teilgruppen interessanter, weil sie für mehr Fälle zutreffen und ihre Umsetzung deshalb größere Gewinnmöglichkeiten erhoffen läßt. Aus diesem Grund ist es hier sinnvoll, die Folgerungsstärke mit einem Schwellwert für eine Mindestabdeckung akzeptierter Teilgruppen zu kombinieren. Damit läßt sich auch die Suche stärker beschränken.

Obwohl die Suche mit k-besten-Akzeptanz weniger aufwendig ist und eine konzentriertere Ergebnismenge liefert, wird hier die Folgerungsstärke im Rahmen einer Suche mit Schwellwert-Akzeptanz eingesetzt, weil in den Probe-Suchläufen die umfassendere Suche mit anschließender manueller Auswahl der interessantesten Teilgruppen aus einer größeren Ergebnismenge interessantere Resultate lieferte und einen besseren Überblick über die Daten bot.

Die Voruntersuchungen zeigten auch, daß häufig, wenn eine Teilgruppenbeschreibung das Akzeptanzkriterium erfüllt, zugleich auch viele direkte oder indirekte Spezialisierungen dieser Teilgruppenbeschreibung das Akzeptanzkriterium erfüllen, falls nicht die weitere Spezialisierung akzeptierter Teilgruppenbeschreibungen explizit verhindert wird. Daher wurde das Akzeptanzkriterium so festgelegt, daß akzeptierte Teilgruppenbeschreibungen von der weiteren Spezialisierung ausgeschlossen werden. Die Suche liefert dann zwar weniger potentiell interessante Teilgruppenbeschreibungen, verursacht aber geringeren Aufwand, und die manuelle Auswahl der subjektiv interessantesten Teilgruppenbeschreibungen aus der Ergebnismenge ist einfacher.

In komplexen Suchräumen, wie auch hier einer vorliegt, dauert eine erschöpfende Suche sehr lange und liefert zunehmend speziellere und damit schwer zu interpretierende und weniger operable Ergebnisse. Daher werden hier die einzelnen Suchläufe manuell abgebrochen, wenn eine ausreichend erscheinende Anzahl interessanter Teilgruppenbeschreibungen gefunden ist und die Entdeckung weiterer interessanter Ergebnisse unwahrscheinlich scheint. Durch die ebenenweise Suche vom Allgemeinen zum Speziellen ist damit in das Suchverfahren eine weiterer Bevorzugung allgemeinerer Regeln gegenüber spezielleren eingebaut.

Zusammengefaßt laufen also Suche und Akzeptanz bei dieser Anwendung ab wie folgt:

- Bewertung der Regeln nach der Folgerungsstärke,
- Akzeptanz abhängig von Schwellwerten für Folgerungsstärke und Mindesthäufigkeit,
- Ausschluß akzeptierter Hypothesen,
- manueller Abbruch eines Suchlaufs nach Durchsuchen einiger Suchebenen,
- manuelle Selektion der interessantesten unter den Hypothesen.

Die Folgerungsstärke ist ein nicht-symmetrisches Interessantheitskriterium und ergibt unterschiedliche Bewertungen für Regeln die $A \rightarrow B$ und $A \rightarrow \overline{B}$. Deshalb werden hier zu jeder der Fragestellungen „guter versus schlechter Kredit“ und „Kreditkarteninhaber versus Nichtinhaber“ zwei Suchläufe durchgeführt, einmal mit der Menge der guten Kredite als Zielgruppe und einmal mit deren Komplement „schlechte Kredite“ als Zielgruppe, beziehungsweise, einmal mit den Kreditkarteninhabern als Zielgruppe und einmal mit deren Komplement, den Kontoinhabern ohne Kreditkarte, als Zielgruppe. Die Schwellwerte für Mindestvorkommen und Folgerungsstärke werden für die verschiedenen Suchläufe jeweils so festgelegt, daß die Menge der Entdeckungen noch überschaubar bleibt.

7.1.3 Der Suchraum für „guter versus schlechter Kredit“

Der potentielle Suchraum der Finanzdaten-Anwendungen ist sehr groß im Vergleich zum Umfang der vorhandenen Daten. Die Relation **transaction** enthält mehr als 10^6 Einträge; damit kann eine riesige Auswahl an Attributen definiert werden, zum Beispiel verschiedene Muster in den Kontobewegungen als Merkmale eines Kontos. Diese Attribute beschreiben jedoch nicht mehr als 682 Kredite beziehungsweise 4500 Konto-Inhaberschaften. Daher müssen geeignete Attribute und Merkmale ausgewählt werden.

Vorbereitung der Daten

Bei der Fragestellung „schlechte versus gute Kredite“ interessieren die Kunden mit faulen bzw. problemlosen Krediten. Schlechte Kredite sind noch laufende oder schon abgeschlossene Kredite, die Probleme verursacht haben, also solche mit Status b oder d. Gute Kredite haben Status a oder c. Insgesamt gibt es Daten zu 682 Krediten, davon sind 76 schlechte Kredite. Bei der Aufbereitung der Daten für diese Aufgabenstellung wurde die Menge der Kredite als untersuchte Grundgesamtheit gewählt. Jeder Kredit identifiziert über die Relation **disposition** einen Kunden, nämlich den Inhaber des Kontos, zu dem der Kredit gehört. Da jedes Konto genau einen Inhaber hat, stehen damit auch die auf den Kontoinhaber bezogenen Daten mit den Krediten in einer 1:1–Beziehung. Um die Suche effizienter zu machen, wurde eine Verbundrelation **ur_loan** berechnet mit Attributen aus den Relationen **account**, **client**, **district** und **disposition** und mit Attributen, die aus der Relation **transaction** abgeleitet sind. Andere Attribute aus der Relation **transaction** wurden nicht verwendet. Die Verbundrelation umfaßt 682 Tupel, die je einem Kredit entsprechen.

Da eine beliebige Anzahl von Daueraufträgen auf einem Konto eingerichtet sein kann, steht die Relation **order** nicht in einer 1:1–Beziehung zu **loans**. Aus der Relation **order** stammende Attribute wurden daher nicht in die Verbundrelation aufgenommen, sondern getrennt gehalten. Auch die Relation **card** blieb separat. Für die Fragestellung „gute versus schlechte Kredite“ waren damit drei Relationen vorhanden. Die entsprechenden Verbinder lauten:

Deklaration 1

$link(ur\_loan,$ $[rel(''ur\_loan($*Loan_Id, Disp_Id, Account_Id, _, _, _, _ Status, Age_Class, Sex, Region, Urbanity, Avg_Salary, Frequency, _, AvgInc, NInc, VarInc, AvgWD, NWD, VarWD, Disp2*$)'')]).$

$link(card,$ $[rel(''card(Disp\_Id, \_, CType, \_)''),$
$from([ur\_loan])]).].$

$link(order,$ $[rel(''order(Account\_Id, Order\_Id, \_, \_, \_, K\_Symbol)''),$
$from([ur\_loan])]).$

Deklaration der Grundpopulation und Zielgruppe

Die Grundpopulation sind hier die verschiedenen Instanzen des Attributs $Loan\_Id$ in der Verbundrelation ur_loan. Die Popualtion setzt sich zusamen aus der Klasse ok, für die das Prädikat $good(Status)$ wahr ist, und der Klasse bad, für die $bad(Status)$ zutrifft. Die Prädikate $good(Status)$ und $bad(Status)$ sind im Hintergrundwissen definiert als

$good(Status) :- \quad Status = a; Status = c.$

$bad(Status) :- \quad Status = b; Status = d.$

Die Angabe $targetclass(ok)$ markiert die problemlosen Kredite als Zielgruppe. Die Deklaration sieht damit so aus:

Deklaration 2

$count(dispositions,$ $[att(''Loan\_Id'')],$ $from([ur\_loan])]).$

$class(loan,$ $[classes([$ *ok*–$''good(Status)'',$
bad–$''bad(Status)'']),$

$targetclass('ok'),$ $from([ur\_loan])]).$

Um die ungünstigen Kredite zur Zielgruppe zu deklarieren, wird $targetclass(ok)$ ersetzt durch $targetclass(bad)$.

Deklaration der Merkmale

Bei der Fragestellung "gute versus schlechte Kredite" werden die folgenden Attribute und Merkmale berücksichtigt.

• ***Age_Class*** ist das Alter des Konto-Inhabers, eingeteilt in Altersgruppen. Alle Altergruppen außer der jüngsten und der ältesten Gruppe umfassen je sieben Jahrgänge. Die Jahrgänge 77 und jünger gehören zur Altersgruppe 0, Jahrgänge 76 bis 70 bilden Altersgruppe 1 und so weiter bis zur Altergruppe 6 mit den Jahrgängen 41 bis 35 und Altersgruppe 7 mit Jahrgängen kleiner als 35.

Deklaration 3

$lit(age\_\leq\_6,$ $[def([''Age\_Class =< 6'']),$ $from([ur\_loan]),$
$excludes([age7\_7])]).$

$lit(age\_\leq\_5,$ $[def([''Age\_Class =< 5'']),$ $from([ur\_loan]),$
$requires([age\_\leq\_6]),$
$excludes([age\_\geq\_7, age\_\geq\_6])]).$

$\vdots$

$$lit(age\_\geq\_7, \quad [def(["Age\_Class >= 7"]), \quad from([ur\_loan]),$$
$$requires([age\_\geq\_6]),$$
$$excludes([age\_\leq\_0, \ldots, age\_\leq\_6])]).$$

• ***Sex*** ist das Geschlecht des Konto-Inhabers, repräsentiert als Wert des Attributs Sex.

Deklaration 4

$$lit(male, \quad [def(["Sex = m"]), \quad from([ur\_loan]),$$
$$excludes([female])]).$$
$$lit(female, \quad [def(["Sex = f"]), \quad from([ur\_loan]),$$
$$excludes([male])]).$$

• ***Region*** ist der Wohnsitz des Konto-Inhabers mit dem Wertebereich

$$prague, c\_bohemia, s\_bohemia, n\_bohemia, w\_bohemia, e\_bohemia,$$
$$s\_moravia, n\_moravia.$$

Deklaration 5

$$lit(prague, \quad [def(["Region = prague"]), \quad from([ur\_loan]),$$
$$excludes([c\_boh, \ldots, n\_mor])]).$$
$$\vdots$$
$$lit(n\_mor, \quad [def(["Region = n\_moravia"]), \quad from([ur\_loan]),$$
$$excludes([prague, c\_boh, \ldots s\_mor])]).$$

• ***Urbanity*** ist die Urbanitätsrate im Wohndistrikt des Kontoinhabers. Die Urbanitätsrate ist der Anteil der Einwohner von Städten an der Gesamtbevölkerung des jeweiligen Distrikts. Sie ist in den Daten in Prozent angegeben. Für die Anwendung sind die eigentlichen Prozentzahlen in den Daten durch die jeweilige Zehner-Ziffer ersetzt. Der Wertebereich des Attributs Urbanity ist also $\{4, 5, \ldots, 10\}$.

Dieses und alle folgenden numerischen Attribute werden wie in Abschnitt 5.2.3 beschrieben behandelt. Dazu werden Merkmale definiert, die den Wert des Attributs mit einer Intervallgrenze vergleichen. Konjunktionen mehrerer dieser Merkmale beschreiben Intervalle unterschiedlicher Länge, die durch entsprechende *req-* und *excl*–Deklarationen zur Hierarchie geordnet sind. Die Intervallgrenzen für das Attribut Urbanity sind seine Werte 4, 5, …, 10.

Deklaration 6

$$lit(urb\_\geq\_4, \quad [def(["Urbanity >= 4"]), \quad from([ur\_loan]),$$
$$excludes([urb\_\leq\_3])]).$$
$$lit(urb\_\geq\_5, \quad [def(["Urbanity >= 5"]), \quad from([ur\_loan]),$$
$$requires([[urb\_\geq\_4]]),$$
$$excludes([urb\_\leq\_3, urb\_\leq\_4])]).$$
$$\vdots$$
$$lit(urb\_\leq\_3, \quad [def(["Urbanity =< 3"]), \quad from([ur\_loan]),$$
$$requires([[urb\_\leq\_4]]),$$
$$excludes([urb\_\geq\_9, \ldots, urb\_\geq\_4])]).$$

• ***Avg_Salary*** ist das Durchschnittseinkommen im Wohndistrikt des Kontoinhabers. Im gesamten Datensatz liegen die Durchschnittseinkommen zwischen 7000 und 13000 Kronen. In den für die Anwendung aufbereiteten Daten wurden als Werte des Durchschnittseinkommens nur die vollen Tausender eingetragen. Die Intervallgrenzen sind damit $8, \ldots, 12$.

Deklaration 7

$$
\begin{array}{lll}
lit(sal\_\leq\_11, & [def([''Avg\_Salary =< 11'']), & from([ur\_loan]), \\
 & excludes([sal\_\geq\_12])]). & \\
lit(sal\_\leq\_10, & [def([''Avg\_Salary =< 10'']), & from([ur\_loan]), \\
 & requires([[sal\_\leq\_11]]), & \\
 & excludes([sal\_\geq\_12, sal\_\leq\_12])]). & \\
\vdots & & \\
lit(sal\_\geq\_12, & [def([''Avg\_Salary >= 12'']), & from([ur\_loan]), \\
 & requires([[sal\_\geq\_11]]), & \\
 & excludes([sal\_\leq\_8, \ldots, sal\_\leq\_11])]). &
\end{array}
$$

• ***Frequency*** ist die Häufigkeit der Erstellung von Kontoauszügen. Mögliche Werte sind *mesicne* (*monatlich*), *obratu* (*wöchentlich*), *tydne* (*sofort*).

Deklaration 8

$$
\begin{array}{lll}
lit(fr\_month, & [def([''Frequency = mesicne'']), & from([ur\_loan]), \\
 & excludes([fr\_imm, fr\_week])]). & \\
\vdots & &
\end{array}
$$

• ***AvgInc*** ist die durchschnittliche Summe der monatlichen Einzahlungen auf das Konto eingeteilt in Intervalle mit den Intervallgrenzen 25000 und 50000. Dieses und die beiden folgenden Attribute beschreiben Eigenschaften von Konten, die aus der Relation transactions abgeleitet sind. Sie wurden mit folgenden SQL-Anfragen berechnet (PRIJEM bedeutet Gutschrift).

```
SELECT ag.account_id, AVG(ag.msum), AVG(mcount),
       SQRT(VARIANCE(ag.msum))
FROM ( SELECT
t2.account_id AS account_id, t2.monat, SUM(t2.amount) AS msum,
COUNT(t2.amount) AS mcount FROM trans t2
WHERE type = PRIJEM GROUP BY t2.account_id, t2.monat) AS ag
GROUP BY ag.account_id"
```

Deklaration 9

$$
\begin{array}{lll}
lit(avgInc\_<\_50000, & [def([''AvgInc < 50000'']), & from([ur\_loan]), \\
 & excludes([avgInc\_>\_50000])]). & \\
lit(avgInc\_<\_25000, & [def([''AvgInc < 25000'']), & from([ur\_loan]), \\
 & requires([[avgInc\_<\_50000]]), & \\
 & excludes([avgInc\_>\_25000])]). & \\
lit(avgInc\_>\_25000, & [def([''AvgInc > 25000'']), & from([ur\_loan]), \\
 & excludes([avgInc\_<\_25000])]). &
\end{array}
$$

$$lit(avgInc\_>\_50000, \quad [def([''AvgInc > 50000'']), \quad from([ur\_loan]),$$
$$requires([[avgInc\_>\_25000]]),$$
$$excludes([avgInc\_<\_50000])]).$$

• ***VarInc*** beschreibt die Schwankungen der durchschnittlichen monatlichen Einzahlungen auf das Konto; eingeteilt in Intervalle mit den Intervallgrenzen 20000 und 40000.

Deklaration 10

$$lit(varInc\_<\_40000, \quad [def([''VarInc < 40000'']), \quad from([ur\_loan]),$$
$$excludes([varInc\_>\_40000])]).$$
$$lit(varInc\_<\_20000, \quad [def([''VarInc < 20000'']), \quad from([ur\_loan]),$$
$$requires([[varInc\_<\_40000]]),$$
$$excludes([varInc\_>\_20000])]).$$
$$lit(varInc\_>\_20000, \quad [def([''VarInc > 20000'']), \quad from([ur\_loan]),$$
$$excludes([varInc\_<\_20000])]).$$
$$lit(varInc\_>\_40000, \quad [def([''VarInc > 40000'']), \quad from([ur\_loan]),$$
$$requires([[varInc\_>\_20000]]),$$
$$excludes([varInc\_<\_40000])]).$$

• ***NInc*** ist die durchschnittliche monatliche Anzahl von Einzahlungen auf das Konto. Mögliche Werte sind $1, 2, 3, 4$. Die Werte bilden die Intervallgrenzen für hierarchisch angeordnete Intervalle.

Deklaration 11

$$lit(numInc\_\geq\_2, \quad [def([''NInc >= 2'']), \quad from([ur\_loan]),$$
$$excludes([numInc\_\leq\_1])]).$$
$$lit(numInc\_\geq\_3, \quad [def([''NInc >= 3'']), \quad from([ur\_loan]),$$
$$requires([[numInc\_\geq\_2]]),$$
$$excludes([numInc\_\leq\_2])]).$$
$$lit(numInc\_\geq\_4, \quad [def([''NInc >= 4'']), \quad from([ur\_loan]),$$
$$requires([[numInc\_\geq\_3]]),$$
$$excludes([numInc\_\leq\_3])]).$$
$$lit(numInc\_\leq\_3, \quad [def([''NInc =< 3'']), \quad from([ur\_loan]),$$
$$excludes([numInc\_\geq\_4])]).$$
$$lit(numInc\_\leq\_2, \quad [def([''NInc =< 2'']), \quad from([ur\_loan]),$$
$$requires([[numInc\_\leq\_3]]),$$
$$excludes([numInc\_\geq\_3])]).$$
$$lit(numInc\_\leq\_1, \quad [def([''NInc =< 1'']), \quad from([ur\_loan]),$$
$$requires([[numInc\_\leq\_2]]),$$
$$excludes([numInc\_\geq\_2])]).$$

• ***AvgWD*** ist die durchschnittliche Summe der monatlichen Abbuchungen von einem Konto. Dieses und die beiden folgenden Attribute entsprechen AvgInc, VarInc beziehungsweise NInc. Sie sind berechnet mit einer analogen SQL-Anfrage, wobei PRIJEM durch

VYDAJ (Abbuchung) ersetzt ist. AvgWD ist eingeteilt in Intervalle mit den Grenzen 25000 und 50000.

Deklaration 12

$$\begin{array}{lll} lit(avgWD\_{<}\_50000, & [def([''AvgWD < 50000'']), & from([ur\_loan]), \\ & excludes([avgWD\_{>}\_50000])]). & \\ \vdots & & \\ lit(avgWD\_{>}\_50000, & [def([''AvgWD > 50000'']), & from([ur\_loan]), \\ & requires([[avgWD\_{>}\_25000]]), & \\ & excludes([avgWD\_{<}\_50000])]). & \end{array}$$

• ***varWD*** beschreibt die Schwankungen in den monatlichen Abbuchungen, eingeteilt in Intervalle mit den Grenzen 20000 und 40000.

Deklaration 13

$$\begin{array}{lll} lit(varWD\_{<}\_40000, & [def([''VarWD < 40000'']), & from([ur\_loan]), \\ & excludes([varWD\_{>}\_40000])]). & \\ \vdots & & \\ lit(varWD\_{>}\_40000, & [def([''VarWD > 40000'']), & from([ur\_loan]), \\ & requires([[varWD\_{>}\_20000]]), & \\ & excludes([varWD\_{<}\_40000])]). & \end{array}$$

• ***NumWD*** ist die durchschnittliche Anzahl der Abbuchungen von einem Konto pro Monat, eingeteilt in Intervalle mit den Grenzen $2, 4, 6$.

Deklaration 14

$$\begin{array}{lll} lit(numWD\_{\leq}\_6, & [def([''NWD =< 6'']), & from([ur\_loan]), \\ & excludes([numWD\_{\geq}\_6])]). & \\ lit(numWD\_{\leq}\_4, & [def([''NWD =< 4'']), & from([ur\_loan]), \\ & requires([[numWD\_{\leq}\_6]]), & \\ & excludes([numWD\_{\geq}\_4])]). & \\ \vdots & & \\ lit(numWD\_{\geq}\_6, & [def([''NWD >= 6'']), & from([ur\_loan]), \\ & requires([[numWD\_{\geq}\_4]]), & \\ & excludes([numWD\_{\leq}\_6])]). & \end{array}$$

• ***Disp2*** gibt an, ob für ein Konto eine zweite Person verfügungsberechtigt ist, und gegebenenfalls das Geschlecht dieser Person. Der Wertebereich ist $\{-, f, m\}$. Kein Konto besitzt neben dem Konto-Inhaber mehr als einen weiteren Verfügungsberechtigten.

Deklaration 15

$$\begin{array}{lll} lit(disp2\_yes, & [def([''\backslash + Disp2 = '-''']), & from([ur\_loan]), \\ & excludes([disp2\_no])]). & \\ lit(disp2\_no, & [def([''Disp2 = '-''']), & from([ur\_loan]), \\ & excludes([disp2\_yes])]). & \end{array}$$

$$
\begin{array}{lll}
lit(disp2\_female, & [def([''Disp2 = f'']), & from([ur\_loan]), \\
 & excludes([disp2\_male]), & \\
 & requires([[disp2\_yes]])]). & \\
lit(disp2\_male, & [def([''Disp2 = m'']), & from([ur\_loan]), \\
 & excludes([disp2\_female]), & \\
 & requires([[disp2\_yes]])]). &
\end{array}
$$

• ***Order:*** aus der Relation Order lassen sich die Merkmale no_order und has_order gewinnen, die besagen, daß zu einer Kontonummer ein Eintrag in der Relation order existiert beziehungsweise nicht existiert.

Deklaration 16

$$
\begin{array}{lll}
lit(no\_order, & [def([''\backslash+ order(Account\_Id, \_, \_, \_, \_, \_)'']), & from([]), \\
 & excludes([has\_order])]). & \\
lit(has\_order, & [def([''order(Account\_Id, \_, \_, \_, \_, \_)'']), & from([]), \\
 & excludes([no\_order])]). &
\end{array}
$$

• ***K_Symbol*** in der Relation order beschreibt den Typ eines Dauerauftrags. Die Typen sind $\{sipo(haushalt), uver(Kredit), pojistne(Versicherung), leasing, unbekannt\}$. Dieses Attribut wird nur für Konten abgefragt, für die ein Eintrag in der Relation order vorkommt. Weil auf einem Konto mehrere Daueraufträge verschiedenen Typs eingerichtet sein können, sind die Merkmale, die den Wert von K_Symbol angeben, nicht als sich gegenseitig ausschließend deklariert.

Deklaration 17

$$
\begin{array}{lll}
lit(order\_household, & [def([''K\_Symbol = sipo'']), & from([order]), \\
 & requires([[has\_order]])]). & \\
\vdots & & \\
lit(order\_unknown, & [def([''K\_Symbol = \ ' '']), & from([order]), \\
 & requires([[has\_order]])]). &
\end{array}
$$

• ***Card:*** Für die Fragestellung „guter versus schlechter Kredit“ bietet die Relation Card Informationen darüber, ob der Kontoinhaber mit einer Kreditkarte über sein Konto verfügt oder nicht.

Deklaration 18

$$
\begin{array}{lll}
lit(has\_card, & [def([''card(Disp\_Id, \_, \_, \_)'']), & from([ur\_loan]), \\
 & excludes([no\_card])]). & \\
lit(no\_card, & [def([''\backslash+ card(Disp\_Id, \_, \_, \_)'']), & from([ur\_loan]), \\
 & excludes([has\_card])]). &
\end{array}
$$

• ***CType*** gibt den Typ einer Kreditkarte an. Die möglichen Typen sind $gold$, $classic$, $junior$. Der Typ einer Kreditkarte wird als Spezialisierung des Merkmals has_card deklariert. Da zu jedem Konto höchstens eine Kreditkarte ausgegeben ist, schließen sich die Merkmale gegenseitig aus.

Deklaration 19

$$
\begin{array}{lll}
lit(junior, & [def([''CType = junior'']), & from([ur\_loan, card]), \\
 & requires([[has\_card]]), & \\
 & excludes([classic, gold])]). & \\
\ldots & & \\
lit(gold, & [def([''CType = gold'']), & from([ur\_loan, card]), \\
 & requires([[has\_card]]), & \\
 & excludes([classic, junior])]). &
\end{array}
$$

7.1.4 Der Suchraum für „Kreditkarteninhaber versus Nichtinhaber“

Vorbereitung der Daten

Bei der Fragestellung „Kreditkarteninhaber versus Nichtinhaber“ stellen die Kunden die interessierenden Einheiten dar. Überraschenderweise verfügt im vorliegenden Datensatz kein Kunde über mehr als ein Konto, so daß die Relation **disposition** jeden Kunden genau einem Konto zuordnet. Auch sind in diesem Datensatz alle Karteninhaber zugleich Inhaber eines Kontos, so daß kein Kunde, der als Nutzer über ein fremdes Konto verfügen kann, eine Kreditkarte besitzt. Da die Tupel der Relation **disposition** mit der Eigenschaft $Type = owner$ eindeutig einen Kontoinhaber und damit potentiellen Kreditkartenbesitzer identifizieren, bilden diese Tupel hier die Grundgesamtheit. Die Zielgruppe bilden die Dispositionen, die auch in der Relation **card** vorkommen. Für diese Fragestellung wurde eine Verbundrelation **ur_disp** mit Attributen erzeugt, die in einer 1:1–Beziehung zu Kreditkartenbesitzern stehen. Da es zu jedem Konto höchstens einen Kredit gibt, können Attribute der Relation **loan** ebenfalls in die Verbundrelation aufgenommen werden. Die Verbundrelation **ur_disp** enthält 4500 Tupel, die je einer Disposition mit $Type = owner$ entsprechen. Hier wurden die Relationen **order** und **loan** nicht mitverbunden. Folgende Verbinder wurden für die Kreditkarten-Suchläufe festgelegt:

$$
\begin{array}{ll}
link(ur\_disp, & [rel(''ur\_disp(\textit{Disp\_Id, Account\_Id, Age\_Class, Sex, Region,} \\
 & \textit{Urbanity, Avg\_Salary, Frequency, \_, AvgInc, NInc, VarInc,} \\
 & \textit{AvgWD, NWD, VarWD, Disp2 , \_, Status, CType, \_})'')]). \\
link(order, & [rel(''order(Account\_Id, \_, \_, \_, K\_Symbol)''), \\
 & from([ur\_disp])]).
\end{array}
$$

Deklaration der Grundpopulation und Zielgruppe

Hier umfaßt die Grundpopulation die verschiedenen Instanzen des Attributs $Disp\_Id$ in der Verbundrelation ur_disp. Sie ist unterteilt in die Gruppe $card$ der Kreditkarteninhaber, für die $Ctype$ ungleich $'-'$ ist, und deren Komplement no_card, für die $CType = '-'$ gilt. Für die Fragestellung „Kreditkarteninhaber versus Nichtinhaber“ werden Suchläufe mit Kreditkarteninhabern als Zielgruppe durchgeführt (deklariert durch $targetclass(card)$) und Suchläufe mit Nichtinhabern als Zielgruppe ($targetclass(no\_card)$). Die Deklarationen lauten:

Deklaration 20

$$
\begin{array}{lll}
count(dispositions, & [att(''Disp\_Id'')], & from([ur\_disp])]). \\
class(card, & [classes([& \textit{card–}'' \backslash + CType =' -''', \\
 & & \textit{no\_card–}''CType =' -''']), \\
 & targetclass('card'), & from([ur\_disp])]).
\end{array}
$$

Deklaration der Merkmale

Die Attribute und die darauf definierten Merkmale für diese Fragestellung stimmen bis auf wenige Unterschiede mit denen der Fragestellung „schlechte versus gute Kredite“ überein. Im $from$-Teil der Deklarationen muß ur_loan ersetzt werden durch ur_disp. Nicht verwendet werden die Merkmale has_card, no_card, $junior$, $classic$, $gold$. Zusätzliche Merkmale für die Fragestellung „Kreditkarteninhaber versus Nichtinhaber“ sind aus dem Attribut $Status$ abgeleitet.

• ***Status*** gibt den Status eines möglicherweise auf dem Konto laufenden Kredits an. Die möglichen Werte sind $\{a, b, c, d, -\}$. Der Wert $-$ bedeutet, daß kein Kreditvertrag besteht. Auf dem Wertebeich wird eine Taxonomie definiert. Das verallgemeinerte Merkmal $loan\_ok$ subsumiert $Status = a$ und $Status = c$. Das verallgemeinerte Merkmal $loan\_bad$ subsumiert $Status = b$ und $Status = d$. Der Status eines Kredits und das Merkmal no_loan werden nicht gleichzeitig für ein Konto abgefragt.

Deklaration 21

$$
\begin{array}{lll}
lit(no\_loan, & [def([''Status = -'']), & from([ur\_disp]), \\
 & excludes([rloan\_bad, \ldots floan\_bad])]). & \\
lit(loan\_ok, & [def([''(Status = c; Status = a)'']), & from([ur\_disp]), \\
 & excludes([loan\_bad, no\_loan])]). & \\
lit(loan\_bad, & [def([''(Status = d; Status = b)'']), & from([ur\_disp]), \\
 & excludes([loan\_ok, no\_loan])]). & \\
lit(rloan\_ok, & [def([''Status = c'']), & from([ur\_disp]), \\
 & requires([loan\_ok]), & \\
 & excludes([rloan\_bad, \ldots, floan\_bad])]). & \\
\vdots & & \\
lit(floan\_bad, & [def([''Status = b'']), & from([ur\_disp]), \\
 & requires([loan\_bad]), & \\
 & excludes([rloan\_bad, \ldots, floan\_ok])]). &
\end{array}
$$

7.2 Ergebnisse

7.2.1 „Gute versus schlechte Kredite“

Problematische Kredite als Zielgruppe

Die Grundpopulation dieser Fragestellung bilden die 682 in der Datenbank beschriebenen Kredite. Die Zielgruppe besteht aus den 76 problematischen Krediten. Der Anteil der Zielgruppe an der Gesamtpopulation liegt bei 11.1 %. Bei dieser Fragestellung muß eine Regel für mindestens 10 Fälle zutreffen und eine Folgerungsstärke von mindestens 0.90 erreichen, um akzeptiert zu werden. Eine mehrstündige Suche durch die Regeln mit bis zu 5 Merkmalen im Wenn-Teil erbrachte 47 Regeln, davon 16 Regeln mit 1 bis 3 Attributen. Diese erschienen subjektiv am interessantesten. Sie sind in Tafel 7.1 wiedergegeben. Für jede Regel $i \in cov(h) \rightarrow i \in T$ sind die Folgerungsstärke $I(h)$ und der Zielgruppenanteil $r = cov(h) \cap T$ angegeben, r in der Form $r = |cov(h) \cap T|/|cov(h)|$, so daß die absoluten Anzahlen der von h abgedeckten Zielgruppenelemente und der von h insgesamt abgedeckten Elemente ersichtlich sind.

Tafel 7.1 Ausgewählte interessante Regeln für die Fragestellung „gute versus schlechte Kredite“ mit problematischen Krediten als Zielgruppe.

Population: Kredite Zielgruppe: Faule Kredite
Anteil der Zielgruppe r = 0.111 = 76 / 682

1.	$VarInc\_>\_20000$	I = 0.946 r= 0.206 = 53 / 257
2.	$disp2\_no, VarWD\_>\_20000$	I= 0.944 r= 0.217 = 44 / 203
3.	$disp2\_no, no\_card$	I= 0.916 r= 0.176 = 71 / 403
4.	$VarInc\_<\_40000, VarWD\_>\_20000$	I= 0.925 r= 0.209 = 40 / 191
5.	$AvgInc\_<\_50000, VarWD\_>\_20000$	I= 0.904 r= 0.198 = 40 / 202
6.	$VarWD\_>\_20000, no\_card$	I= 0.953 r= 0.234 = 39 / 167
7.	$VarWD\_>\_20000, female$	I= 0.928 r= 0.236 = 29 / 123
8.	$disp2\_no, numInc\_\geq\_3$	I= 0.934 r= 0.268 = 22 / 82
9.	$AvgInc\_<\_50000, numInc\_\geq\_3$	I= 0.909 r= 0.253 = 20 / 79
10.	$AvgInc\_>\_25000, female, w\_boh$	I= 0.905 r= 0.412 = 7 / 17
11.	$urb\_>\_4, sal\_<\_12, VarWD\_>\_20000$	I= 0.907 r= 0.198 = 41/207
12.	$sal\_>\_90000, numInc\_\geq\_3$	I= 0.936 r= 0.304 = 17/56
13.	$no\_card, numInc\_\geq\_3$	I= 0.927 r= 0.266 = 21/79
14.	$AvgWD\_<\_50000, numInc\_\geq\_3$	I= 0.903 r= 0.239 = 22 / 92
15.	$AvgInc\_<\_25000, VarWD\_<\_50000$	I= 0.97 r= 0.424 = 14 / 33
16.	$VarWD\_>\_20000, NumWD\_<\_4$	I= 0.909 r= 0.202 = 39 / 193

Tafel 7.2 Beispiele für spezielle akzeptierte Regeln für die Fragestellung „gute versus schlechte Kredite“ mit problematischen Krediten als Zielgruppe.

17. $w\_boh, no\_card, has\_order, order\_unknown, AvgInc\_>\_25000.$
I = 0.949 r= 0.8 = 4 / 5

18. $AvgSal\_<\_11, varWD\_>\_20000, numWD\_\leq\_6, numWD\_\geq\_2.$
I= 0.944 r= 0.217 = 44 / 203

Diese Anzahlen sind für jede Regel $i \in cov(h) \rightarrow i \in T$ zusätzlich graphisch repräsentiert in Form eines Balkens, dessen gesamte Länge der Anzahl $|cov(h)|$ der von h abgedeckten Elemente entspricht. Die Länge des hellen Teils des Balkens entspricht der Zahl der von h abgedeckten Zielgruppenelemente, $|cov(h) \cap T|$, und die Länge des dunklen Teils der Anzahl der Individuen, die der Regel widersprechen, $|cov(h) \cap \overline{T}|$. Die kleine senkrechte Markierung bezeichnet den Zielgruppenanteil, der bei Unabhäbhängigkeit zwischen Voraussetzung $i \in cov(h)$ und Folgerung $i \in T$ der Regel zu erwarten wäre. Je weiter die senkrechte Markierung von der Trennlinie zwischen dem hellen und dem dunklen Balkenteil entfernt ist, desto stärker weicht die Regel von der Erwartung ab und desto interessanter ist sie. Dies bietet ein zusätzliches Kriterium für die Interessantheit einer Regel.

Regel 1 besagt, daß in der Gruppe der Kreditnehmer mit mittleren oder hohen Schwankungen monatlicher Gutschriften der Anteil fauler Kredite recht hoch ist, nämlich 20.6% im Vergleich zu 11.1% in der Gesamtpopulation. Von insgesamt 76 faulen Krediten gehören 53 (die Unterstützer der Regel) zu dieser Gruppe. Diese Regel könnte dahingehend interpretiert werden, daß ein schwankendes Einkommen auf unzuverlässige Kunden hinweist.

Regel 2 macht eine ähnliche Aussage über Kunden mit mittleren oder hohen Schwankungen der monatlichen Ausgaben, für deren Konto es keinen zweiten Nutzer gibt. Hier liegt der Anteil fauler Kredit mit 21.7% sogar noch höher, doch trifft diese Regel für etwas weniger Fälle als Regel 1 zu. 44 der 76 problematischen Kredite gehören zu dieser Gruppe. Regel 3 beschreibt Kontoinhaber ohne Kreditkarte und ohne Konto-Mitnutzer. Diese Regel trifft für 71 der 76 schlechten Kredite zu. Das Zusammentreffen von problematischem Kredit und Fehlen einer Kreditkarte kann allerdings auch durch die Richtlinien der Bank für die Vergabe von Kreditkarten verursacht sein; mit diesem Hintergrundwissen wäre diese Regel nicht so interessant, wie die Kriterien Folgerungsstärke und Zielgruppenanteil nahelegen.

Regel 4 zeigt, daß der Anteil schlechter Kredite bei den Konten mit niedriger oder mittlerer Schwankung des monatlichen Einkommens, aber mittlerer oder hoher Schwankung der monatlichen Ausgaben auf 20.9 % steigt. Auch die Regeln 5 bis 7 deuten an, daß das Risiko schlechter Kredite bei mittlerer oder hoher Schwankung der monatlichen Ausgaben unter bestimmten Bedingungen steigt, nämlich wenn kein Spitzeneinkommen vorliegt (Regel 5) oder kein zweiter Kontonutzer existiert (Regel 6) oder bei weiblichen Kontoinhabern (Regel 7). Regeln 8 und 9 besagen, daß das Kreditrisiko erhöht ist für Konten, bei denen die monatlichen Einzahlungen auf drei oder vier Buchungen verteilt sind und außerdem kein zweiter Kontonutzer existiert (Regel 8) oder außerdem das Gesamteinkommen nicht zum oberen Drittel gehört (Regel 9). Die Regel 10 schließlich betrifft nur eine geringe Zahl von Fällen (nämlich 17), ist aber deshalb überraschend, weil hier bei einer bestimmten Gruppe, nämlich weiblichen Kontoinhabern in Westböhmen, der Anteil der schlechten Kredite trotz mittlerem oder hohem Einkommen auf 41.2 % steigt.

Die Regeln mit mehr Merkmalen, die bei diesem Suchlauf akzeptiert wurden, decken meist nur wenige Fälle ab und sind zumindest ohne weitergehende Kenntnisse des Anwendungsgebiets kaum zu interpretieren. Zwei Beispiele für solche spezielleren Regeln zeigt Tafel 7.2.

Tafel 7.3 Ausgewählte akzeptierte Regeln für die Fragestellung „gute versus schlechte Kredite“ mit gut verlaufenden Krediten als Zielgruppe.

Population: Kredite Zielgruppe: Gute Kredite
Anteil der Zielgruppe r = 0.889 = 606 / 682

1. $disp2\_yes$
I= 1 r= 1.000 = 145 / 145

2. has_card
I= 0.999 r= 0.971 = 165 / 170

3. $has\_order, order\_household$
I= 1 r= 0.955 = 421 / 441

4. $VarInc\_<\_20000$
I= 1 r= 0.946 = 402 / 425

5. $VarWD\_<\_20000$
I= 0.992 r= 0.927 = 405 / 437

6. $numWD\_\geq\_4$
I= 0.999 r= 0.941 = 351 / 373

7. $has\_order, order\_unknown, male$
I= 0.985 r= 0.956 = 109 / 114

8. $AvgInc\_<\_25000, has\_order, order\_unknown$
I= 0.974 r= 0.949 = 111 / 117

Gut verlaufende Kredite als Zielgruppe

Hier werden typischen Eigenschaften oder Verhaltensweisen von Inhabern gut verlaufender Kredite gesucht. Die betrachtete Grundpopulation besteht also wieder aus allen in der Datenbank beschriebenen Krediten, jedoch bilden hier die guten Kredite die Zielgruppe. Entsprechend ist hier der Apriori-Anteil der Zielgruppe sehr hoch, er beträgt 88.9 %. Um die Anzahl akzeptierter Regeln einzuschränken, sind hier mindestens 10 %, das sind 68 Fälle, als Mindestabdeckung für eine akzeptierte Regel verlangt. Die Schwelle für die Folgerungsstärke kann hier streng auf 0.96 festgelegt werden. Mit diesen Einstellungen werden insgesamt 20 Regeln mit bis zu 4 Merkmalen akzeptiert.

Tafel 7.3 listet eine subjektive Auswahl aus den akzeptierten Regeln auf. Regel 1 besagt, daß die Existenz eines zweiten Konto-Zugriffsberechtigten auf einen problemlosen Kreditverlauf hinweist. Diese Regel zeichnet sich dadurch aus, daß sie ausschließlich Zielgruppenfälle abdeckt, das heißt, zu dieser Regel existieren keine widersprüchlichen Fälle

in der Datenbank. Dies ist eine sehr auffällige Entdeckung.

Die zweite Regel beschreibt den Zusammenhang zwischen Kreditkarteninhaberschaft und problemlosem Kreditverlauf. Dieser Zusammenhang kann, wie bereits erwähnt, von der Kreditkartenvergabestrategie der Bank herrühren.

Laut Regel 3 gibt es eine Tendenz, daß Kredite ohne Schwierigkeiten verlaufen, wenn auf dem betreffenden Konto ein Dauerauftrag mit Zweck $household$ eingerichtet ist. Die Regeln 4 und 5 bedeuten, daß eine niedrige Schwankung der monatlichen Gutschriften beziehungsweise Abbuchungen auf einen guten Kreditverlauf hinweisen. Auch deutet es nach Regel 6 auf einen guten Kreditverlauf hin, wenn die Anzahl der monatlichen Abbuchungen eher hoch ist, nämlich größer als 4. Die Regeln 7 und 8 stellen einen Zusammenhang zwischen günstigem Kreditverlauf und Daueraufträgen her, diesmal mit Daueraufträgen unbekannten Typs. Regel 8 scheint deshalb interessant, weil sie Kunden mit eher niedrigen monatlichen Einnahmen betrifft. Insgesamt könnten die Regeln 3 bis 8 so interpretiert werden, daß sie Kunden mit gut organisiertem, überlegtem und damit zuverlässigem Finanzverhalten beschreiben. Diese Regeln ergänzen die Regeln, die sich für die Zielgruppe „schlechte Kredite“ ergaben.

7.2.2 „Kreditkarteninhaber versus Nichtinhaber“

Kreditkarteninhaber als Zielgruppe

Bei diesem Suchlauf bilden die Kontoinhaber vom Typ $owner$ die Grundpopulation mit den Kreditkarteninhabern als Zielgruppe. Das Ziel dabei ist, potentielle neue Kreditkarteninhaber zu identifizieren, nämlich solche, für die die gefundenen Regeln zwar zutreffen, die aber noch nicht über eine Kreditkarte verfügen. Die Grundpopulation umfaßt hier 4500 Kunden, davon gehören 892 zur Zielgruppe. Die Schwellwerte sind festgelegt auf 0.95 als minimale Folgerungsstärke und 360 Fälle als minimal erlaubte Abdeckung. Damit liefert die Suche innerhalb der ersten 4 Suchebenen nur eine geringe Anzahl akzeptabler Regeln. Diese sind in Tafel 7.4 aufgelistet.

Die Regeln 1 bis 4 geben an, daß mittlere oder hohe durchschnittliche monatliche Gutschriften oder Abbuchungen sowie mittlere oder hohe Schwankungen derselben darauf hinweisen, daß eine Kreditkarte für das jeweilige Konto existiert. Diese Zusammenhänge, wie auch Regel 5, die aussagt, daß jüngere Kontoinhaber eher eine Kreditkarte nutzen, sind vermutlich nicht besonders neu oder überraschend. Regel 6 beschreibt wieder den Zusammenhang, daß günstige Kreditverläufe auf eine Kreditkarte hinweisen, der wohl durch die Kreditkartenvergabestrategie der Bank zustandekommt. Regel 7 besagt, daß eher eine Kreditkarte existiert, wenn keine Daueraufträge eingerichtet sind, und Regel 8, daß eher eine Kreditkarte vorhanden ist, wenn die monatlichen Einnahmen durchschnittlich in mehr als zwei Gutschriften pro Monat eingehen.

Kontoinhaber ohne Kreditkarte als Zielgruppe

In dieser zur vorhergehenden komplementären Fragestellung, bei der die Kontoinhaber ohne Kreditkarte die Zielgruppe bilden, ist der Anteil der Zielgruppe mit 3608 an der 4500 Einheiten umfassenden Grundpopulation sehr hoch. Dies entspricht einem Apriori-Anteil der Zielgruppe von 80.2%. Der Schwellwert für die minimale Abdeckung beträgt hier ebenfalls 360 Fälle. Trotz des mit 0.99 sehr hoch festgelegten Schwellwerts für die Folgerungsstärke liefert die Suche sehr viele akzeptable Regeln. Diese sind oft annähernd komplementär zu den im vorangegangenen Suchlauf entdeckten Regeln.

Tafel 7.5 zeigt eine kleine Auswahl. Regel 1 besagt, daß für die Konten, auf denen ein Dauerauftrag vom Typ $household$ eingerichtet ist, eher keine Kreditkarte ausgegeben wurde. Ebenso haben nach den Regeln 2 und 3 Konten mit niedrigen monatlichen Einnahmen bzw. niedrigen monatlichen Ausgaben eher keine Kreditkarte.

Tafel 7.4 Akzeptierte Regeln für die Fragestellung „Kreditkarteninhaber versus Nichtinhaber“ mit Kreditkarteninhabern als Zielgruppe.

Population: Kontoinhaber
Zielgruppe: Kontoinhaber mit Kreditkarte
Anteil der Zielgruppe: r = 0.198 = 892 / 4500

1. $AvgInc\_>\_25000$
I= 1 r= 0.384 = 413 / 1075

2. $AvgWD\_>\_25000$
I= 1 r= 0.400 = 346 / 865

3. $VarWD\_>\_20000$
I= 1 r= 0.327 = 219 / 669

4. $VarInc\_>\_20000$
I= 0.994 r= 0.283 = 200 / 707

5. $Age\_Class\leq\_6$
I= 0.982 r= 0.229 = 875 / 3825

6. $loan\_ok$
I= 0.979 r= 0.272 = 165 / 606

7. no_order
I= 0.961 r= 0.256 = 190 / 742

8. $numInc\_\geq\_2$
I= 0.953 r= 0.224 = 765 / 3417

Falls die Informationen bisher nicht bekannt waren, könnten die Regeln 4 bis 7 interessant sein, da sie die Bank möglicherweise anregen, die Ursachen einer niedrigen Verbreitung von Kreditkarten in Mähren zu zu ermitteln und die Bankdienste entsprechend zu verbessern.

7.2.3 Diskussion der Ergebnisse

Das Verfahren hat für alle untersuchten Aufgabenstellungen potentiell interessante und nützliche Regeln entdeckt. Die Aufgabenstellung „gute versus problematische Kredite“ scheint erfolgreicher gelöst zu sein und interessantere Regeln ergeben zu haben. Besonders die abgeleiteten Attribute über Kontobewegungen kommen in vielen Regeln vor und weisen zusammen mit anderen Attributen auf typische Eigenschaften und Verhaltensweisen zuverlässiger und gut organisierter oder eher unzuverlässiger Kunden hin.

Tafel 7.5 Ausgewählte akzepierte Regeln für die Fragestellung „Kreditkarteninhaber versus Nichtinhaber“ mit Nichtinhabern als Zielgruppe.

Population: Kontoinhaber Zielgruppe: Kontoinhaber ohne Kreditkarte
Anteil der Zielgruppe: r = 0.802 = 3608 / 4500

1. $order\_household$
   I= 0.999 r= 0.826 = 2778 / 3365

2. $AvgInc\_<\_25000$
   I= 1 r= 0.860 = 2946 / 3425

3. $AvgWD\_<\_25000$
   I= 0.993, r= 0.850 = 3089 / 3635

4. $VarInc\_<\_40000, n\_mor$
   I= 0.991 r= 0.840 = 630 / 750

5. $AvgInc\_<\_50000, s\_mor$
   I= 0.993 r= 0.842 = 643 / 764

6. $Age\_Class\geq\_1, s\_mor$
   I= 0.993 r= 0.844 = 593 / 703

7. $Age\_Class\geq\_1, n\_mor$
   I= 0.993, r= 0.843 = 595 / 706

Für die Aufgabenstellung „Kreditkarteninhaber versus Nichtinhaber“ lieferte die Suche zum Teil Regeln, die bereits bekannt sein dürften, zum Beispiel, daß Kreditkartennutzer eher jung sind, über mehr als niedriges Durchschnittseinkommen verfügen und entsprechend auch mittlere oder hohe Ausgaben aufweisen. Interessant ist hier vielleicht, daß die Kunden mit Daueraufträgen vom Typ $household$ wenig Kreditkarten nutzen, während das Nichtvorhandensein von Daueraufträgen auf Kreditkartennutzung hinweist. Aus der Aufgabenstellung „guter versus problematischer Kreditverlauf“ ist bekannt, daß gerade die Kunden, die einen Dauerauftrag vom Typ $household$ eingerichtet haben, als zuverlässiger einzustufen sind. Diese Zusammenhänge könnten auf ein Image-Problem für Kreditkartennutzung hinweisen, etwa derart, daß Kunden der Meinung sind, daß Kreditkarten zu unüberlegtem Umgang mit Geld verleiten. Diesem Problem könnte die Bank mit einer entsprechenden Werbekampagne begegnen.

Bei der Auswertung der Ergebnisse fällt auf, daß gerade die demographischen Attribute, die zusätzliche Hintergrundinformation bereitstellen, in den interessanten Regeln kaum auftauchen. Dies liegt mit daran, daß diese Attribute vor allem in längeren und spezielleren akzeptierten Regeln vorkommen, die ohne genauere Sachkenntnis nur schwer zu interpretieren sind. Eine Erklärung dafür könnte sein, daß diese Attribute die Kunden zu indirekt beschreiben, da sie Eigenschaften des Wohnbezirks eines Kunden, nicht seine individuellen

Eigenschaften sind. Besonders für die Fragestellung „guter versus problematischer Kreditverlauf“ ist die Datenbasis mit 782 betrachteten Fällen für solche statistischen Merkmale sehr klein.

Bei der subjektiven Auswahl der interessanten unter den akzeptierten Regeln wurden hier kurze Regeln bevorzugt. Diese erscheinen interessanter, weil sie einleuchtender und verständlicher sind. Andererseits dürften diese Regeln den Fachgebietsexperten häufig bereits bekannt sein, gerade weil sie intuitiv und leicht zu entdecken sind. Um dies im Einzelfall zu beurteilen, ist jedoch mehr Fachwissen nötig, als hier zur Verfügung steht. In der Praxis ist es jedoch sicherlich sinnvoll, auch längere Regeln mit mehr Bedingungen in die Suche miteinzubeziehen. Eine verbesserte, systematische Vorauswahl von Attributen und die Definition weiterer abgeleiteter Attribute in den frühen Phasen des Data Mining-Prozesses, möglicherweise auch der Einsatz fortgeschrittener Diskretisierungsmethoden [DKS95], könnten zu weiteren interessanten und möglicherweise besser interpretierbaren Resultaten verhelfen. Allerdings wirkt das hier gewählte Vorgehen zur Behandlung numerischer Attribute möglichen Schwächen des eingesetzten Diskretisierungsverfahrens dadurch entgegen, daß Intervalle unterschiedlicher Länge (vom Allgemeinen zu Speziellen) erzeugt und ausgewertet werden.

7.3 Weitere Beiträge zum Discovery Challenge

Insgesamt befaßten sich sieben der Teilnehmer(gruppen) am Discovery Challenge mit den Finanzdaten. Die Bearbeitungen unterscheiden sich in der Definition der Fragestellungen, den eingesetzten Verfahren, den erzielten Ergebnissen und auch deren Präsentation. Die meisten dieser Bearbeitungen behandeln unter anderen oder ausschließlich die Fragestellung „guter versus schlechter Kredit“. Diese Fragestellung bietet sich daher für einen Vergleich der jeweils erzielten Ergebnisse an.

Guha

Ziele. Die Guha-Gruppe [CHA99] bearbeitete die Fragestellung, Charakteristika guter Kunden mit problemlosem Kreditverlauf oder schlechter Kunden mit problematischem Kreditverlauf zu finden.

Das verwendete Verfahren Guha (das bedeutet „general unary hypotheses automaton“) führt eine erschöpfende Suche im Hypothesenraum durch und sammelt die interessanten Hypothesen. Hypothesen sind Regeln mit Voraussetzungs- und Folgerungsteil, die beide aus konjunktiv verknüpften Aussagevariablen bestehen. Während die meisten gängigen Data Mining-Verfahren Einzelhypothesen bewerten, beinhaltet Guha statistische Methoden, die Aussagen über die Validität ganzer Hypothesenmengen treffen können. Das Kriterium zur Bewertung der Hypothesen ist der Fisher-Test. Dieser vergleicht die bedingte Wahrscheinlichkeit für die Folgerung B, wenn die Voraussetzung A vorliegt, $P(B/A)$, mit der unbedingten Wahrscheinlichkeit der Folgerung $P(B)$.

Für den Discovery Challenge hat die Guha-Gruppe den Hypothesenraum auf Regeln beschränkt, deren Voraussetzungsteil aus nur einer oder zwei Bedingungen besteht. Zur zusätzlichen Bewertung der Qualität akzeptierter Regeln wurde hier der Anteil der Zielobjekte an den abgedeckten Objekten herangezogen.

Resultate. Diese Bearbeitung ist sehr ähnlich zu den oben beschriebenen Experimenten, zum Teil wurden jedoch andere Attribute als in der oben beschriebenen Anwendung berechnet und untersucht, so daß andere Ergebnisse gefunden wurden. Einige Zusammenhänge, zum Beispiel $Order = household \rightarrow loan = ok$ wurden in beiden Bearbeitungen entdeckt. Die auffällige Regel $zweiterDisponent \rightarrow loan = ok$ wurde auch von Guha gefunden. Als wichtigste Regel mit einstelliger Voraussetzung für die Beschreibung

gut verlaufender Kredite liefert Guha „günstiger Kreditverlauf, wenn in der Vergangenheit keine Überziehungszinsen für das Konto gezahlt wurden“. In den Guha-Experimenten wurden Regeln mit einer oder zwei Bedingungen im Voraussetzungsteil gesucht. Da hier mehr und zusätzliche Attribute zur Verfügung standen, konnte das System einige Regeln der Länge 2 mit 100% Genauigkeit entdecken, die also nur für Elemente der Zielgruppe zutreffen.

Wizwhy

Ziele. Die Anwender (und Vertreiber) von Wizwhy [LMC+99] befassen sich mit dem Data Mining-Ziel, den Bankkunden zusätzliche Dienste anzubieten, und schlagen drei Data Mining-Aufgabenstellungen vor. Erstens die Vorhersage von problematischen Krediten, zweitens die Erkennung von typischen Kreditkarteninhabern, die jedoch noch keine Kreditkarte haben, und drittens die Erkennung von Kunden, die typischerweise Daueraufträge nutzen, aber derzeit noch keine Daueraufträge eingerichtet haben.

Das verwendete Verfahren Wizwhy wird beschrieben als ein Data Mining-Werkzeug, das einen proprietären Assoziationsregel-Algorithmus dazu verwendet, alle Wenn-Dann-Regeln in einer Datenmenge zu entdecken, die vorgegebene Parametergrenzwerte erreichen. Auf der Basis der entdeckten Regeln faßt Wizwhy die Daten zusammen, weist auf interessante Phänomene hin und trifft Vorhersagen für neue Fälle. Die Aufbereitung der Daten erfolgte mit SQL.

Das von Wizwhy produzierte Ergebnis ist im Artikel [LMC+99] wiedergegeben. Es besteht aus Tabellen mit Objekten aus der Grundpopulation, die die festgesetzten Grenzwerte erreichen, wobei für jedes einzelne der Objekte die Wahrscheinlichkeit angegeben ist, mit der das Zielverhalten (zum Beispiel Probleme bei der Kreditrückzahlung) eintritt. Für jedes Objekt kann Wizwhy die Regeln angeben, aus denen die Wahrscheinlichkeiten abgeleitet sind. Davon sind jedoch nur einzelne exemplarisch abgedruckt.

Resultate. Die Ergebnisse sind hier schwer zu beurteilen oder zu vergleichen, weil der Artikel nicht die gefundenen allgemeinen Regeln auflistet, sondern Listen der Konto-Ids, für die das System eine bestimmte Vorhersage macht. Als wichtigste Regel für die Erkennung ungünstig verlaufender Kredite wird die Regel genannt, deren Negation auch Guha entdeckte: „ungünstiger Kreditverlauf, wenn in der Vergangenheit Überziehungszinsen für das Konto gezahlt wurden“.

Mikšovský et al.

Ziele. Diese Arbeitsgruppe untersuchte zwei Fragestellungen [MŽŠP99]. Erstens versuchte sie, Beziehungen zwischen verschiedenen Bankfilialen zu entdecken, etwa Abhängigkeiten zwischen Zu- und Abnahmen der durchschnittlichen monatlichen Einlagen bei Bankfilialen in verschiedenen Regionen. Ziel war, Gruppen von Regionen mit ähnlichem Verhalten oder Regionen mit vom Durchschnitt abweichendem Verhalten zu bestimmen. Zweitens beschäftigte sie sich auch mit den Unterschieden zwischen problematischen und gut verlaufenden Krediten.

Die verwendeten Data Mining-Verfahren sind hier das ILP-Verfahren Progol und der Entscheidungsbaum-Lerner C5.0. Das ILP-Verfahren Progol setzte die Arbeitsgruppe ein für das Data Mining-Ziel, Verhalten von Bankfilialen zu untersuchen. Da dies keine interpretierbaren Ergebnisse lieferte, folgte ein Versuch mit einfachen Visualisierungsmethoden mit Microsoft Excel. Diese bestätigten, daß es keine offensichtlichen Abhängigkeiten zwischen Bankfilialen gibt.

Zur Vorhersage des Kreditverlaufs wurde der Entscheidungsbaum-Lerner C5.0 auf Daten aus den letzten sechs Monaten vor Kreditbeginn angewendet. Die Aufbereitung der Daten, Konsistenz-Prüfung und Voruntersuchung erfolgte hier mithilfe einer SQL-fähigen Datenbank, nämlich Borland Paradox.

Resultate. Die Datenaufbereitung und Voruntersuchung liefert einige Erkenntnisse über Auffälligkeiten in den Daten, zum Beispiel, daß Verzugszinsen verbucht wurden, ohne daß vorher ein negativer Kontostand auftrat.

Zur Vorhersage schlechter Kredite wurde zunächst ein Entscheidungsbaum gelernt und anschließend der Lernprozeß wiederholt, wobei nur die Attribute Verwendung fanden, die im ersten Entscheidungsbaum in den oberen Knoten am häufigsten auftraten. Dieser zweite Lernschritt ergab eine Menge von je zwölf Regeln zur Beschreibung schlechter oder guter Kredite. Die Regel, daß ein Kredit immer erfolgreich verlaufen ist, wenn es einen zweiten Kontonutzer gab, wurde auch hier entdeckt. Die anderen Regeln fragen meist den minimalen Kontostand und die monatlichen Netto-Einnahmen zu verschiedenen Zeitpunkten vor Kreditbeginn ab.

Pijls

Ziel. Pijls [Pij99] legte sich kein Data Mining-Ziel vorab fest, sondern begann nach der Datenvorbereitung direkt mit der Analyse.

Data Mining-Verfahren. Dabei benutzte er kein fertiges Data Mining-System, sondern verwendete bekannte Unix-Tools wie wc, grep, sed, awk etc., um Häufigkeiten und Zusammentreffen verschiedener Werte in der Datenbank zu finden.

Resultate. Pijls entdeckte einige Eigenschaften der Daten, zum Beispiel daß circa 40% aller Transaktionen Bar-Abbuchungen sind oder daß circa 30 % aller Transaktionen am 30. oder 31. Tag eines Monats stattfinden. Die Interpretation der Ergebnisse wird dadurch erschwert, daß die Ausrichtung auf ein explizites Data Mining-Ziel fehlt.

Da hier kein automatisiertes Entdeckungsverfahren zum Einsatz kam, sondern der Zugang zu den Daten durch den Nutzer direkt in eigens programmierten Unix-Kommandos erfolgte, läßt sich hier auch keine Information daraus ziehen, daß bestimmte Regeln nicht gefunden wurden.

Van der Putten

Ziele. Putten [Put99] betätigte sich auf dem Gebiet des Kreditkarten-Marketing. Erstes Anwendungsziel ist, Bankkunden, die noch keine Kreditkarte haben, zur Benutzung von Kreditkarten zu gewinnen (*cross selling*). Als zweites Ziel schlägt er vor, die Kreditkartennutzung bei solchen Kunden zu verbessern, die bereits über eine Kreditkarte verfügen (*upgrading*). Dazu sollten Kundenprofile für Kreditkarteninhaber erstellt beziehungsweise Unterschiede zwischen Gold-Karteninhabern versus Classic-Karteninhabern und Classic-Karteninhabern versus Junior-Karteninhabern festgestellt werden. Außerdem sollten für beide Anwendungsziele Vorhersagemodelle induziert werden.

Das Modell für *cross selling* sollte potentielle Kreditkartenkunden erkennen. Das Modell für *upgrading* sollte eine Vorhersage für die Intensität liefern, mit der ein Kunde seine Kreditkarte typischerweise nutzt, um diese erwartete mit der tatsächlichen Nutzungsintensität eines Kunden zu vergleichen. Es zeigte sich jedoch, daß die vorhandenen Daten nicht ausreichten, um ein solches Modell zu erzeugen.

Data Mining-Verfahren. Putten setzte die visuellen Data Mining-Umgebung DataDetective ein. Um interessante Eigenschaften der Zielgruppen zu entdecken, verwendete er die vom System bereitgestellten Standardmethoden zum Entdecken univariater Abweichungen (*univariate deviation detection*), und zur Induktion von Vorhersagemodellen den k-*nearest neighbours*–Algorithmus des Systems.

Resultate. Diese Arbeit bearbeitet die Fragestellung, wie sich der Einsatz von Kreditkarten ausdehnen oder verbessern läßt und befaßt sich nicht mit Krediten. Daher werden die Resultate hier nicht diskutiert. Das Papier legt einen Schwerpunkt auf Erklärungen, wie gefundene Ergebnisse genutzt werden können.

InfoZoom

Data Mining-Ziele. Das Ziel der Anwendung von InfoZoom [SB99] ist, typische Merkmale problematischer versus günstiger Kreditverläufe zu erkennen.

Data Mining-Verfahren. Spenke und Beilke setzen das bei der GMD entwickelte und über eine Spin-off Firma vermarktete interaktive visuelle Data Mining–System InfoZoom ein. InfoZoom stellt die analysierten Daten in Tabellenform dar. Durch interaktives Komprimieren, Sortieren oder Ausblenden von Teilen der Daten erhält der Benutzer graphische, zum Teil animierte Rückmeldung über Mengenverhältnisse in den Daten und kann dadurch interessante Zusammenhänge erkennen.

Resultate. Mit InfoZoom sind mehrere interessante Regeln zur Beschreibung günstig oder ungünstig verlaufender Kredite gefunden worden. Darunter sind beispielsweise auch die Zusammenhänge zwischen positiv verlaufenden Krediten und der Existenz eines zweiten Kontobenutzers oder der Existenz von Daueraufträgen des Typs $household$, die den Regeln 1 beziehungsweise 3 in der Tafel 7.3 entsprechen. Eine möglicherweise interessante Regel, die von anderen Systemen nicht entdeckt wurde, ist zum Beispiel, daß der Anteil problematischer Kredite wesentlich geringer ist unter solchen Konten, die mindestens eine Buchung von oder zu einer Bank mit Kürzel WX aufweisen. Ist diese Buchung eine Gutschrift, so gibt es sogar nie Probleme bei der Kreditrückzahlung. Diese beiden Zusammmenhänge illustrieren, wie bei der Anwendung von InfoZoom durch fortgesetzte Spezialisierung des Voraussetzungsteils interessanter Regeln der dargestellte Zusammenhang verschärft werden kann — ein Vorteil, den der interaktive Ansatz mit sich bringt.

7.4 Diskussion

7.4.1 Vergleich der Beiträge

Der Discovery Challenge verlangte, den gesamten Prozeß einer Data Mining-Anwendung abzudecken, von der Definition der Geschäftsziele über die Durchführung des Data Mining (mit Datenaufbereitung, Datenanalyse und Festlegung von Evaluationskriterien) bis zur Präsentation des entdeckten Wissens mit Erklärungen für die Datenbankeigner, wie die Resultate genutzt werden können [BZ99]. Bei vielen Bearbeitungen liegt der Schwerpunkt jedoch auf dem Data Mining-Schritt, in dem der eigentliche Data Mining-Algorithmus zum Regel- oder Klassifikatorlernen zur Anwendung kommt. Ein Grund hierfür ist, daß die Daten schon gut strukturiert und weitgehend aufbereitet zur Verfügung gestellt wurden.

Ausnahmen bilden die Arbeit von van der Putten [Put99], der einen Schwerpunkt auf die Definition der Data Mining-Ziele und die Nutzung der Data Mining-Ergebnisse legte, und das Vorgehen von Mikšovský et al., die bei der Datenvorbereitung und -sichtung vor dem Einsatz der eigentlichen Data Mining-Algorithmen einige interessante Aufälligkeiten der Datenbank aufdecken konnten. Bei dem visuellen interaktiven Ansatz von InfoZoom

sind die Data Mining-Phasen Datenexploration und Regelentdecken miteinander verwoben.

Der Vergleich der Ergebnisse der verschiedenen Arbeiten legt nahe, daß der Einfluß des jeweils verwendeten Suchalgorithmus und der Interessantheitskriterien eher gering ist, solange eine systematische Suche im Regelraum stattfindet. Dies zeigt sich darin, daß einige Regeln von mehreren Verfahren gefunden wurden. Vor allem die zu 100% zutreffende Regel „wenn zweiter Kontonutzer, dann problemloser Kredit“, haben mit Ausnahme von WizWhy alle Verfahren entdeckt, die diese Aufgabenstellung bearbeitet haben.

Als interaktiver Ansatz führt das System InfoZoom keine automatische Suche durch, sondern überläßt die Verantwortung für Vollständigkeit und Organisation der Suche dem Anwender. Die so gefundenen Ergebnisse sind bei dieser Anwendung jedoch gleichwertig mit den durch die automatisierten Verfahren gelieferten Resultaten. Wenn man dem die Bearbeitung von Pijls gegenüberstellt, wird deutlich, daß komfortable und mächtige Data Mining-Systeme und -Umgebungen für den Erfolg einer Data Mining-Anwendung sehr wichtig sind.

Großen Einfluß auf die Qualität der Data Mining-Ergebnisse hat die Auswahl und Definition der bei der Suche berücksichtigten Attribute. Beispielsweise sind m.E. die Regeln in Tafel 7.3 interessant, die die Stabilität der Gutschriften und Abbuchungen für ein Konto berücksichtigen. Solche Regeln lassen sich nur finden, wenn entsprechende abgeleitete Attribute definiert werden. Deshalb ist es wichtig, daß ein Data Mining-Verfahren die Definition von abgeleiteten Attributen gut unterstützt. Da die Berechnung komplexer abgeleiteter Attribute viel Rechenaufwand erfordern kann, ist ihre Verwendung in interaktiven Systemen, wo schnelle Reaktionszeiten erwünscht sind, möglicherweise weniger naheliegend. Andererseits kann ein interaktiver, explorativer Zugang zu den Daten die Entdeckung interessanter abgeleiteter Attribute fördern.

Die Bearbeitungen zeigen auch, daß sich auch die verwendeten Interessantheitskriterien nicht stark auf die Ergebnisse einer Data Mining–Anwendung auswirken. Letztendlich sind Interpretierbarkeit und Anwendbarkeit ausschlaggebend dafür, welche gefundenen Regeln der Anwender als interessant einstuft. Diese Eigenschaften von Regeln lassen sich bekanntermaßen nur schwer in formale Interessantheitskriterien fassen, so daß die Kriterien eine vernünftige Vorauswahl für die subjektive manuelle Bewertung treffen sollten. Dabei können zusätzliche Bewertungsmaßstäbe (wie der Anteil der Treffer unter den abgedeckten Fällen) hilfreich sein, ebenso die visuelle Darstellung der Regelqualität wie bei InfoZoom oder durch Balkendiagramme wie in den Abbildungen 7.1 bis 7.5.

7.4.2 Fazit

Der Discovery Challenge der PKDD’99 bot eine gute Möglichkeit, eine Data Mining-Anwendung anhand echter Daten durchzuspielen. Die in dieser Arbeit entwickelten Ausdrucksmöglichkeiten der Hypothesensprache konnten dabei sinnvoll eingesetzt werden:

- Einbeziehung mehrerer Relationen,
- Intervallhierachien auf diskretisierten numerischen Attributen und
- Spezialisierungsbeziehungen zwischen Merkmalen wie die zwischen Existenz einer Kreditkarte und deren Typ.

Der Vergleich mit anderen Beiträgen zum Discovery Challenge hat gezeigt, daß der hier entwickelte Ansatz zur Lösung einer solchen Anwendung prinzipiell geeignet und dazu in der Lage ist, zu ähnlichen Systemen gleichwertige Ergebnisse zu liefern.

Als stapelverarbeitender Ansatz bringt er spezifische Vor- und Nachteile mit sich.Um die Aufbereitungs- und Erkundungsphasen sowie die Auswertung der Ergebnisse im Data Mining-Prozeß zu unterstützen, ist die Kombination dieses autonom laufenden Suchverfahrens mit interaktiven Systemteilen sinnvoll, wie sie heute auch die großen integrierten Data Mining-Systeme bieten.

Die Visualisierung der Daten und Regeln ist sehr hilfreich für die Interpetation der Ergebnisse. Die Abbildungen 7.1 bis 7.5 zeigen, daß die graphische Darstellung der Häufigkeitsverteilung von Ziel- und Nicht–Zielelementen einer Regel als Balkendiagramm wesentlich einsichtiger ist als lediglich ihre Angabe in Form von Zahlen. Zusätzlich wären für die Auswertung der akzeptierten Regeln bequeme interaktive Inspektionsmöglichkeiten für die Datenbasis wünschenswert, um gezielt ergänzende Informationen beispielsweise über ähnliche Regeln einholen zu können. Vielsprechend erscheint die Kombination eines stapelorientierten Systems mit einem interaktivem visuellen System. Mit dem interaktiven visuellen System könnten erste Einblicke in die Daten und Anregungen für interessante Attribute gewonnen werden, während das stapelorientierte System für eine gründliche und systematische Suche auch nach Regeln mit einer größeren Anzahl von Bedingungen dienen würde. Die dabei gefundenen Regeln ließen sich dann interaktiv wieder genauer inspizieren. Daraus könnten sich dann möglicherweise neue Anregungen für weitere Suchläufe ergeben.

Kapitel 8

Schlußbemerkungen

Data Mining und KDD ist derzeit ein bedeutendes, junges und aktives Gebiet, das großes kommerzielles Potential aufweist. Eine seiner Wurzeln ist das Maschinelle Lernen, ein etabliertes Teilgebiet der Künstlichen Intelligenz, das sich mit Theorie und Methoden der automatischen Extraktion von Wissen aus Daten befaßt. Die Induktive Logikprogrammierung ist im Schnittpunkt zwischen Maschinellem Lernen und Logikprogrammierung angesiedelt. Sie befaßt sich mit Extraktion von Wissen aus Daten, die im Formalismus der Logikprogrammierung repräsentiert sind. Damit ist die ILP besonders interessant für KDD und Data Mining, denn der prädikatenlogische Formalismus der Logikprogrammierung ist eng verwandt zu den weit verbreiteten relationalen Datenbanken und bietet darüberhinaus wesentlich größere Ausdruckskraft als die im Maschinellen Lernen sonst üblichen aussagenlogischen Darstellungsformen. Der Einsatz von ILP-Methoden für KDD und Data Mining wird neuerdings auch als Relationales Data Mining bezeichnet. Das steigende Interesse am Relationalen Data Mining manifestiert sich unter anderem in der Durchführung von Sommerschulen (zum Beispiel in Verbindung mit der ECML/PKDD'2002), Workshops zum Multi-Relational Data Mining auf wichtigen KKD-Konferenzen (zum Beispiel auf der KDD-2002) und in der Veröffentlichung eines einschlägigen Überblicksbuchs [DL01].

Zu den Anwendungsgebieten von KDD und Data Mining gehören die Teilgruppenanalyse und das Finden interessanter Zusammenhänge. Diese Arbeit hat sich zum Ziel gesetzt, Methoden zur sicheren Beschränkung des Suchraums bei der Suche nach interessanten Teilgruppen im Rahmen der ILP zu erarbeiten und zu evaluieren. Die Arbeit leistet damit einen Beitrag zur technischen Seite des relationalen Data Mining.

8.1 Zusammenfassung

Die Beiträge der Arbeit sind im einzelnen:

1. eine Formalisierung der Teilgruppenanalyse im Rahmen der ILP,
2. die Herleitung von Optimumschätzfunktionen zur Verteilungsungewöhnlichkeit und zur Folgerungsstärke,
3. die Erweiterung der Apriori-Kandidatengenerierung um *req*- und *excl*-Bedingungen,
4. ein ILP-Sprachbias für die Teilgruppenanalyse, der die Anwendung der Teilmengenbedingung zur Beschränkung des Suchraums erlaubt,
5. ein SQL-Sprachbias für die Teilgruppenanalyse in multi-relationalen Datenbanken,
6. die Genex-Repräsentation, die die Suchraumbeschränkung anhand von Taxonomien und Allgemeinerbeziehungen zwischen Merkmalen in einen Apriori-artigen Suchalgorithmus integriert,

7. eine Methode zur Behandlung diskretisierter numerischer Attribute, die die Suchraumbeschränkung anhand von Allgemeinerbeziehungen zwischen Intervallen vereinheitlicht mit der Suchraumbeschränkung anhand von Taxonomien,

8. experimentelle Evaluation der Wirksamkeit der verschiedenen Möglichkeiten zur Suchraumbeschränkung,

9. die Anwendung der entwickelten Ansätze auf ein „echtes“ Data Mining-Problem und

10. zu allen genannten Punkten außer 8. und 5 ausführliche Vergleiche mit verwandten Arbeiten.

8.2 Diskussion

Die Schwerpunkte der Arbeit liegen bei 3., 4., 5. und 8. Die Einzelheiten wurden bereits im Verlauf der Arbeit ausführlich diskutiert. Hier werden die Beiträge nochmals in den Gesamtzusammenhang von KDD und ILP eingeordnet.

Formalisierung. Die gewählte Formalisierung der Teilgruppenanalyse im Rahmen des relationalen Data Mining ist bekannten Ansätzen zu verwandten Aufgabenstellungen ähnlich, um den Vergleich der entwickelten Methoden und gewonnenen Erkenntnisse mit diesen Ansätzen und möglicherweise die Integration zu ermöglichen.

Optimumschätzfunktionen. Die hergeleiteten Optimumschätzfunktionen bilden eigenständige Beiträge dieser Arbeit. Da die Optimumschätzfunktionen jedoch nur in Verbindung mit den genannten Interessantheitsfunktionen verwendet werden können, haben sie einen begrenzten Einsatzbereich.

***req*- und *excl*-Bedingungen.** Die *req*- und *excl*-Bedingungen sind eine neuartige Erweiterung des Apriori-Algorithmus, die es erlaubt, seine Hypothesensprache einzuschränken, an spezifische Anwendungen anzupassen und dabei Vorwissen und Absichten der Anwender einzubeziehen. Die Teilmengenbedingung, die der Apriori–Algorithmus zur Beschränkung des Suchraums einsetzt, leistet auch für eine durch *req*- und *excl*–Bedingungen verkleinerte Hypothesensprache eine bestmögliche und sichere Beschränkung der Suche. Verwandte Arbeiten zum deklarativen Sprachbias für den Apriori- oder einen Apriori-ähnlichen Algorithmus sind beispielsweise [SVA97, HF95b]; [KMV99] setzen Regel-Schablonen (*templates* ein, um anschließend an einen Apriori-Suchlauf die von diesem gelieferte Menge von Assoziationsregeln weiter einzugrenzen.

ILP–Sprachbias. Der in dieser Arbeit entwickelte ILP–Sprachbias und der dazugehörige Spezialisierungsoperator sind neue Beiträge zur ILP. Der Ansatz gestattet die wirkungsvolle und kostengünstige Beschränkung des Suchraums mittels der Teilmengenbedingung. Diese wurde meines Wissens in der ILP bisher nicht erfolgreich eingesetzt.

SQL–Sprachbias. Neben einem ILP–Sprachbias und ILP-Spezialisierungsoperator bietet der hier entwickelte Ansatz auch einen Sprachbias und Spezialisierungsoperator für SQL–Suchräume. Über geeignete Schnittstellen kann das Suchverfahren dann direkt auf SQL-Datenbanken operieren. Im Unterschied zu anderen ILP-Ansätzen wie Mobal/RDT DB [LM95], die logisch repräsentierte ILP-Hypothesen in SQL-Anfragen zur Auswertung dieser Hypothesen transformieren, erlaubt er die Deklaration der Hypothesensprache direkt in SQL-Ausdrücken. Damit lassen sich auch Sprachmittel von SQL einsetzen, die über die Logikprogrammierung hinausgehen. Ein solcher Sprachbias und Spezialisierungsoperator

ist meines Wissens in der ILP bisher noch nicht entwickelt, auch wenn [BDR96] die potentielle Nützlichkeit eines solchen Ansatzes diskutieren. Arbeiten aus der KDD, die sich mit dem SQL-basierten Data Mining befassen, sind beispielsweise [STA98, AS96, TS98].

Genex-Repräsentation. Die Genex-Repräsentation ist eine neue Erweiterung und Ergänzung zum Apriori-Algorithmus, die die Suchraumbeschränkung anhand von Subsumtionsbeziehungen zwischen Merkmalen in die Suchraumbeschränkung mittels der Teilmengenbedingung integriert. Sie bewirkt, daß alle bekannten Subsumtionsbeziehungen zwischen Hypothesen zur Beschränkung der Suche verwendet werden, ohne den Suchraum unnötig fein zu untergliedern (vergleiche Abschnitt 6.4). Wie weit sie auch für die ursprüngliche Anwendung des Apriori-Algorithmus, das Finden häufiger Assoziationen in Transaktionsdatenbanken, nützlich sein kann, wurde in dieser Arbeit nicht untersucht. Das Finden häufiger Assoziationen in Transaktionsdatenbanken involviert meist größere Datenbanken und vor allem umfangreichere Hypothesensprachen, als hier für die relationale Teilgruppenanalyse angenommen wurden. Die existierenden Algorithmen verwenden meist sehr spezialisierte Datenstrukturen, die möglicherweise mit der Genex-Repräsentation nicht harmonieren. Experimente zum Vergleich des entwickelten Ansatzes mit bestehenden Verfahren zum Finden generalisierter oder quantitativer Assoziationen und gegebenenfalls die Entwicklung geeigneter Datenstrukturen oder Auswertungsmethoden stehen noch aus.

Behandlung numerischer Attribute. In dieser Arbeit wird vorgeschlagen, numerische Merkmale statt als Test auf Enthaltensein in einem Intervall als Vergleich mit nur einer Intervallgrenze zu definieren. Es wurde demonstriert, daß der in der Arbeit entwickelte Spezialisierungsoperator mithilfe geeigneter *req*- und *excl*-Bedingungen die Intervallgrenzen zu Intervallen abnehmenden Umfangs kombiniert, also vom Allgemeinen zum Speziellen hin. Dadurch wird die Behandlung numerischer Attribute mit dem Abstieg in Taxonomien und in den ILP-Spezialisierungsoperator integriert.

In Apriori-basierten Algorithmen zur Suche nach häufigen Assoziationen wurde dieser Ansatz meines Wissens bisher nicht verwendet. Ob er für diesen Zweck effizient sein kann, ist noch zu untersuchen. Die einheitliche Behandlung der numerischen und der strukturierten Attribute zusammen mit den *req*- und *excl*-Bedingungen ist ein Beitrag zur Entwicklung von flexibel einsetzbaren und adaptierbaren Data Mining–Systemen, die auf dem Apriori-Algorithmus aufbauen.

Experimente. Mit einer prototypischen Implementation der in dieser Arbeit entwickelten Ansätze wurden experimentelle Untersuchungen durchgeführt und der Nutzen der verschiedenen Methoden zur Beschränkung des Suchraums evaluiert und verglichen. Die Experimente wurden mit einer ein-relationalen und einer multi-relationalen Datenbank mit nicht-deterministischen Verbindungen und unterschiedlich spezifizierten Aufgabenstellungen durchgeführt. Die Resultate der Experimente sind teilweise recht unterschiedlich, so daß davon ausgegangen werden kann, daß die verschiedenen experimentellen Anwendungen eine gewisse Bandbreite möglicher Anwendungssituationen abdecken. Experimente mit künstlichen Datenbanken, deren untersuchungsrelevante Eigenschaften anhand geeigneter Parameter systematisch variiert werden, wie sie beispielsweise Srikant und Agrawal für generalisierte Assoziationsregeln durchgeführt haben [SA95b], sind im Rahmen der ILP nicht angemessen, da die Ausdrucksmöglichkeiten der ILP-Sprachen zu viele Variationsmöglichkeiten eröffnen. Ähnliche Vergleiche mit weiteren, möglichst auch echten Anwendungen könnten weitere Einsichten liefern.

Die Ergebnisse aus den Experimenten geben einen Eindruck von der Wirksamkeit verschiedener Methoden zur sicheren Beschränkung des Suchraums bei einer vollständigen Suche in individuenzentrierten Anwendungen. Die Teilmengenbedingungen sowie Optimumschätzfunktionen haben sich als sehr wirkungsvolle und zuverlässige Methoden zur

Beschränkung des Suchraums erwiesen, während der Beitrag der Taxonomien zur Suchraumbeschränkung zwischen verschiedenen Anwendungen stark schwankte und in einigen Fällen nur gering war. Der Ausschluß von Hypothesen, die von akzeptierten Regeln subsumiert sind, ergibt deutlich kleinere Mengen akzeptierter Regeln und reduziert den Suchaufwand deutlich; der Verzicht auf ausführlichere Ergebnisse aus den Suchläufen zahlt sich also bei der Reduktion des Aufwands aus.

Ein wichtiges Ergebnis dieser Versuche ist, daß die Teilmengenbedingung zur Beschränkung des Suchraums für multi-relationale Datenbanken und ILP-Sprachen genauso wirkungsvoll sein kann wie für ein-relationale Datenbanken. Im Vergleich zu einem Suchalgorithmus, der den Suchraum mittels eines Spezialisierungsoperators durchläuft und beschränkt, werden durch die zusätzliche Berechnung der Teilmengenbedingung mehr Subsumtionsbeziehungen zwischen Hypothesen zur Beschränkung des Suchraums nutzbar. Die Experimente haben ergeben, daß der Suchraum dadurch stark eingeschränkt wird. Es ist daher beim Entwurf eines Systems zur multi-relationalen Teilgruppenanalyse empfehlenswert, den Suchraum so anzulegen, daß diese Subsumtionsbeziehungen ausgewertet werden können. Für Anwendungen, wo die Berechnung von θ–Subsumtionsbeziehungen zu aufwendig oder nicht möglich ist, weil etwa die Hypothesensprache in SQL definiert ist, beschreibt diese Arbeit eine Möglichkeit, die Hypothesensprache mithilfe der Teilmengenbedingung zu strukturieren.

Anwendung. Der entwickelte Ansatz wurde im Rahmen des Discovery Challenge der PKDD'99 auf ein echtes Data Mining-Problem, das Data Mining in Finanzdatenbanken, angewendet [Ber99b]. Dabei zeigte sich, daß der ILP-Sprachbias zur Deklaration von Data Mining-Suchräumen geeignet und angemessen ist. Alle der darin verfügbaren Sprachmittel konnten sinnvoll eingesetzt werden: das Data Mining auf mehreren Relation wie auch die Deklaration von Subsumtionsbeziehungen auf strukturierten oder numerischen Attributen. Abgeleitete Merkmale und Hintergrundwissen sind einfach einzubeziehen. Die mit dem Ansatz entdeckten Regeln sind annähernd gleichwertig zu denen, die mit ähnlichen Verfahren entdeckt wurden.

8.3 Ausblick

Diese Arbeit ist technisch orientiert. Sie realisiert und integriert verschiedene Möglichkeiten zur Beschränkung des Suchraums für die Teilgruppenanalyse im relationalen Data Mining in einem einheitlichen Suchalgorithmus und entwickelt mit dem in Kapitel 4 eingeführten ILP-Sprachbias und der Erweiterung des Apriori-Suchalgorithmus einen Ansatz, die Teilmengenbedingung auch zur Beschränkung des Suchraums bei der Suche in multi-relationalen Datenbanken einzusetzen. Die entwickelten Ansätze wurden implementiert; das resultierende System wurde für Experimente verwendet und auch eine „echte" Data Mining-Aufgabe konnte damit bearbeitet werden. Das Ziel der Arbeit war jedoch nicht, ein vollwertiges, eigenständiges Data Mining-System zu entwickeln, sondern, auch durch die experimentellen Untersuchungen, zur Entwicklung und Verbesserung von Komponenten von Data Mining–Systemen beizutragen und anzuregen. Das angestrebte Ziel ist erreicht, wenn die entwickelten Verfahren oder Ideen in Data Mining–Algorithmen und –Systeme Eingang finden. Es bieten sich einige Ansatzpunkte, die Arbeit weiterzuführen.

Definition von Suchräumen. Die Deklaration der Hypothesensprache im hier entwickelten Formalismus kann sehr umfangreich und aufwendig werden und ist entsprechend fehlerträchtig. Diesem Problem kann man abhelfen durch Deklarationengeneratoren, die die vom System benötigten Sprachdeklarationen ganz oder teilweise automatisch generieren. Eine Möglichkeit dazu ist, einen geeigneten Sprachbias höherer Abstraktion, etwa nach Art des ISP-Bias (siehe Abschnitt 4.4), bereitzustellen, mit dem einfach und kompakt Hypothesensprachen spezifiziert werden können, und diesen Sprachbias vor den eigentlichen

Suchalgorithmus derart vorzuschalten, daß aus diesen abstrakteren und kompakteren Deklarationen automatisch die umfangreicheren Deklarationen erzeugt werden, die der hier entwickelte Suchalgorithmus benötigt. Ein solcher vorgeordneter Sprachbias würde jedoch Einbußen an Flexibilität mit sich bringen.

Eine m. E. interessantere Fortführung der Arbeit ist die Entwicklung eines von der logischen Repräsentation losgelösten Sprachbias, der dem Anwender verschiedene geeignete Teil-Suchräume für vom Anwender ausgewählte Attribute der Datenbank vorgibt, wie die in Abschnitt 4.3 beschriebenen „einfachen Kombinationen“, „Kombinationen mit n-fachen Vorkommen“, „Sequenzen bis Länge m“ oder die in Abschnitt 5.2 beschriebenen „kombinierenden Taxonomien“, „ausschließenden Taxonomien“, Intervalleinteilungen für numerische Attribute etc. Ein solcher Sprachbias ließe sich möglicherweise auch in die graphische Benutzeroberfläche eines umfassenden Data Mining-Systems integrieren. Dabei ist auch ein integriertes System mit komfortabler graphischer Benutzerschnittstelle vorstellbar, das — nach Spezifikation einer Individuenpopulation und Fremdschlüsselbeziehungen durch den Anwender — Attribute und Relationen, in denen solche Hypothesen eventuell zu entdecken sind, weitgehend selbständig bestimmt und dem Anwender zur Auswahl präsentiert. Eine solche Deklaration der Hypothesensprache wäre anwendungsfreundlicher als ein Sprachbias, der die Deklaration von Literalen verlangt, da sie die Deklaration der Hypothesensprache mehr aus Anwendersicht und weniger aus der technischen Sicht des Suchalgorithmus betrachtet, und würde, zusammen mit dem ISP- und dem Fremdverbindungsbias (siehe Abschnitt 4.4), einen weiteren, auf individuenzentrierte Lernprobleme abgestimmten ILP-Sprachbias bilden.

Verhältnis zwischen aussagenlogischen und relationalen Lernproblemen. Die Forschung befaßt sich schon länger damit, das Verhältnis zwischen aussagenlogischen und relationalen Lernproblemen besser zu verstehen. Das *multiple instance*–Problem und die sogenannten individuenzentrierten Lernprobleme sind als den aussagenlogischen Lernproblemen verwandte Versionen relationaler und prädikatenlogischer Lernprobleme erkannt und Datenrepräsentationen jeweils entsprechender Komplexität entwickelt worden. In Beziehung dazu stehen auch Arbeiten wie [JN02a, JN02b], die sich mit Besonderheiten der Merkmalsauswahl beim Klassifikatorenlernen aus propositionalisierten relationalen Daten befassen.

Diese Entwicklung scheint jedoch noch nicht abgeschlossen. Möglicherweise wird man beginnen, die individuenzentrierten Lernprobleme nach dem Grad ihrer „Relationalität“ noch feiner einzuteilen[1]. Der Grad der „Relationalität“ eines Lernproblems wird dabei wohl vom Ausmaß der Nichtdeterminiertheit von $1{:}n$–Verbindungen abhängen, also davon, ob n für Werte zwischen beispielsweise 1 und 5 oder aber zwischen 1 und 10000 steht. Auch die Struktur des Verbindungsgeflechts zwischen den Relationen ist dabei wohl von Bedeutung: Wenn die Relationen und die Verbindungen zwischen ihnen graphisch repräsentiert werden als Knoten beziehungsweise Kanten, können die Verbindungen, im einfachen Fall einen Baum bilden mit einer die Individuenpopulation repräsentierenden Relation als Wurzel, während möglicherweise mehrfache Verbindungen zwischen zwei Relationen einen erhöhten Grad an „Relationalität“ bedeuten. Für die vorliegende Arbeit läßt sich daraus möglicherweise eine Klasse individuenzentrierter relationaler Lernprobleme definieren, für deren Hypothesensprache die Teilmengenbeziehung anstelle der θ-Subsumtion als Ordnungsrelation ausreicht.

Datenbankformate und –anfragesprachen. Der hier entwickelte Ansatz zur Suche interessanter Teilgruppen in relationalen Datenbanken strukturiert den Suchraum mit der Teilmengenbeziehung statt mit der sonst für die Ordnung relationaler Suchräume verwendeten θ-Subsumtion. Über die Teilmengenbeziehung hinausgehende Struktur des Hypothesenraums läßt sich mithilfe der *req*- und *excl*-Bedingungen modellieren; Der Ansatz

[1] C. Rouveirol, Podiumsdiskussion auf der ILP'2002; P. Flach, persönliche Kommunikation.

verlagert also gewissermaßen den Aufwand von den θ-Subsumtionstests in die Deklaration der Hypothesensprache. Der offensichtliche Nachteil dieses Vorgehens sind die komplexen und aufwendigen Deklarationen; ein bedeutender Vorteil ist, daß der Ansatz dadurch von der Syntax der Hypothesen unabhängiger wird und in uniformer Weise für in Prolog- und für in SQL formulierte Hypothesen- und Anfragesprachen tauglich ist. Möglicherweise läßt er sich auch für weitere Datenbankformate und -anfragesprachen von vergleichbarer Ausdrucksstärke verwenden.

Anhang A

SQL-Eingabedateien

A.1 SQL–Deklarationen für KRK ohne Taxonomie

Gezeigt sind die Deklarationen für den (einzigen) Verbinder, Individuenpopulation und Zielklasse sowie für die Merkmale, die das Figurenpaar WK/WR betreffen. Die gesamte Eingabedatei enthält zusätzlich noch die entsprechenden Deklarationen für die Figurenpaare WK/BK und WR/BK. Die Deklarationen unterschieden sich nur in den Definitionsteilen von den entsprechenden Prolog–Deklarationen.

$count(pos,$ $[att(''krk.id''),$
$from([krk])]).$
$class(class,$ $[att(''krk.class''),$
$targetclass('''1'''),$
$from([krk])]).$

$link(krk,$ $[rel(''krk'')]).$

$lit(wkc\_eq\_wrc,$ $[def([''krk.wkingcol = krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_neq\_wrc, wkc\_l\_wrc, wkc\_g\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_neq\_wrc,$ $[def([''krk.wkingcol <> krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_eq\_wrc, wkc\_l\_wrc, wkc\_g\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_l\_wrc,$ $[def([''krk.wkingcol < krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_eq\_wrc, wkc\_neq\_wrc, wkc\_g\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_g\_wrc,$ $[def([''krk.wkingcol > krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_eq\_wrc, wkc\_neq\_wrc, wkc\_l\_wrc, wkc\_ad\_wrc])]).$
$lit(wkc\_ad\_wrc,$ $[def([''krk.wkingcol - krk.wrookcol\ in\ (-1, 0, 1)'']),$
$from([krk]),$
$excludes([wkc\_eq\_wrc, wkc\_neq\_wrc, wkc\_l\_wrc, wkc\_g\_wrc])]).$
$lit(wkr\_eq\_wrr,$ $[def([''krk.wkingrow = krk.wrookrow'']),$
$from([krk]),$
$excludes([wkr\_neq\_wrr, wkr\_l\_wrr, wkr\_g\_wrr, wkr\_ad\_wrr])]).$

$$
\begin{array}{ll}
lit(wkr\_neq\_wrr, & [def([''krk.wkingrow <> krk.wrookrow'']),\\
 & from([krk]),\\
 & excludes([wkr\_eq\_wrr, wkr\_l\_wrr, wkr\_g\_wrr, wkr\_ad\_wrr])]).\\
lit(wkr\_l\_wrr, & [def([''krk.wkingrow < krk.wrookrow'']),\\
 & from([krk]),\\
 & excludes([wkr\_eq\_wrr, wkr\_neq\_wrr, wkr\_g\_wrr, wkr\_ad\_wrr])]).\\
lit(wkr\_g\_wrr, & [def([''krk.wkingrow > krk.wrookrow'']),\\
 & from([krk]),\\
 & excludes([wkr\_eq\_wrr, wkr\_neq\_wrr, wkr\_l\_wrr, wkr\_ad\_wrr])]).\\
lit(wkr\_ad\_wrr, & [def([''krk.wkingrow - krk.wrookrow\ in\ (-1, 0, 1)'']),\\
 & from([krk]),\\
 & excludes([wkr\_eq\_wrr, wkr\_neq\_wrr, wkr\_l\_wrr, wkr\_g\_wrr])]).
\end{array}
$$

A.2 SQL–Deklarationen für KRK mit Taxonomie

Die Deklarationen stimmen bis auf die *req*- und *excl*-Bedingungen mit den Deklarationen für den Suchraum ohne Taxonomie überein.

$count(pos,$ $[att(''krk.id''),$
$from([krk])]).$

$class(class,$ $[att(''krk.class''),$
$targetclass('''1'''),$
$from([krk])]).$

$link(krk,$ $[rel(''krk'')]).$

$lit(wkc\_ad\_wrc,$ $[def(['' krk.wkingcol - krk.wrookcol\ in\ (-1,0,1)'']),$
$from([krk]),$
$excludes([wkc\_neq\_wrc])]).$

$lit(wkc\_neq\_wrc,$ $[def(['' krk.wkingcol <> krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_ad\_wrc])]).$

$lit(wkc\_eq\_wrc,$ $[def(['' krk.wkingcol = krk.wrookcol'']),$
$from([krk]),$
$requires([wkc\_ad\_wrc])]).$

$lit(wkc\_l\_wrc,$ $[def(['' krk.wkingcol < krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_g\_wrc]),$
$requires([wkc\_neq\_wrc])]).$

$lit(wkc\_g\_wrc,$ $[def(['' krk.wkingcol > krk.wrookcol'']),$
$from([krk]),$
$excludes([wkc\_l\_wrc]),$
$requires([wkc\_neq\_wrc])]).$

$lit(wkr\_ad\_wrr,$ $[def(['' krk.wkingrow - krk.wrookrow\ in\ (-1,0,1)'']),$
$from([krk]),$
$excludes([wkr\_neq\_wrr])]).$

$lit(wkr\_neq\_wrr,$ $[def(['' krk.wkingrow <> krk.wrookrow'']),$
$from([krk]),$
$excludes([wkr\_ad\_wrr])]).$

$lit(wkr\_eq\_wrr,$ $[def(['' krk.wkingrow = krk.wrookrow'']),$
$from([krk]),$
$requires([wkr\_ad\_wrr])]).$

$lit(wkr\_l\_wrr,$ $[def([''krk.wkingrow < krk.wrookrow'']),$
$from([krk]),$
$excludes([wkr\_g\_wrr]),$
$requires([wkr\_neq\_wrr])]).$
$lit(wkr\_g\_wrr,$ $[def([''krk.wkingrow > krk.wrookrow'']),$
$from([krk]),$
$excludes([wkr\_l\_wrr]),$
$requires([wkr\_neq\_wrr])]).$

A.3 SQL–Deklarationen für Phonetik ohne Taxonomie

$count(sylids,$ $[att(''syl.syllable''),$
$from([syl])]).$
$class(accent,$ $[targetclass('acc'),$
$from([syl]),$
$att(''$CASE WHEN $syl.accent <>' 0'$ THEN $'acc'$
ELSE $'no\_acc'$ END$'')]).$

$link(syl,$ $[rel(''syllables'')]).$
$link(w,$ $[rel(''words)''),$
$def([''syl.word = w.word'']),$
$from([syl])]).$
$link(avg,$ $[rel(''avg\_syl)'')]).$
$link(t1,$ $[rel(''phonemes''),$
$def([''t1.syllable = syl.syllable'']),$
$from([syl])]).$
$link(t2,$ $[rel(''phonemes''),$
$def([''t2.syllable = syl.syllable'']),$
$from([syl])]).$

$\vdots$

$link(t8,$ $[rel(''phonemes''),$
$def([''t8.syllable = syl.syllable'']),$
$from([syl])]).$
$link(p1,$ $[rel(''phonemes''),$
$def([''p1.syllable = syl.syllable'']),$
$from([syl])]).$
$link(p2,$ $[rel(''phonemes''),$
$def([''p2.syllable = syl.syllable'']),$
$from([syl])]).$

$\vdots$

$link(p39,$ $[rel(''phonemes''),$
$def([''p39.syllable = syl.syllable'']),$
$from([syl])]).$

$lit(long\_syl,$ $[def([''syl.len\_expected \geq syl.len\_measure'']),$
$from([syl]),$
$excludes([short\_syl])]).$

$lit(short\_syl,$ $[def([''syl.len\_expected < syl.len\_measure'']),$
$from([syl]),$
$excludes([long\_syl])]).$

$lit(high\_sylpeak,$ $[def([''syl.peak\_height \geq avg\_syl.avg\_peak\_height'']),$
$from([syl, avg]),$
$excludes([low\_sylpeak])]).$

$lit(low\_sylpeak,$ $[def([''syl.peak\_height < avg\_syl.avg\_peak\_height'']),$
$from([syl, avg]),$
$excludes([high\_sylpeak])]).$

$lit(high\_tonalalign,$ $[def([''syl.tonal\_align \geq avg\_syl.avg\_tonal\_align'']),$
$from([syl, avg]),$
$excludes([low\_tonalalign])]).$

$lit(low\_tonalalign,$ $[def([''syl.tonal\_align < avg\_syl.avg\_tonal\_align'']),$
$from([syl, avg]),$
$excludes([high\_tonalalign])]).$

$lit(short\_dist2nextp,$ $[def([''syl.dist2nextp < avg\_syl.avg\_dist2nextp'']),$
$from([syl, avg]),$
$excludes([long\_dist2nextp])]).$

$lit(long\_dist2nextp,$ $[def([''syl.dist2nextp \geq avg\_syl.avg\_dist2nextp'']),$
$from([syl, avg]),$
$excludes([short\_dist2nextp])]).$

$lit(diphtong,$ $[def([''t1.type =' diphtong''']),$
$from([t1]),$
$excludes([$übrige Vokale$])]).$

$lit(frikativ,$ $[def([''t2.type =' frikativ''']),$
$from([t2]),$
$excludes([$übrige Vokale$])]).$

$\vdots$

$lit(dv\_aI,$ $[def([''p1.phoneme =' dv\_aI''']),$
$from([p1]),$
$excludes([$übrige Vokale$])]).$

$lit(dv\_aU,$ $[def([''p2.phoneme =' dv\_aU''']),$
$from([p2]),$
$excludes([$übrige Vokale$])]).$

$lit(dv\_OY,$ $[def([''p3.phoneme =' dv\_OY''']),$
$from([p3]),$
$excludes([$übrige Vokale$])]).$

$\vdots$

$lit(pc\_k,$ $[def([''p37.phoneme =' pc\_k''']),$
$from([p37])]).$
$lit(pc\_p,$ $[def([''p38.phoneme =' pc\_p''']),$
$from([p38])]).$
$lit(pc\_t,$ $[def([''p39.phoneme =' pc\_t''']),$
$from([p39])]).$
$lit(adj,$ $[def([''w.tagIN('adja',' adjd')'']),$
$from([w]),$
$excludes([\text{übrige Tags}])]).$
$lit(adja,$ $[def([''w.tag =' adja''']),$
$from([w]),$
$excludes([\text{übrige Tags}])]).$
$lit(adjd,$ $[def([''w.tag =' adjd''']),$
$from([w]),$
$excludes([\text{übrige Tags}])]).$

$\vdots$

Anhang B

Extended Abstract

B.1 Introduction

Knowledge discovery and data mining are concerned with the discovery of valid, novel, potentially useful, and understandable patterns in data. Most data mining algorithms require that the data are represented as a single attribute-value table. In contrast, relational data mining techniques are applicable directly to multi-relational databases. These techniques are developed mainly in the area of Inductive Logic Programming (ILP). ILP is a subfield of machine learning that uses subsets of first order logic as languages for the representation of data and learning results. The formalism of relational data bases also is a subset of first order logic.

Finding all interesting patterns in databases is an important data mining task. It is defined as follows: Given a database D, a pattern language L and an acceptance criterion $q : (L, D) \rightarrow \{\textit{true, false}\}$, find all patterns $p \in L$ where $q(p, D)$ is true. As the task definition demands that all patterns are discovered that are interesting in terms of the acceptance criterion q, discovery algorithms have to conduct a complete search through the pattern space and heuristic pruning is not admissible. Due to the expressivity of first-order languages, ILP search spaces typically are very large. Therefore, safe pruning techniques that prune the search space without loss of acceptable patterns are crucial for ILP pattern discovery systems.

This thesis is concerned with pruning techniques that are applicable for pattern discovery within a restricted ILP setting where all patterns describe subgroups of a fixed population of individuals. This variant of pattern discovery is also known as subgroup discovery. Several systems have been developed for this task in the framework of ILP, e.g., Warmr, Midos, and Tertius [Deh98, Wro97, FL01].

Section B.2 defines the formal framework of pattern discovery assumed here. Section B.2.2 introduces the basic search algorithm, and section B.3 elaborates on additional pruning techniques that are applicable for the task. Section B.4 reports on experiments investigating the effectiveness of pruning methods; section B.5 summarizes and concludes. For further information see [Web02, Web00, Web99b, Web99a, Web98b, Web98a, Web97].

B.2 Subgroup discovery in the framework of Inductive Logic Programming

B.2.1 Formal framework

The task of finding interesting patterns addressed here assumes that the patterns describe subgroups of a fixed population of individuals. It is similar to the settings of [Deh98, FL01,

Wro97]. The population of individuals is defined as the set of distinct bindings of specified variables in the database at hand.

Definition 39 *The population P is the set of all ground substitutions θ of specified variables $\{V_1, \ldots, V_n\}$ occurring in the population literal pop for which popθ is true in the database D, i. e., $P = \{(V_1 \ldots, V_n)\theta \mid D \models pop\theta$ and $(V_1 \ldots, V_n)\theta$ is ground$\}$.*

A pattern p is a logical expression that has to share variables with the population literal pop. The form and construction of p are explained below.

Definition 40 *The coverage of a pattern p is the set of individuals $(V_1, \ldots, V_n)\theta$ for which $D \models (pop \wedge p)\theta$. The coverage of p is denoted $cov(p)$.*

The interestingness of patterns is evaluated with respect to a target group. The target group is a subset of the population that is in the focus of interest in the given application. T is defined by a logical expression t.

Definition 41 *The target group T is the subset of the population for which the logical expression t is true, i. e., $T = \{(V_1 \ldots, V_n)\theta | D \models (pop \wedge t)\theta$ and $(V_1 \ldots, V_n)\theta$ is ground$\}$.*

The rating of the interestingness of a pattern is computed from the size of the coverage of a pattern, $|cov(p)|$, and the number of target objects the pattern covers, $|cov(p \wedge t)|$. The set of target objects covered by a pattern p is termed the supporting set of p. The set of non-target objects covered by a pattern p is termed the contradicting set of p. The sizes of these sets are computed by appropriate database queries. A data mining system using Prolog for data storage and access can compute the sizes of the coverage $C = cov(p)$ and supporting set $S = cov(p \wedge t)$ of a pattern p by the following queries:

$$findall((V_1, \ldots, V_n), (pop, p), C), length(C, SC).$$
$$findall((V_1, \ldots, V_n), (pop, p, t), S), length(S, SS).$$

B.2.2 Basic search algorithm

Specialization operator. The task definition of interesting pattern discovery demands that all patterns that meet the acceptance criterion are discovered. Therefore, a complete search through the pattern language is required. In ILP, the search for patterns is commonly organized as the systematic, repeated application of a specialization operator. Specializing a pattern p produces patterns p' that are subsumed by p. For patterns describing subgroups, a pattern p is said to subsume a pattern p' (written $p \succ p'$), if $cov(p) \supseteq cov(p')$. The specialization operator traverses the search space top-down. It starts with the most general pattern in the pattern language and specializes it and the resulting patterns until the search space is traversed completely. Generally, there are several ways to generate a pattern. In order to avoid redundancy, search algorithms take care to restrict the application of specialization operators such that, for any pattern p, exactly one sequence of specialization steps producing p is allowed.

Basic pruning. Acceptance criteria typically are derived from numerical interestingness functions for patterns. Patterns are defined as acceptable if their interestingness reaches or exceeds a certain threshold. For interestingness functions that are monotonous to the size of the coverage of a pattern, specializing a non-acceptable pattern will always produce non-acceptable patterns because specialization of a pattern produces patterns with at most equal coverage size. Consequently, patterns with too small coverage need not be specialized, but can be pruned from the search space. Thus, the specialization operator implements a basic form of safe pruning. It is realized in most ILP pattern discovery systems.

Algorithm overview. Table 2.16 shows a basic search algorithm for subgroup discovery based on a specialization operator. Its inputs are a database D, the declaration of a population of individuals pop, and a specialization operator ρ. The output is a set of accepted patterns H. The algorithm alternates between two phases cand_gen and cand_test. The input to the cand_gen phase is a set S of patterns that have already been evaluated and found worth further specialization. The set S is initialized with the pattern pop that describes the complete population. Application of the specialization operator ρ generates from S the set S' of so-called candidate patterns that will be evaluated by querying the database in the subsequent cand_test phase. The inputs to the cand_test phase are the set S' of candidates, the set H of patterns that are accepted so far, and the database D. Evaluation $eval(h, D)$ of a candidate pattern h yields one of three results: if the result equals $accept$, the pattern h is added to the set H of accepted patterns; if the result is $prune$ then h is not to be specialized further, and if the result is $keep$ than h is added to the set S so that it will be specialized in the next cand_gen phase. (Line 18 of the algorithm is needed for restriction to most general patterns as explained below in subsection B.3.4.)

B.3 Additional pruning techniques

The basic search algorithm can be improved by additional pruning techniques. The pruning techniques addressed in this thesis are (1) optimum estimates, (2) a pruning technique based on subset tests that is derived from the Apriori search algorithm, (3) pruning based on taxonomies, and (4) to consider only most general patterns as interesting. Methods (1) to (3) are safe pruning techniques, that is, they detect all acceptable patterns that are detected by a complete search not pruning the search space, whereas method (4) reduces the number of accepted patterns.

B.3.1 Optimum estimate functions

Many practically relevant interestingness functions are not monotonous to coverage size, for example, interestingness functions based on statistical tests. For some interestingness functions, optimum estimate functions can be derived. Given a pattern p, an optimum estimate function for an interestingness function f computes an upper bound for the interestingness values $f(p')$ that patterns p' that are subsumed by p can reach. If the upper bound for a pattern p is smaller than the threshold required by the acceptance criterion, p needs not to be specialized. Pruning based on optimum estimates is safe for correct optimum estimates that compute true upper bounds. The thesis contributes optimum estimates for the interestingness functions distributional unusualness [Wro97] and implication intensity [GL93].

Distributional unusualness is a well-known interestingness function for subgroup discovery. It was adapted for relational data mining and combined with a frequency threshold σ_m by Wrobel [Wro97].

Definition 42 *The distributional unusualness du of a pattern p is defined as*

$$du(p) = \begin{cases} -1 & \text{if } \frac{|cov(p)|}{|P|} < \sigma_m \\ \frac{|cov(p)|}{|P|} \cdot \left(\frac{|cov(p \wedge t)|}{|cov(p)|} - \frac{|cov(t)|}{|P|} \right) & else \end{cases}$$

The first branch of the definition of du allows to prune patterns with too small coverage. There are three optimum estimate functions for the second branch of the definition of

distributional unusualness

$$
\begin{aligned}
du_{oe1}(p) &= \frac{|cov(p \wedge t)|}{|P|} \cdot \left(1 - \frac{|cov(t)|}{|P|}\right) \\
du_{oe2}(p) &= \frac{|cov(p \wedge t)|}{|P|} - \sigma_m \cdot \frac{|T|}{|P|} \\
du_{oe3}(p) &= \frac{|cov(p)|}{|P|} \cdot \left(1 - \frac{|cov(t)|}{|P|}\right)
\end{aligned}
$$

Optimum estimate functions du_{oe1} and du_{oe2} were developed in the thesis. Optimum estimate du_{oe1} expresses the observation that a most favorable specialization operation of a pattern p excludes all and only the non-target individuals from $cov(p)$. Optimum estimate du_{oe2} is based on the idea that a favorable specialization excludes as much non-target objects as can be excluded without violating the minimum frequency constraint $cov(p) \geq \sigma_m \cdot |P|$. Optimum estimate du_{oe3} is a weaker version of du_{oe1} that is used in experiments on the effect of optimum estimates.

Implication intensity was developed by [GL93] and investigated as a measure of interest for KDD by [FDPB95]. It is a statistical measure for the strength of the implication expressed by a rule $A \rightarrow B$. Here, it is used to rate how strong the fact that an individual i is covered by a pattern p implies that i belongs to the target group T, i.e., the intensity of the implication $i \in cov(p) \rightarrow i \in T$.

Definition 43 *Let $\overline{T} = P \setminus cov(t)$, $\lambda = |cov(p)| \cdot \frac{|\overline{T}|}{|P|}$, and Φ the cumulated standard normal distribution. The implication intensity II of a pattern p wrt. a target group $cov(t)$ is*

$$
II(p) = \begin{cases} 1 - \sum_{i=0}^{|cov(p) \cap \overline{T}|} \frac{\lambda^i}{i!} \cdot \mathbf{e}^{-\lambda} & if \lambda \leq 3 \\ 1 - \Phi\left(\frac{|cov(p) \cap \overline{T}| - \lambda}{\sqrt{\lambda}}\right) & else \end{cases}
$$

The first branch of the definition computes the implication intensity based on the Poisson distribution, the second branch approximates the Poisson distribution by the normal distribution which is sufficiently exact for $\lambda > 3$ [FDPB95].

Similar to distributional unusualness, implication intensity can be combined with a minimum frequency threshold σ_i. Then, a pattern p is accepted as interesting if $II(p)$ reaches a threshold $1 - \alpha$ and if its coverage $cov(p)$ meets a minimum frequency threshold σ_i, i.e., the following must hold for any acceptable pattern p.

$$
\begin{aligned}
II(p) &\geq 1 - \alpha \\
\frac{|cov(p)|}{|P|} &\geq \sigma_i
\end{aligned}
$$

If the threshold σ_i is chosen such that $\sigma_i > \frac{3}{|\overline{T}|}$, the normal distribution branch of II applies for all acceptable patterns. Two optimum estimates for the normal distribution branch of implication intensity [Web00] were derived in the thesis.

$$
II_{oe1}(p) = 1 - \Phi\left(-\sqrt{|cov(p)| - |cov(p) \cap \overline{T}|} \cdot \frac{\sqrt{|\overline{T}|}}{\sqrt{|P|}}\right)
$$

II_{oe1} uses the same principle as the optimum estimate du_{oe1} for distributional unusualness, namely, that a most favorable specialization of pattern l would exclude exactly the contradicting instances $cov(p) \cap \overline{T}$ from its coverage $cov(p)$.

II_{oe2} is an improvement of II_{oe1}. It uses the fact that a most favorable specialization operation must keep enough contradicting individuals to meet the minimum frequency

constraint. For mincov $= \lceil \sigma_i \cdot |P| \rceil$ the minimum size of the coverage of a pattern that is required by the minimum frequency constraint, and $\lambda = mincov \cdot \frac{|\overline{T}|}{|P|}$,

$$II_{oe2}(p) \quad = \quad 1 - \Phi\left(\frac{(mincov - cov(p)) - \lambda}{\sqrt{\lambda}}\right)$$

B.3.2 Apriori-like pruning

Searching and pruning with a specialization operator as sketched so far has a weakness. If the search algorithm detects a pattern p with insufficient coverage, it stops further specialization of the pattern p. Thus, it prunes only such patterns from the search space, that are generated by a sequence of specialization operations including the pattern p. Other patterns that are subsumed by p, but generated by specialization sequences not including p remain in the search space.

Apriori search algorithm. The well-known Apriori algorithm [AMS+96] that was developed for finding association rules in transaction databases implements a pruning strategy that exploits all subsumption relationships between patterns for pruning. The first step of finding association rules is the search for frequent associations of items in the transactions. This task is an instance of finding all interesting patterns in a database where the patterns are associations of items, and the acceptance criterion is frequency. A pattern is frequent if the size of its coverage reaches a certain threshold. The coverage of a pattern is the number of transactions in the given database which contain the pattern as a subset.

The pattern language of this application is very simple. A pattern is a subset of a fixed set of items. All subsets of the item sets belong to the pattern language. The core idea of the Apriori search algorithm is based on the observation that, if a given pattern p is frequent, all patterns that are subsets of p are frequent as well. The Apriori search algorithm generates and evaluates patterns from general to specific, and stores the frequent patterns, i.e., the ones that are worth further specialization, in a set F. For any newly generated pattern p, the algorithm checks the *subset condition*. The subset condition is true for a pattern p if all subsets of p that are one item shorter than p, occur in S. If the subset condition is not true for a pattern, the pattern is subsumed by an infrequent pattern, and therefore, it is infrequent as well.

Table 4.2 shows the basic search algorithm augmented by a check of the subset condition. The only difference to the basic algorithm shown in table 2.16 is the subset check in line 10'.

Apriori-like ILP search algorithm. Most ILP approaches order the hypothesis language by θ-subsumption. However, the cost of detecting relationships of θ-subsumption between hypotheses is so high that it often prohibits extensive checks for pruning in pattern discovery. The thesis proposes an approach that allows to apply the subset condition for ILP languages, thus making it possible to prune the search space without expensive θ-subsumption tests. The approach includes a novel ILP language bias and an extension of the Apriori search algorithm.

The novel ILP language bias makes a distinction between so-called property literals and linkage literals. A hypothesis is a conjunct of property literals and linkage literals. Basically, a linkage literal is a variabilization of a database predicate. A property literal defines a property of individuals using variables introduced by linkage literals. A pattern language is specified by declaring all property and linkage literals that can occur in patterns. The property literals play the role of the items in the original Apriori search algorithm, so that, in the view of the algorithm, a pattern is a set of property literals. The specialization operator for this hypothesis language specializes a pattern by adding another property literal. It is completed to an ILP hypothesis by adding the linkage literals that introduce the variables used in the property literals.

Essentially, the ILP specialization operator augments the original Apriori specialization operator with *excludes* and *requires* constraints that allow to restrict the pattern language to meaningful combinations of property literals. For each property literal, an *excludes* list and a *requires* list may be declared. The *excludes* list of a property literal l is a list of property literals that must not co-occur with l in a hypothesis. The *requires* list is a list of property literals that must occur in any hypothesis containing l. A pattern is called legal if it obeys all *requires* and *excludes* constraints. In the search algorithm, the subset condition has to be modified such that it only checks occurrence of legal patterns in the set S of patterns stored for further specialization. The approach is described in more detail in [Web00].

The approach has been implemented and used for experiments and a real-world dataset. In the current state of the system, linkage and property literals as well as the *requires* and *excludes* constraints have to be declared manually.

SQL pattern language. The approach is also suitable for searching a space of patterns expressed in SQL. The algorithm generates SQL queries if the declarations provide AS–parts of the SQL queries instead of linkage literals and the WHERE -parts of SQL queries instead of property literals. Appendix A shows some example declarations; section 4.2.6 shows an example SQL query.

B.3.3 Subsumption relationships between property literals

A property literal l subsumes a property literal l' (written $l \succ l'$) iff $cov(pop \wedge l) \supseteq cov(pop \wedge l')$. Subsumption relationships between property literals stem from, e.g., taxonomies on attribute domains. For example, in a database modeling a blocks world that includes a relation $objects(OId, Shape, Size)$, the property literal $Shape = square$ is more special than the property literal $Shape = rectangle$. Subsumption relationships between property literals induce subsumption relationships between patterns: if a pattern p contains a property literal l with $l \succ l'$, and l is replaced by l' in p, the resulting pattern $p' = (p \setminus \{l\}) \cup \{l'\}$ is subsumed by p, i.e., $p \succ p'$. Thus, subsumption relationships between property literals offer an opportunity to prune the search space.

***Genex* representation.** The thesis develops an approach by which pruning based on subsumption relationships between property literals can be integrated with pruning based on the subset condition. It requires to represent patterns in *genex* form [Web00]. A pattern p is in *genex* form iff for all $l \in p$ the following holds: if there is a property literal l' declared for the pattern language with $l' \succ l$, then $l' \in p$. The *genex* representation of a pattern is unique, i.e., it does not allow syntactical variants.

The specialization operator is forced to generate patterns in *genex* form if for every subsumption relationship $l \succ l'$ a *requires* constraint is declared such that the subsumed literal l' requires the more general literal l, i. e., $l \in req(l')$. Given the appropriate *requires* constraints, pruning based on subsumption relationships between property literals is realized along with subset pruning and does not require any extra procedures.

Numerical attributes. The approach also integrates treatment of numerical attributes in a uniform manner. In pattern discovery, numerical attributes often are discretized and their range is segmented into intervals which can be arranged into a hierarchy. Figure 5.3 depicts an example hierarchy. Given appropriate declarations of property literals and *requires* and *excludes* constraints, the specialization operator generates these intervals from general to specific as illustrated in figure 5.4. To this aim, property literals have to be declared that compare the attribute value of objects with the interval boundary rather than directly test if the attribute value of objects belongs to an interval [Web00].

B.3.4 Restriction to most general patterns

The pruning methods described so far realize safe pruning, i.e., they do not change the set of accepted patterns. A further means to restrict the search space is to define that only most general patterns be interesting. Then, whenever a pattern is accepted, all patterns it subsumes can be pruned from the search space. This is realized by treating accepted patterns similar to patterns that can be pruned due to insufficient coverage or optimum estimate. Obviously, the restriction to most general patterns cannot be considered a safe pruning technique but rather implements a different acceptance criterion by which fewer patterns are accepted. In the context of the thesis, it is interesting to investigate the reduction of the size of the search space that is achieved by the restriction to most general patterns.

B.4 Experiments

The effectiveness of the additional pruning methods was investigated in a series of experimental search runs. For the experiments, three specific task settings were defined; two databases were searched. The experiments were conducted with the prototype implementation of the algorithm. The system is realized in C and Prolog and runs on Solaris work stations.

B.4.1 Task settings

Three data mining task settings with different interestingness functions and acceptance criteria are specified for the experiments.

Assoc. The first task setting, referred to as *assoc*, uses two interestingness functions, *frequency* and *confidence*. They are defined as follows.

Definition 44 *The frequency supp of pattern p with respect to a target group t in a database D is supp*$(p) = \frac{|cov(p \wedge t)|}{|cov(t)|}$. *The confidence conf of pattern p wrt. to a target group t in a database D is conf*$(p) = \frac{|cov(p \wedge t)|}{|cov(p)|}$.

The assoc task setting demands to find all patterns p, the support and confidence of which reach certain thresholds, i.e., $accept(p, D)$ is true iff supp$(p) \geq \sigma_a$ and conf$(p) \geq \sigma_c$.

Obviously, the support of a pattern is monotonous to the coverage, and allows to prune the search space accordingly. The confidence is not monotonous to the coverage. A non-trivial optimum estimate for confidence does not exist, since there is always the chance of a specialization step that excludes exactly the contradicting individuals from the coverage of the pattern, yielding a pattern with maximum confidence 1.

Midos. The second task setting is called midos. Its interestingness function is distributional unusualness. The task is to find the k most interesting patterns wrt. distributional unusualness with sufficient frequency. A pattern p is accepted if $\frac{|cov(p)|}{|P|} \geq \sigma_m$ and $du(p) > min\{du(h) \mid h \in H\}$ where H is the set of patterns accepted so far. When a pattern p is accepted, it is added to H if $|H| < k$. If $|H| = k$, a new pattern p is accepted by substituting it to the h with minimal $du(h)$ in H.

Impint. The interestingness function of the third task setting impint is implication intensity II. As explained above, a pattern p is accepted iff $II(p) \geq 1 - \alpha$ and $\frac{cov(p)}{|P|} \geq \sigma_i$. In the impint setting, acceptance is restricted to most general patterns; consequently, accepted patterns are excluded from further specialization.

B.4.2 Databases and search spaces

The experiments were conducted with two databases, namely the well-known King-Rook-King (KRK) database describing chess boards of the KRK end game, and a phonetics database. For each database, a search space is defined in two variants. The pattern languages of both variants are identical, i.e., both search spaces include the same patterns. The difference is that the tax variant models taxonomies and generalization relationships between property literals, whereas the no_tax variant does not. Consequently, the no_tax variant does not offer the possibility of pruning based on taxonomies and generalization relationships between property literals. For the assoc and midos task setting, the search through the tax and the no_tax variant of a search space discovers the same set of acceptable patterns. For the impint setting, this is not the case because the restriction to most general patterns interacts with the taxonomy information. The taxonomy induces subsumption relationships between patterns that allow to prune the subsumed patterns due to the restriction to most general patterns. In the no_tax variant of the search space, these subsumption relationships are not known, and more patterns have to be accepted.

The KRK database. The KRK chess dataset represents chess board positions in the KRK endgame with three pieces left on the board: White King (*WK*), White Rook (*WR*), and Black King (*BK*). It contains 20,000 facts of the form

$$krk(\mathit{Id}, Class, \mathit{WKC}, \mathit{WKR}, \mathit{WRC}, \mathit{WRR}, \mathit{BKC}, \mathit{BKR}).$$

This is the only linkage literal of the application. The variable Id stands for identifiers of board positions, $Class$ takes the value 1 for illegal board positions and 0 for legal board positions. *WKC* is the column position of *WK*, *WKR* is the row position of *WK* and so on. Column and row positions are represented as integers ranging from 0 to 7. The population is defined as (distinct) ground instantiations of the variable Id in the literal $krk(\mathit{Id}, Class, \mathit{WKC}, \ldots)$. The property literal $\mathit{Class} = 1$ identifies target objects.

Thirty property literals are defined that compare row positions or column positions of the chess pieces. E.g., the following property literals refer to the column positions of *WK* and *BK*: "$\mathit{WKC} = \mathit{BKC}$", "$\mathit{WKC} \neq \mathit{BKC}$", "$\mathit{WKC} > \mathit{BKC}$", "$\mathit{WKC} < \mathit{BKC}$", and "$N1\ is\ \mathit{WKC} - \mathit{BKC}, (N1 = 1; N1 = -1; N1 = 0)$". The last literal defines $adjacent(\mathit{WKC}, \mathit{BKC})$.

For the tax variant, the system was provided with the information that property literals of the form $X \neq Y$ are more general than property literals of the form $X < Y$ and $X > Y$ and that property literals $adjacent(X, Y)$ are more general than property literals of the form $X = Y$.

The Phonetics database. The phonetics database provides scientific data on German speech recordings, collected and prepared for scientific phonetics analysis [Rap98]. The database consists of three relations *syllables* with 32 attributes and 18109 tuples, *phonemes* with 10 attributes and 48093 tuples, and *words* with 7 attributes and 7867 tuples. In order to limit the size of the pattern language, property literals are defined only for a selection of the available attributes. The abbreviated database schemas are as follows:

$$syllables(\mathit{SId}, \mathit{WId}, \mathit{LExp}, \mathit{LMes}, \mathit{D2P}, \mathit{TAlign}, \mathit{PHeight}, Acc)$$
$$phonemes(\mathit{PId}, \mathit{SId}, Phoneme, Type)$$
$$words(\mathit{WId}, Tag)$$

The attributes *SId*, *WId* and *PId* identify instances of syllables, words, and phonemes. They are primary keys of the respective relations. The experiments investigate the population of syllables, identified by the key attribute *SId* of relation $syllables$. The population literal is $syllables(\mathit{SId}, \mathit{WId}, \ldots, Acc)$. The attribute Acc has value 1 if the respective syllable has a pitch accent, otherwise, its value is 0. The target group is defined by "$Acc = 1$".

The attributes *LExp*, *LMes*, *D2P*, *TAlign*, *PHeight* are continuous numerical attributes of the spoken syllables. Eight property literals compare these numerical attributes of a syllable with expected values or with the average value of syllables in that attribute, e. g., "*LMes* $<$ *LExp*".

The link between syllables and words is established via the variable *WId*. The tag of a word is the part-of-speech tag, i.e., it marks the function of the word in the context of the sentence, e.g., noun, verb, etc. [STST99]. For tags, a taxonomy is available, with maximum depth 3 and maximum branching factor 9. There are 40 basic tags, as defined in the Stuttgart-Tübingen-Tag-Set [Rap98], and 14 generalized tags. E.g., tag adja (attributives Adjektiv) and tag adjd (adverbiales or prädikatives Adjektiv) generalize to adj (Adjektiv). Correspondingly, 54 property literals are defined, e.g., "$word($*WId*$, Tag), Tag = adja$", "$word($*WId*$, Tag), Tag = adjd$", or "$word($*WId*$, Tag), (Tag = adja; Tag = adjd)$" for generalized tags. As a syllable belongs to exactly one word, and a word has exactly one tag, the language is restricted such that, in each pattern, at most one property literal concerning tags is allowed in the no_tax variant of the search space. The tax variant allows co-occurrences of such patterns only for the *genex* representation.

Further property literals refer to the kind and type of the phonemes of a syllable. There are 39 distinct kinds of phonemes and 8 types that generalize the phonemes. The link between syllables and phonemes, that is defined via the variable *SId*, is non-determinate, since a syllable, in general, consists of several phonemes. Patterns can describe combinations of phoneme kinds or phoneme types in syllables. For instance, the following pattern describes syllables containing a short vowel and a fricative:

$$syllables(\textit{SId}, \textit{WId}, \dots), phonemes(\textit{PId}1, \textit{SId}, P1, T1),$$
$$T1 = short, phonemes(\textit{PId}2, \textit{SId}, P2, T2), T2 = fric.$$

B.4.3 Search runs

Four series of search runs are conducted in each task setting. A first series of search runs ("all") exploits all available possibilities to prune the search space. In the assoc setting, the subset condition and subsumption relationships between pairs of property literals are exploited for pruning. In the midos and impint settings, additionally, the best available optimum estimates were computed, i. e., du_{oe1} and du_{oe2} in the midos setting, and II_{oe2} in the impint setting.

The second series evaluates the contribution of optimum estimates to the reduction of search costs. As the assoc setting does not offer any optimum estimates, no search runs for assoc are conducted in this series.

The third series of search runs searches the no_tax version of search spaces and, thus, makes no use of subsumption relationships between pairs of property literals for pruning.

The fourth series evaluates the reduction of search efforts gained by the restriction to most general patterns. These search runs solve a modified impint task setting where specializations of accepted patterns remain in the search space.

Each series consists of three search runs. The search runs were executed with different settings of a parameter of the acceptance criterion. In the assoc task setting, the acceptance criterion accepts all patterns with minimum confidence 0.8 and minimum frequency σ_a where σ_a takes values $0.1, 0.2, 0.3$ and 0.4 in different search runs. The confidence threshold influences the number of acceptable patterns, but has no effect on pruning. It is set to 0.8 for all search runs. In the midos setting, the parameter σ_m is $0.0001, 0.1, 0.2$ and 0.3 in different search runs. The parameter k is 10 in all search runs. In the impint setting, the parameter σ_i is fixed to $S = 0.005$ in all search runs. The parameter α takes values $0.0001, 0.005, 0.01, 0.05$. The parameter settings vary the acceptance criterion and, consequently, the number of acceptable patterns as well as the extent to which pruning is possible. In the assoc and midos settings, the acceptance criterion grows stricter with increasing σ_a and σ_i, respectively. In the impint setting, the acceptance criterion also works as

a pruning criterion: due to the restriction to most general patterns (aka), accepting a pattern allows to prune its specializations. With a tight acceptance criterion (i.e., small α), less pruning based on aka and more pruning by the other methods (subset condition, optimum estimates and taxonomy) is possible, and vice versa.

B.4.4 Results

The results of the search runs are summarized in tables 6.1 and 6.2. "Kandidaten/otm" is the number of patterns generated by the specialization operator, "Kandidaten/tm" is the number of patterns remaining for evaluation after pruning by the subset condition, "Zeit/tm" is the time in seconds required for checking the subset condition, "Zeit/gesamt" is the total run time of the search run in seconds. "Par" is the parameter setting (σ_a for assoc, σ_m for midos, α for impint) of the search run. All search runs except the one marked with $*$ have completed the search. The search run marked with $*$ was stopped after about 22.5 hours. Until then, 77849 patterns had been evaluated.

Apriori-like pruning versus basic pruning. Figure 6.2 illustrates the gain of pruning with the subset condition ("all") versus basic pruning based on a specialization operator ("ohne_tm"). The number of patterns that have to be evaluated is depicted on the y-axis, different parameter values are depicted on the x-axis. The graphs show that the effect of complete pruning is quite reliable, i. e., it occurs in all search settings. It reduces the number of patterns that have to be evaluated by about 25 to 50%. A comparison of the columns "Zeit/tm" and "Zeit/gesamt" in tables 6.1 and 6.2 makes clear that the cost of checking the subset condition is negligible compared to the total run time.

Pruning based on taxonomies. Figure 6.3 compares the numbers of evaluated patterns in the tax variant of the search space ("all") to the numbers of evaluated patterns in the no_tax variant ("ohne_tax"). The graphs illustrate that the effect of pruning based on the taxonomies is not uniform over the different search runs. The result of the search run with $\sigma_m = 0.1$ with the KRK database in the midos setting is remarkable. Although the taxonomy based pruning clearly reduces the number of patterns, the search run takes longer (see table 6.2). The reason probably is that the taxonomy affects the order in which patterns are evaluated. The acceptance criterion of the midos k best search is dynamical; as the search progresses, better and better patterns are discovered, and the acceptance criterion becomes stricter.

In the no_tax search run, some patterns that are expensive to evaluate can be pruned, whereas, in the tax search space, they have to be evaluated earlier when the acceptance criterion is still too weak to prune these patterns.

Effect of optimum estimates. The graphs in figure 6.4 illustrate the contribution of optimum estimates to pruning of the search space. In the midos setting, the graph labeled "all" refers to search runs using optimum estimates du_{oe1} and du_{oe2}. The graph labeled "schwach_oe refers to search runs that use the optimum estimates du_{oe2} and the weaker version of du_{oe1}, du_{oe3}. The graph labeled "ohne_oe" refers to search runs that employ du_{oe2} and neither du_{oe1} nor du_{oe3}. In the impint setting, the graph labeled "all" refers to search runs using optimum estimate dII_{oe2}. The graph labeled "schwach_oe" refers to search runs that use the weaker optimum estimate II_{oe1}. The graph labeled "ohne_oe" refers to search runs that do not use an optimum estimate for pruning. The figure shows that optimum estimates are a powerful and reliable means for pruning the search space. Furthermore, the impint search runs demonstrate that a minor improvement of an optimum estimate can have a great effect on its power.

Restriction to most general patterns. Figure 6.5 shows the number of evaluated patterns when the restriction to most general patterns applies ("all") versus the number of evaluated patterns when the restriction is waived ("ohne_aka"). The restriction leads to a considerable reduction of the size of the search space. Table 6.13 gives the corresponding numbers of accepted patterns.

B.5 Summary and conclusions

The thesis intends to contribute to the development or improvement of algorithms for multi-relational subgroup discovery. Its main contributions are the ILP language bias and search algorithm for multi-relational subgroup discovery and the experiments on the effectiveness of pruning methods for subgroup discovery.

The experiments provide a comparison of different methods for pruning the search space for subgroup discovery in an ILP framework. The results, in particular, are that optimum estimates and the subset condition have produced good and reliable pruning effects, and that pruning based on the subset condition can have a similarly good effect for search in a multi-relational data base as it has for search in a single-relation database. Instead of the subset condition, ILP systems typically use θ-subsumption for detecting subsumption relationships between patterns. However, computation of θ-subsumption is very costly. Some existing ILP systems, therefore, only use what here is termed basic pruning for restricting the search space. The results of this paper indicate that it is worthwhile to design the pattern language or search space such that additional subsumption relationships can be exploited for pruning.

The language bias and search algorithm developed in the thesis allow to apply the subset condition for search in multi-relational subgroup discovery. They uniformly integrate pruning based on taxonomies, numerical attributes, and other subsumption relationships between features of individuals. A further interesting characteristic of the approach is that it is also applicable as an SQL language bias.

The application of the approach to the discovery challenge of PKDD'99 has shown that the language bias is well suited for the task of relational subgroup discovery, and that its expressivity is practically useful. Its discoveries were comparable to those made by similar data mining systems.

Further work. The ILP language bias developed here allows for flexible and fine-grained declaration of pattern languages for specific applications. At present, it is a drawback that the declaration of a pattern language can become quite complicated and extensive. This could be remedied by adding a more abstract, higher-level language bias to the system such that the necessary low-level declarations can be generated automatically out of the higher-level declarations. The development of a suitable high-level language bias specifically adapted to subgroup discovery and, ideally, usable via the graphical interface of an integrated data mining system seems a promising direction for future work.

The additions to the Apriori search procedure, namely, the *requires* and *excludes* conditions, as well as the *genex* approach for the treatment of generalized and quantitative attributes that is realized via *requires* and *excludes* conditions are also applicable for classical association mining. Their practical usefulness for this task is still to be investigated.

Literaturverzeichnis

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference on Management of Data*, Washington, D.C., 1993.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and I. Verkamo. Fast discovery of association rules. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[AS96] R. Agrawal and K. Shim. Developing tightly-coupled data mining applications on a relational database system: Methodology and experience. In *Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, 1996.

[BDR96] H. Blockeel and L. De Raedt. Relational knowledge discovery in databases. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 1–13. Stockholm University, Royal Institute of Technology, 1996.

[Ber99a] P. Berka. Guide to the financial data set. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[Ber99b] P. Berka, editor. *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[BMK98] I. Bratko, S. Muggleton, and A. Karalič. Applications of Inductive Logic Programming. In R.S. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining*. John Wiley and Sons Ltd., Chichester, 1998.

[BR98] Hendrik Blockeel and Luc De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.

[BS99] L. Badea and M. Stanciu. Refinement operators can be (weakly) perfect. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer-Verlag, 1999.

[BW98] O. Büchter and R. Wirth. Discovery of association rules over ordinal data: A new and faster algorithm and its application to basket analysis. In *Proc. PAKDD-98*, 1998.

[BZ99] P. Berka and J. Zytkow. Preface. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[CCK+99] P. Chapman, J. Clinton, T. Khabaza, T. Reinartz, and R. Wirth. The CRISP-DM process model, March 1999. http://194.19.161.12/.

[CCK+00] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. CRISP-DM 1.0 step-by-step data mining guide. http://www.crisp-dm.org/, 2000.

[CGT90] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer–Verlag, Berlin Heidelberg, 1990.

[CHA99] D. Coufal, M. Holeňa, and Sochorová. A. Coping with discovery challenge by GUHA. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[Cla87] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*. Plenum Press, New York, 1987.

[DDR97] L. Dehaspe and L. De Raedt. Mining association rules in multiple relations. In N. Lavrac and S. Dzeroski, editors, *Proc. Seventh International Workshop on Inductive Logic Programming (ILP'97)*, volume 1297 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1997.

[Deh98] Luc Dehaspe. *Frequent pattern discovery in first-order logic*. PhD thesis, K. U. Leuven, December 1998.

[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous attributes. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, 1995.

[DL97] T. G. Dietterich and T. Lathrop, R. H. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[DL01] S. Džeroski and N. Lavrač, editors. *Relational Data Mining*. Springer, 2001.

[DR98] L. De Raedt. Attribute value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *Proceedings of the 8th International Workshop Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 1–8. Springer-Verlag, 1998.

[DRB93a] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proc. of IJCAI-93, Chambéry*, 1993.

[DRB93b] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In S. Muggleton, editor, *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, pages 25–40. J. Stefan Institute, 1993.

[DRVL95] L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Conference on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[DT99] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 1(3):7–36, 1999.

[DT01] L. Dehaspe and H. Toivonen. Discovery of relational association algorithms. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*. Springer, 2001.

[Dže95] S. Džeroski. *Numerical constraints and learnability in inductive logic programming*. PhD thesis, Faculty of Electrical Engineering and Computer Science, University of Ljubljana, Ljubljana, Slovenia, 1995.

[Dže96a] S. Džeroski. Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 118–152. The MIT Press, 1996.

[Dže96b] Sašo Džeroski. Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[Dže01] S. Džeroski. Relational data mining applications: An overview. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*. Springer, 2001.

[Fay97] U. Fayyad. Knowledge discovery in databases: An overview. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *Lecture Notes in Artificial Intelligence*, pages 3–16. Springer-Verlag, 1997.

[Fay98] U. Fayyad. Mining databases: towards algorithms for knowledge discovery. *Bulletin of the technical committee on data engineering*, 21(1), march 1998. special issue on Mining of large datasets.

[FDPB95] L. Fleury, C. Djeraba, J. Philippe, and H. Briand. Contribution of the implication intensity in rules evaluations for knowledge discovery in databases. In Y. Kodratoff, G. Nakhaeizadeh, and C. Taylor, editors, *Workshop Notes of the ECML-95 Workshop Statistics, Machine Learning and Knowledge Discovery in Databases*, 1995.

[Fei98] J. Feist. Zur Diskussion: Data Mining in der Praxis. *Künstliche Intelligenz*, 1, 1998.

[FL99] P. Flach and N. Lachiche. 1BC: A first-order Bayesian classifier. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 92–103. Springer-Verlag, 1999.

[FL01] P. Flach and N. Lachiche. Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning*, 42(1/2):61–95, 2001.

[Fla95] P.A. Flach. *Conjectures: An Inquiry Concerning the Logic of Induction*. PhD thesis, Katholieke Universiteit Brabant, 1995.

[Fla99a] Peter A. Flach. Knowledge representation for inductive learning. In Anthony Hunter and Simon Parsons, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'99)*, volume 1638 of *Lecture Notes in Artificial Intelligence*, pages 160–167. Springer-Verlag, July 1999.

[Fla99b] Peter A. Flach. Knowledge representation for inductive learning. Presentation at ECSQARU'99, 1999.
www.cs.bris.ac.uk/~flach/presentations/ECSQARU99HTML/sld001.htm

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[FPSSU96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[GL93] R. Gras and A. Larher. L'implication statistique, une nouvelle méthode d'analyse de données. *Mathématique, Informatique et Sciences Humaines*, (120), 1993.

[Han99] J. Han. Data mining. In *Encyclopedia of Distributed Computing*. Kluwer Academic Publishers, 1999.

[Hel89] N. Helft. Induction as nonmonotonic inference. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 149–156. Morgan Kaufmann, 1989.

[HF95a] J. Han and Y. Fu. Discovery of multi-level association rules from large databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95)*, pages 420–431, Zurich, Switzerland, 1995.

[HF95b] J. Han and Y. Fu. Meta-rule-guided mining of association rules in relational databases. In *Proc. of 1995 Int'l Conf. on Knowledge Discovery and Deductive and object-oriented databases (KDOOD'95)*, pages 420–431, Singapore, 1995.

[HK91] P. Hoschka and Willi Klösgen. A support system for interpreting statistical data. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 19, pages 325–345. AAAI/MIT Press, Cambridge, MA, 1991.

[HMWG98] J. Hipp, A. Myka, R. Wirth, and U. Güntzer. A new algorithm for faster mining of generalized association rules. In *Proc. 2nd PKKD*, 1998.

[JN02a] D. Jensen and J. Neville. Linkage and autocorrelation cause bias in accuracy estimates. In *Proceedings of the 12th International Workshop on Inductive Logic Programming*. Springer, 2002.

[JN02b] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufman, 2002.

[KL94] J-U. Kietz and M. Lübbe. An efficient subsumption algorithm for inductive logic programming. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 97–106. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.

[KLF01] S. Kramer, N. Lavrač, and P. Flach. Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*. Springer, 2001.

[Klö96] W. Klösgen. Explora: a multipattern and multistrategy discovery assistant. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[Klö99] W. Klösgen. Applications and research problems of subgroup mining. In *Proc. ISMIS'99*, number 1609 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999.

[Klu97] D. Klunzinger. Darstellung von Assoziationsregeln im Kontext Warenkorbanalyse. Diplomarbeit Nr. 1537, Universität Stuttgart, Fakultät Informatik, 1997.

[KMV99] M. Klemettinen, H. Mannila, and I. Verkamo. Association rule selection in a data mining environment. In *Principles of Data Mining and Knowledge Discovery (PKDD'99)*, number 1704 in Lecture Notes in Artificial Intelligence, pages 372 – 377. Springer-Verlag, 1999.

[Kod99] Y. Kodratoff. Knowledge discovery in texts: A definition and applications. In *Proc. ISMIS'99*, number 1609 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999.

[KW92] J.U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive Logic Programming*. Academic Press, 1992.

[KZ] Willi Kloesgen and Jan Zytkow. Machine discovery terminology. http://orgwis.gmd.de/projects/explora/terms.html.

[LD94] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

[LF00] Nicolas Lachiche and Peter Flach. A first-order representation for knowledge discovery and bayesian classification on relational data. In Pavel Brazdil and Alipio Jorge, editors, *PKDD2000 workshop on Data Mining, Decision Support, Meta-learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*, pages 49–60. 4th International Conference on Principles of Data Mining and Knowledge Discovery, September 2000.

[LF01] Nada Lavrač and Peter Flach. An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, 4(2), 2001.

[Lin94] Guido Lindner. Logikbasiertes Lernen in relationalen Datenbanken. Technical Report LS-8 Report 12, Universität Dortmund, Fachbereich Informatik, 1994.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, New York, 2nd edition, 1987.

[LM95] G. Lindner and K. Morik. Coupling a relational learning algorithm with a database system. In Y. Kodratoff, G. Nakhaeizadeh, and C. Taylor, editors, *Workshop Notes of the ECML-95 Workshop Statistics, Machine Learning and Knowledge Discovery in Databases*, 1995.

[LMC+99] B. Levin, A. Meidan, A. Cheskis, O. Gefen, and I. Vorobyov. PKDD'99 discovery challenge — financial domain. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[LWZ+96] N. Lavrač, I. Weber, D. Zupanič, D. Kazakov, O. Štěpánková, and S. Džeroski. ILPNET repositories on WWW: Inductive Logic Programming systems, data sets and bibliography. *AI Communications*, 9(4), 1996.

[MB88] S. Muggleton and W. Buntine. Machine invention of first order predicates by inverting resolution. In *Proceedings of the 5th International Workshop on Machine Learning*, pages 339–351. Morgan Kaufmann, 1988.

[MDR94] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[MT97a] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[MT97b] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. Technical Report C-1997-8, University of Helsinki, Department of Computer Science, 1997.

[MTV94] H. Mannila, H. Toivonen, and I. Verkamo. Efficient algorithms for discovering association rules. In *AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[Mug95] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[Mül99] Michael Müller. *Interessantheit bei der Entdeckung von Wissen in Datenbanken*. PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1999.

[MWKE93] K. Morik, S. Wrobel, J. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications*. Academic Press, 1993.

[MŽŠP99] P. Mikšovský, F. Železný, O. Štepánková, and M. Pěchouček. Financial data challenge. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[NCdW97] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, February 1997.

[NRA+96] C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, 1996.

[Pij99] W. Pijls. Discovery challenge, financial data. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[Plo70] G. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

[Plo71] G. Plotkin. *Automatic methods of inductive inference*. PhD thesis, University of Edingburgh, 1971.

[PS98] G. Piatetsky-Shapiro. Data mining and knowledge discovery: The next generation (slides), 1998.
http://www.kdnuggets.com/gpspubs/gps-pub-recent.html.

[Put99] P. van der Putten. Promoting credit card usage by mining transaction data. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[Qui90] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Rae97] L. De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, pages 187–201, 1997.

[Rae00] L. De Raedt. Towards practical ILP. In *Proceedings of the PKDD 2000 Workshop on Data Mining,Decision Support, Meta-learning and ILP: Forum for Practical Problems*, Lyon, Sept 2000.

[Rap98] S. Rapp. Automatic labelling of german prosody. In *Proc. of Int. Conference on Spoken Language Processing (ICSLP'98)*, 1998.

[RD94] L.De Raedt and S. Džeroski. First order jk clausal theories are pac-learnable. *Artificial Intelligence*, 70:275–392, 1994.

[Rei84] R. Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. Schmidt, editors, *On conceptual modeling: perspectives from artificial intelligence, databases, and programming languages*. Springer, 1984.

[Rei87] Raymond Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Databases*. Plenum Press, New York, 1987.

[Rob65] J.A. Robinson. A machine oriented logic based on the resolution principle. *JACM*, 12:23–41, 1965.

[SA95a] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the VDLB Conference*, Zürich, Schweiz, 1995.

[SA95b] R. Srikant and R. Agrawal. Mining generalized association rules (expanded version). Technical Report RJ9963, IBM Research Division, Almaden Research Center, San Jose, CA, 1995.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. of the ACM Sigmod Conference on Management of Data*, Montreal, Canada, 1996.

[Sam93] C. Sammut. The origins of inductive logic programming: A prehistoric tale. In S. Muggleton, editor, *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, pages 127–148. J. Stefan Institute, 1993.

[SB99] M. Spenke and C. Beilken. Visual, interactive data mining with InfoZoom — the financial data set. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[Sch87] J. W. Schmidt. Datenbankmodelle. In *Datenbank-Handbuch*. Springer-Verlag, 1987.

[Sch92] U. Schöning. *Logik für Informatiker*. BI Wissenschaftsverlag, 1992.

[ST96] A. Silberschatz and A. Tuzhilin. what makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8:970–974, December 1996.

[STA98] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with databases: alternatives and implications. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Seattle, Washington, 1998.

[STST99] A. Schiller, S. Teufel, C. Stöckert, and C. Thielen. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart und Seminar für Sprachwissenschaft, Universität Tübingen, 1999.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proc. of the 3rd Int. Conference on Knowledge Discovery in Databases and Data Mining*, 1997.

[Tau94] B. Tausend. *Beschränkungen der Hypothesensprache und ihre Repräsentation in der Induktiven Logischen Programmierung*. Dissertation, Fakultät Informatik, Universität Stuttgart, 1994.

[TS98] S. Thomas and S. Sarawagi. Mining generalized association rules and sequential patterns using SQL queries. In *Proceedings of the 4th International conference on Knowledge Discovery and Data Mining (KDD 98)*, New York, 1998.

[Ull82] J. D. Ullman. *Principles of Database Systems*. Computer Science Press, Rockville, MD, 1982.

[WD95] S. Wrobel and S Džeroski. The ILP description learning problem: Towards a general model-level definition of data mining in ILP. In Morik K. and J. Herrmann, editors, *Proceedings of the annual workshop of the GI Special Interest Group Machine Learning (GI FG 1.1.3)*, Research Report 580. Univ. Dortmund, 1995.

[Web97] I. Weber. Discovery of first-order regularities in a relational database using offline candidate determination. In N. Lavrac and S. Dzeroski, editors, *Proc. Seventh International Workshop on Inductive Logic Programming (ILP'97)*, volume 1297 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1997.

[Web98a] I. Weber. A declarative language bias for levelwise search of first-order regularities. In F. Wysotzki, P. Geibel, and C. Schädler, editors, *Proc. FGML-98*, Technical Report 98/11 des Fachbereiches Informatik, TU Berlin, 1998. (Vorversion).

[Web98b] I. Weber. On pruning strategies for discovery of generalized and quantitative association rules. In L. Bing, W. Hsu, and W. Ke, editors, *Proc. of Knowledge Discovery and Data Mining Workshop, (Pricai'98)*, 1998.

[Web99a] I. Weber. A declarative language bias for levelwise search of first-order regularities. In *Proc. ISMIS'99*, number 1609 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999.

[Web99b] I. Weber. Discovery of interesting rules and subgroups in a financial database. In P. Berka, editor, *Workshop Notes on Discovery Challenge. Workshop at 3rd European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD'99)*, 1999.

[Web00] I. Weber. Levelwise search and pruning strategies for first-order hypothesis spaces. *Journal of Intelligent Information Systems*, 14(2–3):217–239, 2000.

[Web02] I. Weber. Experimental investigation of pruning methods for relational pattern discovery. In S. Matwin and C. Sammut, editors, *Proc. Twelfth International Workshop on Inductive Logic Programming (ILP'02)*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2002.

[Wel99] Oliver Welte. Evaluierung eines Prozeßmodells für Data Mining Projekte. Diplomarbeit 1705, Stuttgart, Univ., Fakultät Informatik, Diplomarbeit Nr. 1705, Stuttgart, 1999.

[Wir89] R. Wirth. Completing logic programs by inverse resolution. In K. Morik, editor, *Proceedings of the 4th European Working Session on Learning*. Pitman, 1989.

[WMJ00] S. Wrobel, K. Morik, and Th. Joachims. Maschinelles Lernen und Data Mining. In G. Görz, C.-R. Rollinger, and J. Schneeberger, editors, *Handbuch der Künstlichen Intelligenz*, (3. Auflage). Oldenbourg, 2000.

[Wro97] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In J. Komorowski and J. Zytkow, editors, *Proc. First European Symposium on Principles of Knowledge Discovery and Data Mining*. Springer, 1997.

[WWSE96] Stefan Wrobel, Dietrich Wettschereck, Edgar Sommer, and Werner Emde. Extensibility in data mining systems. In Evangelos Simoudis, Jia Wei Han, and Usama Fayyad, editors, *Proc. 2nd International Conference On Knowledge Discovery and Data Mining*, pages 214 – 219. AAAI Press, 1996.