Maschinelles Lernen
Skript zur Vorlesung

Katharina Morik*

12. April 1999

*Anregungen und Kritik bitte per e-mail an morik@ls8.informatik.uni-dortmund.de

Inhaltsverzeichnis

1 Einführung

Das maschinelle Lernen gehört zu den Fähigkeiten, deren Verfügbarkeit auf einem Rechner bereits als Ziel formuliert wurde, als der erste praktische Rechnereinsatz mit ENIAC in Philadelphia gelungen war. Die Idee dabei war, daß Programmierer von Routinearbeiten entlastet und Programme schneller erstellt werden sollten. Für Alan Turing war die Lernfähigkeit eines Rechners die wichtigste Intelligenzleistung [Turing, 1987]. Er empfahl, einen Rechner „zu erziehen“, so daß er seine Leistungen verbessert, da man unmöglich alles einprogrammieren könne. Insofern waren maschinelles Lernen und Programmsynthese damals noch nicht unterschieden. Diese Unterscheidung kam erst mit den Produktionensystemen, die ja „Wissen“ von dem Verarbeitungsprogramm (Interpreter) unterschieden. Maschinelles Lernen wurde dann eingegrenzt auf den automatischen Erwerb von Regeln oder die Verbesserung von Regelmengen. Diese Eingrenzung ergab sich einmal aus der psychologischen Motivation von Produktionensystemen. Produktionen wurden ja von [Simon, 1978] gerade deshalb als adäquate Repräsentation kognitiver Einheiten ausgewählt, weil sie dadurch, daß sie klein, gleich strukturiert und nicht miteinander verzahnt seien, leichter erlernbar sind. Zum anderen folgte die Eingrenzung aus der praktischen Notwendigkeit, den Regelerwerb für Expertensysteme zu unterstützen. Heute sind Produkte zum induktiven Regelerwerb auf dem Markt erhältlich und das Gebiet ist wieder breiter, in seinen Fragestellungen vielfältiger geworden.

1.1 Definitionen des Lernens

Die erste Frage, die meist gestellt wird, ist, wie wir Lernen definieren können, so daß wir einem Rechner eine – eingeschränkte – Lernfähigkeit zusprechen können. Die bekannte Definition von [Simon, 1983] lautet:

Definition 1: Lernen (nach Simon). *Lernen ist jede Veränderung eines Systems, die es ihm erlaubt, eine Aufgabe bei der Wiederholung derselben Aufgabe oder einer Aufgabe derselben Art besser zu lösen.*

Diese Definition ist aus zwei Gründen kritisiert worden: sie deckt auch solche Phänomene ab, die man üblicherweise nicht als Lernen bezeichnet, und sie deckt nicht alle dem Lernen zugerechneten Phänomene ab. Ein Beispiel dafür, daß Lernen nicht der einzige Grund für eine verbesserte Leistung ist, stammt von [Michalski, 1986]. Wenn es die Aufgabe ist, etwas zu schneiden, so wird die Leistung dadurch verbessert, daß man ein schärferes Messer nimmt. Das Messerschärfen ist aber kein Lernen. Nur das Herausfinden, daß mit Hilfe eines schärferen Messers das Schneiden zu verbessern ist, wäre Lernen. Simons Definition kann diese beiden Fälle aber nicht unterscheiden. Auch das zufällige Verwenden eines schärferen Messers würde seine Definition erfüllen. Die Lernfähigkeit von Programmen könnte gemäß Simons Definition dadurch nachgewiesen werden, daß wir dasselbe Programm auf einem schnelleren Rechner laufen ließen. Das System, Rechner und Programm, würde dann dieselbe Aufgabe schneller lösen. Als Verbesserung der Definition könnte man vorschlagen, daß alle Teile eines Systems verändert werden, um die Leistung zu steigern. Das würde dann aber das Hinzufügen einer Regel und die dadurch gesteigerte Leistung eines regelbasierten Systems bei gleichbleibendem Interpreter ausschließen. Man hätte dann gerade die Methode ausgeschlossen, die maschinelles Lernen erst ermöglichte, nämlich die Trennung von lernbaren Einheiten und nicht-lernbarer Verarbeitung.

Michalski gibt auch ein drastisches Beispiel dafür an, daß Leistungssenkung ein Lernergebnis sein kann [Michalski, 1986]. Wenn Zwangsarbeiter einen Weg finden, wie sie weniger leisten können, so wäre das ein Beispiel für Lernen. Sie könnten lernen, wie man weniger tut und doch gleich beschäftigt aussieht. Damit weist Michalski auf die Zielabhängigkeit des Leistungsbegriffs hin. Die Arbeiter verbessern ihre Leistung der Vortäuschung und verschlechtern ihre Arbeitsleistung. Je nachdem, wie man die Aufgabe definiert, fällt ihre Tätigkeit unter Simons Definition, oder nicht. Scott argumentiert gegen die Leistungsmessung bei der Definition vom Lernen [Scott, 1983]. Er führt als Beispiel einen Spaziergänger in einer ihm noch unbekannten Stadt an, der an der öffentlichen Bibliothek vorbeikommt. Während er diese wahrnimmt, lernt er etwas über die Stadt, ohne irgendeine Aufgabe zu haben, für deren Lösung er wissen muß, ob und wo es eine Bibliothek gibt. Erst wenn ein Passant ihn nach dem Weg zur Bibliothek fragt, kann er das Gelernte einsetzen – und zwar schon beim ersten Passanten, nicht erst bei der Wiederholung. Simon hat einen Test angegeben, der auch bei dem Spaziergänger ergeben würde, daß der gelernt hätte, jedoch keine Definition. Der Test gehört nicht zum Lernen selbst. Der Spaziergänger lernt unabhängig davon, ob er getestet wird. [Scott, 1983] definiert Lernen ohne Rückgriff auf eine gegebene Leistung:

Definition 2: Lernen (nach Scott). *Lernen ist ein Prozeß, bei dem ein System eine abrufbare Repräsentation von vergangenen Interaktionen mit seiner Umwelt aufbaut.*

Damit ist die Leistung potentiell beobachtbar, weil die neue Repräsentation abrufbar ist. Das Lernen selbst ist aber unabhängig davon, ob sein Ergebnis jemals gebraucht wird. Auch wird eine Leistungssenkung durch Lernen nicht ausgeschlossen. So könnte jemand, der nur eine einzige Aussage über etwas weiß, wenn genau nach dieser gefragt wird, womöglich schneller antworten als jemand, der erst aus der Fülle seiner Informationen die passende heraussuchen muß. Ähnlich ist auch Michalskis Definition [Michalski, 1986]:

Definition 3: Lernen (nach Michalski). *Lernen ist das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.*

Beide Definitionen setzen einen Prozeß voraus, der Repräsentationen verwendet. Wieweit dieser durch Lernen aufgebaut oder verändert wird, bleibt offen. Schon aus dieser kurzen Diskussion über die Definition von Lernen wird deutlich, daß Lernen ähnlich schwierig zu fassen ist wie Intelligenz. Es bleibt unser umgangssprachliches Verständnis von dem, was für uns Lernen ist, als Anregung und als Richtschnur.

1.2 Drei Motivationen für das maschinelle Lernen

Maschinelles Lernen hat – wie alle anderen Teilgebiete der KI – drei verschiedene Motivationen: eine kognitionswissenschaftliche, eine theoretisch-technische und eine praktische, anwendungsorientierte. Für das maschinelle Lernen sind die einzelnen Ziele:

- Prinzipien menschlichen Lernens sollen mit Hilfe von operationalen Modellen untersucht werden.
- Insbesondere der induktive Schluß soll operationalisiert werden, aber auch die Verwendung anderer Schlußfolgerungen (Deduktion und Abduktion) zum Lernen soll untersucht werden.
- Die Arbeit am Rechner soll durch dessen Lernfähigkeit dem Benutzer erleichtert werden.

Nach einem kurzen Überblick über diese drei Ausrichtungen konzentrieren sich die darauf folgenden Abschnitte auf Verfahren, also den zweiten Aspekt. Allerdings wird auch der Nutzen der Verfahren, sowohl in kognitionswissenschaftlicher Hinsicht als auch in industriellen Anwendungen, angeführt.

1.3 Menschliches und maschinelles Lernen

Die kognitive Orientierung verwendet psychologische Arbeiten zur Begriffsbildung. Die Struktur, Verwendung und der Erwerb von Begriffen bei Kindern sind Gegenstand vieler Untersuchungen. Hier werden nur einige zusammengefaßt, um einen Einblick in wichtige Fragestellungen zu geben. Literatur zum Einstieg in dieses Thema wird am Ende des Abschnitts angeführt.

Die Begriffsbildung kann in zwei Phänomenbereiche unterteilt werden: die Aggregation und die Charakterisierung oder Definition. Die **Aggregation** gruppiert Objekte, Ereignisse und Sachverhalte der Welt in Klassen oder Kategorien. Eine *Kategorie* ist die Extension eines Begriffs. Die **Charakterisierung** beschreibt eine Kategorie, so daß für neue Objekte entschieden werden kann, in welche Kategorie sie gehören. Die intensionale Beschreibung der Kategorie dient also zur Bestimmung der Klassenzugehörigkeit. Ein Objekt wird erkannt als Beispiel eines Begriffs, wenn die Charakterisierung des Begriffs das Objekt abdeckt. Ein *Begriff* ist eine mentale, kognitive Einheit, die sich auf eine Kategorie bezieht. Damit gibt es drei Phänomenbereiche:

$$\text{Aggregation} \rightarrow \text{Charakterisierung} \rightarrow \text{Klassifikation (Erkennung)}$$

Die Einteilung dient der Strukturierung wissenschaftlicher Arbeit. Die Phänomenbereiche sollen nicht als Phasen eines linearen Ablaufs beim Menschen verstanden werden.

1.3.1 Gründe für die Aggregation

Zunächst könnte man als einen guten Grund dafür, Objekte der Welt zu einer Kategorie zusammenzufassen, angeben, daß sie ein Merkmal gemeinsam haben. Sie sind sich ähnlich. Da aber Merkmale nicht bereits in der Welt vorkommen, sondern ihrerseits gebildet werden, könnten wir umgekehrt für jede Zusammenstellung von Objekten ein Merkmal einführen, das genau für diese Menge gilt. Bei k Objekten gibt es prinzipiell 2^k Mengen von Objekten. Tatsächlich verwenden Menschen aber nicht so viele Kategorien. Es muß also noch zusätzliche Gründe geben, warum Kategorien gebildet werden. Drei Gründe, die in der Literatur diskutiert wurden, werden im folgenden angeführt.

Einige Objekte spielen eine wichtige Rolle für bestimmte Handlungen. Damit begründen die Handlungen einen Bedarf für eine Kategorie. Wenn der Bedarf nur kurzfristig und einmalig ist, so werden Kategorien ad hoc gebildet und danach nicht weiterhin verwendet[1], ansonsten wird die Kategorie konventionalisiert. Quine sah in der Notwendigkeit, etwas vorhersagen zu können, das Motiv für individuelle und gesellschaftliche Kategorienbildung [Quine, 1977]. Ein neuer Begriff wird dann eingeführt, wenn er Objekte klassifizieren kann, deren Verhalten wir vorhersagen wollen. Als Beispiel für eine zunächst unsinnige Kategorie, die aber durch einen bestimmten Handlungszusammenhang sinnvoll werden kann, führen Murphy und Medin gestreifte Objekte mit mehr als einem Bein an, die zwischen 11 und 240 kg wiegen [Murphy und Medin, 1985]. Im Kontext eines Spielfilms, in dem diese Objekte Außerirdische sind, die die Menschheit bedrohen, wird die Kategorie sinnvoll. Es ist

[1] Zu ad hoc Kategorien siehe [Barsalou, 1983].

dann wichtig zu erkennen, wer dieser Kategorie angehört, wie er sich verhalten wird und wie Menschen sich vor ihm schützen können. Eine andere üblicherweise sinnlose Kategorie besteht aus Primzahlen und Äpfeln. Wenn dies aber die einzigen Gesprächsthemen für die Kollegin Wilma sind, so erhält die Kategorie einen Bezug zu anderen Kategorien (Wilma, Gespräche) und ist nicht mehr absurd [Murphy und Medin, 1985]. Murphy und Medin betonen die Begriffsstruktur, die unterschiedliche Begriffe im Zusammenhang repräsentiert. Erst durch den Zusammenhang wird eine Kategorie oder ein Begriff sinnvoll.

Ein Bedarf an Kategorien wird auch durch ihre Verwendung für die Charakterisierung anderer Kategorien gegeben. Zum Beispiel ist es sinnvoll, die Kategorie *Räder* zu bilden, wenn wir Fahrzeuge definieren wollen. Im Zuge der Charakterisierung von Fahrzeugen entsteht eine neue Begriffsbildungsaufgabe. Es ist einfach praktischer, einen Begriff *Räder* zu haben, als stets die zugehörigen Objekte aufzuzählen: schließlich umfaßt der Begriff eine potentiell unendliche Menge. Nebenbei hebt dieses Beispiel den Zusammenhang von Begriffen hervor: Begriffe werden nicht isoliert voneinander gebildet.

Oft untersuchen psychologische Experimente die Charakterisierung von Kategorien, die von den Psychologen vorgegeben werden. Dabei kann es sich um existierende oder um künstlich gebildete Zusammenstellungen von Objekten handeln. Zu der Kategorie des Belebten (*living thing*) gibt es seit Piaget eine Fülle von Untersuchungen, die verschiedene Charakterisierungsansätze jeweils einer bestimmten Altersstufe zuordnen. Eine Untersuchung von Susan Carey gibt obendrein Hinweise auf die Aggregation von Objekten [Carey, 1985]. In ihrem Exeriment sollten die Kinder zunächst belebte Objekte aufzählen. Das war für fast alle Kinder kein Problem. Man kann annehmen, daß sie diese Kategorie bereits vor dem Experiment kannten. Als sie aber Beispiele unbelebter Objekte anführen sollten, hatten die Kinder Schwierigkeiten. Sie gaben Beispiele für unbelebte Objekte, tote Menschen oder Tiere, Fabelwesen und Abbildungen von Menschen und Tieren (z.B. im Fernsehen) an. Also führten sie unterschiedliche Kategorien an, aus denen sie – möglicherweise ad hoc – *Nicht-Belebtes* bildeten. Interessant ist dabei, daß diese neue Kategorie unter verschiedenen Gesichtspunkten in Bezug auf die gegensätzliche Kategorie gebildet wurde. Dies ist ein weiterer Hinweis darauf, daß Kategorien und Begriffe im Zusammenhang gebildet werden.

1.3.2 Probleme der Charakterisierung

Der klassische Ansatz zur Erklärung der Begriffsbildung betrachtet die Charakterisierung als das Finden solcher Merkmale, die alle Beispiele bzw. Instanzen eines Begriffs gemeinsam haben. Ein Begriff ist dann durch eine Menge solcher Merkmale repräsentiert. Daß die Ähnlichkeit von Objekten nicht ausreicht, eine Kategorie zu bilden, haben wir bereits oben festgestellt. Aber auch zur Charakterisierung reichen ähnliche Merkmale nicht aus.

Das erste Problem des klassischen Ansatzes ist die **Herkunft der Merkmale**. Dimensionen wie Farbe, Größe oder Formen werden selbst erst gebildet, sie sind nicht vorgegeben. Merkmale stammen aus der Wahrnehmung. Land hat gezeigt, daß die Farbwahrnehmung nicht nur auf der Wellenlänge beruht, sondern ebenso auf der Textur des Objekts und der Lichtreflexion [Land, 1983]. Es liegt an dem menschlichen Körper, daß Farben so wahrgenommen werden. Ein Vogel mag Farben anders erfahren. Biologische Untersuchungen können also über einen Aspekt der Herkunft von Merkmalen Auskunft geben: ihre Verankerung in der Wahrnehmung (Stichwort: *symbol grounding*). Sie können jedoch nicht die kulturellen und situationsspezifischen Unterschiede der Wahrnehmung erklären. Lenneberg wies die Abhängigkeit der Farbwahrnehmung von der durch Wörter einer natürlichen

Sprache gegebenen Einteilung des Farbspektrums nach [Lenneberg, 1967]. Farben werden als mehr in der Mitte des Bereiches, der durch ihr Wort bezeichnet wird, wiedergeben, als sie wirklich waren. So wurde ein grünliches Blau blauer wahrgenommen, wenn die Sprache kein eigenes Wort für diesen Farbton besitzt. Türkis wird von Menschen, die das Wort „türkis“ in ihrem aktiven Sprachschatz haben, genauer von Blau abgegrenzt, als von solchen Versuchspersonen, die nur *blau* und *grün* verwenden. Damit werden Unterschiede von Farbtönen, die zum selben Begriff gehören, verringert. Gleichzeitig werden Unterschiede zwischen Farbtönen verschiedener Begriffe verstärkt. Auch der Einfluß der (situationsbedingten) Erwartungen auf die Farbwahrnehmung wurde erwiesen. Die übliche Farbe eines Objektes wird auch dann gesehen, wenn eigentlich eine andere gegeben ist. Es gibt also eine Rückwirkung des sprachlichen und begrifflichen Wissens auf die Wahrnehmung. Insofern erklärt die Verankerung von Merkmalen in der Wahrnehmung wenig.

Das zweite Problem des klassischen Ansatzes ist die **Auswahl von Merkmalen**. Selbst wenn wir einen Prozeß annähmen, der aus Wahrnehmungen Merkmale formt, so könnten zur Charakterisierung eines Begriffs doch fast unendlich viele Merkmale herangezogen werden. Weitere Einschränkungen sind nötig. Wie schon bei der Kategorienbildung kann auch bei der Charakterisierung die Definition anderer Begriffe zur Auswahl der Merkmale herangezogen werden. Begriffe werden im Zusammenhang definiert. Es werden Merkmale ausgewählt, die solche Begriffe unterscheiden, die nicht verwechselt werden sollen. Die Gegensatz-Beziehung von Begriffen wählt nur die Merkmale zur Charakterisierung aus, die für alle gegensätzlichen Begriffe anwendbar sind und sie unterscheiden. Carey beobachtete außerdem, daß Kinder, wenn sie einmal bestimmte Merkmale dafür benutzten, einen Begriff zu charakterisieren, gegensätzliche Begriffe mit anderen Werten derselben Merkmale definierten [Carey, 1985]. Dies wird *Konsistenz der Charakterisierung* genannt. Eine Folge dieses Prinzips ist, daß Änderungen eines Begriffs Folgen für seine Gegensatz-Begriffe haben. Zusammen mit der Gegensatzrelation zwischen Begriffen hilft die Unterbegriffsrelation bei der Merkmalsauswahl. Voneinander abzugrenzen sind ja nur solche Begriffe, die überhaupt verwechselbar sind. Insbesondere gegensätzliche Unterbegriffe desselben Oberbegriffs werden mit denselben Merkmalen beschrieben.

Susan Carey betont die Abhängigkeit der Begriffsstruktur von dem Wissen eines Menschen [Carey, 1985]. Die Definition des Belebten hängt ab von dem Wissensstand über Biologie. Auch Keil und Kelly zeigen, daß Versuchspersonen mit wenig Wissen über einen Sachbereich eher beschreibende Merkmale auswählen, während zur Verwendung definitorischer Merkmale mehr Wissen nötig ist [Keil und Kelly, 1987]. Die Verschiebung von Beschreibungen zu Definitionen ist damit nur indirekt einer Alterstufe zuzuschreiben – sie ist die Folge des wachsenden Wissens. Kinder wie Laien bevorzugen leicht erkennbare, Fachleute – und für das Alltagswissen sind Erwachsene Fachleute – nutzen gut abgrenzende Merkmale. [Murphy und Medin, 1985] sprechen von einem Netzwerk erklärender Merkmale. Sie geben dafür Beispiele an, daß eine Theorie Merkmale auszuwählen vermag und auch Merkmale korreliert. Biologische Theorien über das Wachstum von Pflanzen geben *besteht_aus_Zellen* und *wächst* den Vorzug vor der Farbangabe. Diese Merkmale hängen zusammen. Sie gelten für alle Pflanzen und werden also auch an z.B. Karotten vererbt. Würde man nun erfahren, daß Karotten gar nicht aus Zellen bestehen, so müßte man den Begriff *Pflanze* ändern. Trifft man hingegen auf blaue Karotten, so sind von dieser Änderung andere Begriffe nicht betroffen. Definitorische Merkmale kann man an dem Ausmaß der Konsequenzen für andere Begriffe erkennen. Definitorische Merkmale charakterisieren Oberbegriffe derart, daß sie an Unterbegriffe weitergegeben werden können.

Das dritte Problem des klassischen Ansatzes besteht in der **Begriffsrepräsentation**. Eine reine Ansammlung von Merkmalen strukturiert Begriffe nicht. Die Beziehungen zwischen Begriffen ebenso wie die Beziehungen zwischen Merkmalen scheinen aber sehr wichtig zu sein und sollten deshalb repräsentiert werden.

> *In order to characterize knowledge about and use of a concept, we must include all of the relations involving that concept and the other concepts that depend on it.* [Murphy und Medin, 1985, S. 297]

Begriffliche Gegensätze und Unterbegriffe sollten zusammen mit der Konsistenz ihrer Charakterisierungen und der Vererbung definitorischer Merkmale dargestellt werden. Wir können Merkmale verallgemeinern zu Begriffen, so daß die Relationen zwischen Merkmalen in derselben Weise behandelt werden wie Begriffsrelationen. Tatsächlich ist es ja nicht einzusehen, warum Zellen mal ein Merkmal sind (wenn wir Pflanzen beschreiben wollen), mal selbst der Begriff sind, der definiert werden soll (wenn wir über Zellen sprechen). Ist die Begriffsrepräsentation nur eine eine Liste von Merkmalen, so hängt sie von dem jeweiligen Gesprächsgegenstand ab. Werden Begriffe jedoch durch ihre Zusammenhänge untereinander dargestellt, so kann – ohne eine Änderung der Repräsentation – auf einen Begriff als Gesprächsgegenstand zugegriffen werden oder als Charakterisierung eines anderen Begriffes. Zum Beispiel können *Karottenfarbe* und *Aprikosenfarbe* als Unterbegriffe von *Modefarbe* genauso genutzt werden wie zur Charakterisierung der jeweiligen Pflanzen. Ein weiterer Vorteil der Vereinheitlichung von Begriffen und Merkmalen besteht in der Änderbarkeit der Begriffsstruktur. Wenn wir über Zellen etwas hinzulernen, ändert sich die Charakterisierung von Pflanzen, die ja aus Zellen bestehen, automatisch – wir müssen keinen zusätzlichen Prozeß annehmen, der dies neue Wissen in den Begriff Pflanze überträgt.

Schließlich soll nicht verschwiegen werden, daß auch diese Sicht auf Begriffe noch nicht alle Probleme löst. Gerade alltägliche Begriffe wie *Tasse* oder *Schuhe* werden auch durch Zusammenhänge zwischen Begriffen noch nicht hinreichend erklärt. Das Wesentliche einer Tasse ist weder ihre Form noch ihre Unterbegriffs-Beziehung zu Behältern, sondern daß wir daraus trinken. Natürlich kann man eine Relation *wird-benutzt-für* einführen, aber das wäre nur ein netter Name. Tatsächlich ist die Tätigkeit des Trinkens selbst das Entscheidende für die Feststellung, ob etwas eine Tasse ist oder nicht. Es sind also nicht nur Beziehungen zu anderen Begriffen, sondern auch zu Handlungen, die Alltagsbegriffe ausmachen.

1.3.3 Beiträge aus dem maschinellen Lernen

Das maschinelle Lernen ist zunächst dem klassischen Ansatz gefolgt und hat den Aggregationsschritt vorausgesetzt. So gibt es eine Fülle von Systemen, die aus vorgegebenen Beispielen für einen Begriff dessen Charakterisierung induzieren. Systeme zum *conceptual clustering* – obwohl auch meist ähnlichkeitsbasiert – beschreiben immerhin den Aggregationsschritt mit. Ein Oberbegriff wird zu einer Hierarchie von Begriffen verfeinert, wobei ähnliche Objekte zusammengruppiert werden. Beispiele sind UNIMEM [Lebowitz, 1987] und COBWEB [Fisher, 1987].

Die Notwendigkeit für komplexere Repräsentationen und das Einbeziehen von Hintergrundwissen wurde von Ryszard Michalski schon 1983 und von Yves Kodratoff und Jean-Gabriel Ganascia 1986 dargestellt [Michalski, 1983], [Kodratoff und Ganascia, 1986]. Insbesondere die logik-orientierten Ansätze beschäftigen sich mit der Generalisierung aus

eingeschränkt prädikatenlogischen Formeln [Morik, 1987][Muggleton und Buntine, 1988] [Rouveirol und Puget, 1990][Kietz und Wrobel, 1992][De Raedt, 1991].

Der Bedarf für einen Begriff wird bei CLUSTER/S durch einen Zielgraphen expliziert, der das *conceptual clustering* Verfahren steuert [Michalski und Stepp, 1986]. Stefan Wrobel [Wrobel, 1989] – wie auch schon [Emde et al., 1983] – beschreiben den Bedarf, der durch Ausnahmen einer ansonsten erfolgreichen Regel gegeben ist: sie sollen durch einen neuen Begriff zusammengefaßt werden, auf den eine zusätzliche Prämisse der Regel verweist. Das System KLUSTER beschreibt den Bedarf für einen neuen Begriff aufgrund der Unfähigkeit, mit den vorhandenen Begriffen eine Kategorie zu charakterisieren ([Morik und Kietz, 1989], [Kietz und Morik, 1994]).

Aktuelle Arbeiten versuchen, die Verankerung der Begriffsbildung in der Welt zu modellieren. So schlägt Stefan Wrobel einen kognitiv motivierten Forschungsrahmen vor, in dem strikt inkrementell gelernt wird [Wrobel, 1991]. Das heißt, die Eingabedaten stellen einen Strom von Informationen dar, der nicht vollständig gespeichert wird. Vielmehr werden die Daten nach und nach strukturiert und diese Strukturierung auf nachfolgende Eingaben angewandt. Revisionen können nicht anhand aller bereits gegebenen Daten überprüft werden. Lorenza Saitta und Attilio Giordana schlagen eine Begriffsstruktur vor, in der Merkmale Handlungen zur Verifizierung des Merkmals darstellen [Giordana und Saitta, 1990]. Die Verankerung von Bewegungs- begriffen wie *durch die Tür gehen* oder *eine Wand entlang gehen* wurde durch Lernen aus Bewegungs- und Wahrnehmungsdaten eines mobilen Roboters im europäischen Projekt BLearn untersucht [Morik und Rieger, 1993] [Klingspor et al., 1996]. Ein interdisziplinäres Forschungsprogramm der European Science Foundation, Learning in Humans and Machines(LHM), untersucht die Beziehungen zwischen didaktischen, kognitionspsychologischen und Arbeiten der KI. So wurden beispielsweise empirische Daten über kindliche Erklärungen des Tag/Nacht-Zyklus im System MOBAL modelliert. Mit diesem operationalen Modell konnten dann Experimente gemacht werden, deren Ergebnisse nun durch weitere empirische Untersuchungen validiert werden müssen [Mühlenbrock und Morik, 1998]. Es gibt bereits ein Buch über LHM [Reimann und Spada, 1996], eine Serie von weiteren Büchern soll 1998 erscheinen.

1.4 Deduktion, Induktion und Abduktion

Der deduktive Schluß ist bisher am längsten und gründlichsten untersucht worden. Im maschinellen Lernen steht der induktive Schluß im Vordergrund. Neuerdings wird auch der abduktive Schluß einbezogen. Wir betrachten im folgenden die drei Schlüsse der Reihe nach.

Beginnen wir mit dem **deduktiven Schluß**. Unter Deduktion versteht man die formale Schlußfolgerungsmethode in der Logik. Mit einem deduktiven Schluß leitet man aus wahren Aussagen andere wahre Aussagen ab. Deduktives Schließen kann durch Schlußregeln realisiert werden. Schlußregeln sollen **korrekt** sein, d. h. sie sollen aus Wahrem nichts Falsches ableiten. Weiterhin sollte eine Menge von Schlußregeln **vollständig** sein, d. h. mit ihr sollte man alle wahren Aussagen ableiten können. Zwei Schlußregeln werden uns im folgenden öfter begegnen:

$$\frac{A, A \rightarrow B}{B} \quad (ModusPonens) \qquad\qquad \frac{\forall x P(x)}{P(a)} \quad (Instantiierung)$$

Schlußregeln werden wie folgt angewendet: Wenn alle Aussagen oberhalb der Linie wahr sind, dann ist auch die Aussage unterhalb der Linie wahr.

Die wiederholte Anwendung dieser (und/oder anderer) Schlußregeln definiert den Begriff der Folgerung.

Definition 4: Deduktiver Schluß. *Aus einer Menge von Formeln B folgt (deduktiv) A (mit Bezug auf die Schlußregeln), gdw. es eine Folge von Schlußregelanwendungen gibt, die A ableitet. Wir schreiben dies als $B \vdash A$.*

Ein Beispiel für die Anwendung von sowohl der Instantiierungsregel als auch des Modus Ponens ist der folgende Schluß:

$$\forall x | sterblich(x) \leftarrow mensch(x), mensch(uta) \vdash sterblich(uta)$$

Umgekehrt geht man beim **induktiven Schluß** vor. Man schließt hier nicht vom Allgemeinen auf das Spezielle, sondern vom Speziellen auf das Allgemeine.

Definition 5: Induktiver Schluß. *Gegeben eine Menge D von Grundbeispielen der Form $c_{m+1} \leftarrow c_1, ..., c_m$. Die Hypothese H folgt induktiv aus D und dem Hintergrundwissen B, gdw.*

- $B \cup H \vdash D$,
- $B \nvdash D$ *und*
- $B \cup D \nvdash \neg H$

Wir schreiben dies als $D \cup B \prec H$.

Grundbeispiele sind Aussagen, die keine Variablen enthalten. Die Beispiele folgen erst dann aus der Theorie, wenn der allgemeine Satz H (die Hypothese) hinzugenommen wird, vorher nicht. Die Hypothese ist genereller als die Grundbeispiele, weil diese aus ihr ableitbar sind. Außerdem ist die Hypothese konsistent mit der Theorie und den Beispielen, d.h. aus Theorie und Beispielen folgt nicht die Negation der Hypothese. Betrachten wir zunächst ein Beispiel für einen induktiven Schluß, bei dem das Hintergrundwissen B leer ist.

Beispiel:
$D : sterblich(uta) \leftarrow mensch(uta), sterblich(udo) \leftarrow mensch(udo), sterblich(uwe) \leftarrow mensch(uwe) \prec$
$H : \forall x | sterblich(x) \leftarrow mensch(x)$

Jetzt nehmen wir noch eine Theorie als Hintergrundwissen B hinzu. Hier ist der induktive Schluß die Umkehrung der Instantiierungsregel.

Beispiel:
$B : \forall x | säugetier(x) \leftarrow mensch(x)$,
$D : sterblich(uta) \leftarrow mensch(uta), sterblich(udo) \leftarrow mensch(udo), sterblich(uwe) \leftarrow mensch(uwe) \prec$
$H : \forall x | sterblich(x) \leftarrow säugetier(x)$

Wie man leicht sieht, ist der induktive Schluß nicht wahrheitserhaltend.

Der **abduktive Schluß** schließlich wird sehr unterschiedlich aufgefaßt. Im einfachsten Falle ist er eine Umkehrung des Modus Ponens und dreht die Implikation um.

$$\frac{B, A \rightarrow B}{A} \qquad (Abduktion)$$

Definition 6: Abduktiver Schluß. *H folgt aus dem Hintergrundwissen B und Beobachtungen D abduktiv, gdw.*

$$B \cup H \vdash D$$

Wir schreiben dies als $B \cup D \succ H$.

Beispiel:
$B : \forall x | sterblich(x) \leftarrow mensch(x)$,
$D : sterblich(uta) \succ$
$H : mensch(uta)$

Auch der abduktive Schluß ist nicht wahrheitserhaltend. Die folgenden Abschnitte handeln überwiegend von dem induktiven Schluß.

1.5 Anwendungen maschinellen Lernens

Maschinelles Lernen wird überwiegend eingesetzt, um eine Menge von Regeln aus Daten zu gewinnen oder eine gegebene Regelmenge zu verbessern. Die Regeln werden dann entweder direkt von Menschen verwendet oder in ein System eingebettet und damit dessen Benutzern zur Verfügung gestellt. Einerseits spiegeln die Regeln (wenn wir ein zuverlässiges Lernverfahren verwenden) die Daten in verständlicher Form wieder. Sie ermöglichen es, die Daten zu inspizieren und einen Überblick zu bekommen (*Beschreibung*). Andererseits können die Regeln zur Lösung neuer Fälle genutzt werden (*Vorhersage*). Bei wissensbasierten Systemen (Expertensystemen) führt maschinelles Lernen zu einer erheblichen Verkürzung der Entwicklungszeit einer Wissensbasis, da viele Regeln nicht von Hand entwickelt und eingetragen werden müssen (*Wissenserwerb*).

Donald Michie berichtet über erfolgreiche Anwendungen von Lernverfahren, die Entscheidungsbäume induzieren [Michie, 1989]. Dabei muß das Lernergebnis nicht unbedingt von einem Expertensystem genutzt werden. Oft hilft bereits das Ausdrucken des Entscheidungsbaumes als Merkzettel. Die Erstellung des komprimierten Merkzettels ist die Leistung des Lernverfahrens. So führt Michie den Erfolg bei einer Anwendung für die NASA darauf zurück, daß Menschen selten mehr als 3–5 Faktoren auf einmal berücksichtigen können. Falls mehr als 5 Faktoren zu einer Entscheidung beitragen, ist ein induzierter Entscheidungsbaum, der auf der Grundlage aller vorliegender Daten und aller Faktoren gebildet wurde, hilfreich. Insofern hilft das statistisch basierte Lernen bei der Analyse von Daten, deren Ergebnis in eine verständliche, geordnete Form übertragen wird. Diese Analyseleistung war auch bei einem anderen von Michie angeführten Beispiel ausschlaggebend für den Erfolg. Im Bankenbereich der Kreditvergabe werden sicherheitshalber Kredite nicht vergeben, die in einer Grauzone liegen. Mit Hilfe eines Produktes, das auf ID3 beruht, konnten Daten über zurückgezahlte und nicht zurückgezahlte Kredite analysiert werden. Das Ergebnis strukturiert diese Grauzone, so daß mehr Kredite sicher vergeben werden können. Da das Ergebnis die Faktoren nennt, die ausschlaggebend für eine sichere Kreditvergabe sind, kann die Erwartung für das Kreditvolumen anhand statistischer Kenntnisse aktuell angepaßt werden. Ein Seiteneffekt war, daß der Bank bessere Kundenprofile für ihren Kundendienst zur Verfügung stehen. Schließlich konnte Michie von einer Firma eine schriftliche Bestätigung erhalten, daß die Produktivität einer Fabrik von 83% auf 95% gesteigert werden konnte durch den Einsatz induktiven Lernens[2].

[2] Dies entspricht einer Umsatzsteigerung von 10 000 US$ im Jahr.

Oft ist eine Induktionskomponente in eine Wissenserwerbsumgebung für eine Expertensystem-Hülle integriert. So wird zum Beispiel das System MOBAL[3] für unterschiedliche Anwendungen erprobt [Morik et al., 1993]. Hier zwei Beispiele:

Medizin: Ein medizinischer Sachbereich wurde einerseits mit Hilfe von benutzergegebenen Regeln dargestellt. Andererseits lernte das System typische Therapieabläufe aus im Krankenhaus gesammelten und von einem Arzt klassifizierten (und bereinigten) Daten. Mit Hilfe der Konsistenzprüfung von MOBAL wurden Abweichungen festgestellt, die dann analysiert wurden [Morik et al., 1994]. Die eigentliche Anwendung des Lernens bestand also darin, eine statistisch basierte und verständlich formulierte Übersicht über Therapieabläufe aus Daten zu erhalten.

Sicherheit in Telekommunikationsnetzen: Diese Anwendung des Systems MOBAL entspricht genauer dem klassischen Anwendungsbereich maschinellen Lernens: eine Wissensbasis zur Zugangsberechtigung von Benutzern zu bestimmten Rechnerleistungen wurde mit Hilfe des Systems erstellt. Dabei unterstützte das System verschiedene Aufgaben der Modellierung des Sachbereichs ([Sommer et al., 1994][Fargier, 1991]). Es ging darum, die Zugangsberechtigung unterschiedlicher Personen(-gruppen) anhand konkreter Fälle zu lernen und als Sicherheitskonzept in Form von Regeln zu formulieren.

Wenn in dem Aufbau und der Verfeinerung von Wissensbasen mit Hilfe maschinellen Lernens bisher auch die meisten Erfahrungen gesammelt wurden, so gibt es doch keinen prinzipiellen Grund, sich darauf zu beschränken. Vielmehr kann jedes System durch Lernfähigkeit verbessert werden. Die wichtigsten Anwendungsfelder sind gegenwärtig:

Lernen aus (Hyper-)texten: Das Lernen neuer Begriffe aus Texten wurde mit dem wit-System versucht [Reimer und Pohl, 1991]. Der WebWatcher unterstützt Benutzer des WWW beim *browsing* durch Lernen [Joachims et al., 1997]. Zur Unterstützung des *information retrieval* durch Lernen von Benutzerinteressen gibt es eine Fülle von Arbeiten (z.B.: [Lang, 1995][Balabanovic und Shoham, 1995][Pazzani et al., 1996][Lieberman, 1995]). Der Erwerb von Grammatiken aus Texten wurde früh untersucht, aber wegen der hohen Komplexität abgebrochen. Gegenwärtig wird die Annäherung an eine Grammatik unter Hinzuziehen eines Orakels (Benutzers) versucht [Adriaans et al., 1993].

Lernen in der Robotik: In der Robotik wird einerseits die Planungskomponente verbessert ([Dillmann, 1988][Segre, 1988][Zercher, 1991]), zum anderen die Ausführungskomponente [Kaelbling, 1991]. Es gibt aber auch Ansätze, gerade die Verbindung zwischen Begriffen der Planungsebene und den Sensor- und Handlungsdaten zu verbessern [Klingspor et al., 1996].

Wissensentdeckung in Datenbanken (data mining): Hier geht es darum, unübersichtliche Datensammlungen nach Regularitäten zu untersuchen oder sogar alle gültigen und interessanten Regeln zu finden. Dies wird bisher vor allem mit statistischen Methoden versucht (Stichwort: explorative Datenanalyse). Maschinelle Lernverfahren, die meist einen statistischen Kern enthalten, gehen in der Aufbereitung ihrer Ergebnisse über rein statistische Verfahren hinaus, indem sie verständliche Regeln

[3] MOBAL wurde an der Gesellschaft für Mathematik und Datenverarbeitung in St. Augustin entwickelt.

ausgeben. Außerdem werden die Hypothesen für gültige Regeln vom System selbst aufgestellt und nicht vom Benutzer formuliert. Ein schnelles Verfahren für binäre Attribute wie sie in Warenhausdaten vorkommen (jede Ware ist ein Attribut, 1 heißt, daß sie gekauft wurde, ein Datenbanktupel ist ein Einkauf) ist APRIORI [Agrawal et al., 1996]. Ein prädikatenlogisches Verfahren zum Regellernen mit direktem Datenbankzugriff wird in der Vorlesung besprochen (Abschnitt 5.5.1).

2 Lernen als Suche

Tom Mitchell hat Lernen aus Beispielen als Suche beschrieben [Mitchell, 1982]. Beispiele sind in der Lernliteratur einer Kategorie zugeordnete (klassifizierte) Aussagen. Im Gegensatz dazu sind Beobachtungen nicht klassifiziert. Beim Lernen aus Beispielen wurde also die Aggregation bereits vom Benutzer oder einem anderen System vorgenommen. Beim Lernen aus Beobachtungen gehört die Aggregation zur Lernaufgabe. Die Aufgabe, aus Beispielen zu lernen, ist:

Definition 7: Lernen als Suche. *Sei das folgende gegeben:*

- *Beschreibungssprache LE für Beispiele*
- *Definitionssprache LH für den Begriff*
- *Menge P positiver Beispiele (Beispiele für den Begriff)*
- *Menge N negativer Beispiele (Nicht-Beispiele für den Begriff)*
- *Abgleichsprädikat, das Beispiele klassifiziert (covers)*

Lernen wird als die Suche in LH nach einer Hypothese $c \in LH$ aufgefaßt, die folgende Bedingungen erfüllt:

- $\forall p \in P$, $covers(c, p)$ *ist wahr*
- $\forall n \in N$, $covers(c, n)$ *ist falsch*

Der Such- bzw. Hypothesenraum für Begriffe ist die Menge aller mit Hilfe von LH bildbaren Ausdrücke. Das sind alle möglichen Charakterisierungen, für die dann festgestellt werden muß, ob sie alle positiven Beispiele abdecken und kein negatives. Das Abgleichsprädikat kann etwa durch die logische Folgerung realisiert werden: $covers(c, e)$ gdw. $c \models e$. Der einfachste Lernalgorithmus ist der **Aufzählungsalgorithmus**: er zählt alle in LH bildbaren Ausdrücke auf (Hypothesengenerierung) und prüft für jeden, welche Beispiele (und Nicht-Beispiele) abgedeckt werden (Hypothesentest). Sobald die Zielbedingung gilt, hält der Algorithmus an.

Der Aufzählungsalgorithmus funktioniert natürlich nur für aufzählbare Sprachen und ist nicht gerade effizient. Ein übliches Verfahren, einen Algorithmus, der Hypothesen generiert und testet, effizienter zu machen, besteht darin, Bedingungen des Testens bereits bei der Generierung zu berücksichtigen. Im Falle des induktiven Lernens wissen wir, daß wir eine Hypothese suchen, die genereller ist als die Beispiele und spezieller als eine Aussage, die sowohl positive als auch negative Beispiele abdeckt. Wir tun also gut daran, die Hypothesen nach ihrer Allgmeinheit anzuordnen, um dann schrittweise generellere oder speziellere Hypothesen zu generieren. Gehen wir von den Beispielen aus, um schrittweise

generellere Hypothesen zu erzeugen bis alle positiven Beispiele abgedeckt werden, spricht man von einem *bottom-up* Verfahren. Gehen wir von einer alles abdeckenden Aussage aus, die wir schrittweise spezialisieren bis sie kein negatives Beispiel mehr abdeckt, spricht man von einem *top-down* Verfahren. Die Suche in einem strukturierten Raum möglicher Hypothesen kann beschnitten werden, was bei der Suche im unstrukturierten Raum nicht möglich ist. Dort müssen ja alle Hypothesen betrachtet werden, weil man keinen Anhaltspunkt hat, wo im Hypothesenraum der Zielbegriff liegen könnte.

Tom Mitchell schlug für Definitionssprachen eine Halbordnung (*quasi-ordering*) aufgrund der spezieller-als- bzw. genereller-als-Relation vor. Wenn Begriffe mit Hilfe von Attributwerten charakterisiert werden, die sich in einer Hierarchie entlang dieser Relation partiell anordnen lassen, läßt sich der Suchraum als Kreuzprodukt der geordneten Attributwerte immer (halb–) ordnen.

Definition 8: Spezieller-als Relation. *$c1$ ist spezieller als $c2$ genau dann, wenn*

$$\forall e \in LE gilt : covers(c1, e) \rightarrow covers(c2, e),$$

Lemma 9: *$c1$ ist spezieller als $c2$ genau dann, wenn*

$$\{e \in LE \mid covers(c1, e)\} \subseteq \{e \in LE \mid covers(c2, e)\}$$

Anders herum: $c2$ ist eine Generalisierung von $c1$, weil $c2$ alle Beispiele abdeckt, die $c1$ auch abdeckt, und zusätzlich vielleicht noch mehr Beispiele. Mit dieser Angabe kann entschieden werden, ob eine Hypothese genereller oder spezieller als eine andere ist. Dies reicht aber noch nicht aus. Wenn wir schrittweise generalisieren bzw. spezialisieren wollen, müssen wir minimal generellere und minimal speziellere Hypothesen zu einer Hypothese finden.

Definition 10: Speziellste Generalisierung. *Seien $g, g', c \in LH, e \in LE$ und $\neg covers(c, e)$, dann ist g die speziellste Generalisierung von c in Bezug auf e genau dann, wenn*

- *c ist spezieller als g und $g \neq c$*
- *$covers(g, e)$ und*
- *$\neg\exists g' \in LH$ mit $g' \neq g$, so daß g genereller als g' ist und $covers(g', e)$.*

Es wird also g durch Generalisierung um einen Schritt erzeugt: zwischen sie und die bisherige Generalisierung c paßt keine andere Generalisierung g' mehr. Dies ist insbesondere sinnvoll, wenn e ein positives Beispiel ist, das abgedeckt sein soll.

Entsprechend läßt sich auch die Spezialisierung formalisieren.

Definition 11: Generellste Spezialisierung. *Seien $s, s', c \in LH, e \in LE$ und $covers(c, e)$, dann ist s die generellste Spezialisierung von c in Bezug auf e genau dann, wenn*

- *s ist spezieller als c und $s \neq c$*
- *$\neg covers(s, e)$ und*
- *$\neg\exists s' \in LH$ mit $s' \neq s$, so daß s spezieller ist als s' ist und $\neg covers(s', e)$.*

Es wird also s durch Spezialisierung um einen Schritt erzeugt. Dies ist insbesondere sinnvoll, wenn e ein negatives Beispiel ist, das nicht abgedeckt sein soll.

Jetzt lassen sich drei Lernalgorithmen angeben, die alle die Lernaufgabe wie oben angeführt lösen. Es wird dabei angenommen, daß die allgemeinsten Begriffe aus LH alle Beispiele abdecken, die speziellsten alle Beispiele ausschließen. Oft gibt es mehrere Lösungen für ein Lernproblem. Die angeführten Verfahren halten nach der ersten Lösung an. Alternativ könnte auch ein Präferenzkriterium die beste aus allen Lösungen heraussuchen.

Top-down Lernverfahren: Beginne mit den allgemeinsten bildbaren Hypothesen;
solange noch negative Beispiele abgedeckt werden, wende auf die Hypothese das schrittweise Spezialisieren an;
wenn eine Hypothese kein negatives Beispiel abdeckt, gib diese Hypothese aus und halte an.

Bottom-up Lernverfahren: Beginne mit den speziellsten bildbaren Hypothesen;
solange noch nicht alle positiven Beispiele abgedeckt werden,
wende das schrittweise Generalisieren an;
wenn eine Hypothese alle positiven Beispiele abdeckt, gib diese Hypothese aus und halte an.

Tom Mitchell führte zusätzlich zu diesen beiden Algorithmen die bi-direktionale Suche im Versionenraum (*versions space*) ein, die Spezialisierung und Generalisierung kombiniert [Mitchell, 1982]. Es werden gleichzeitig zwei Mengen bearbeitet: die Menge aller aktuellen Generalisierungen und die Menge aller aktuellen Spezialisierungen. Jedes Element dieser Mengen ist möglicherweise die gesuchte Hypothese. Die Mengen enthalten also alternative Hypothesen. Der Versionenraum beinhaltet die Hypothesen der aktuellen Generalisierungen, die Hypothesen der aktuellen Spezialisierungen und die Hypothesen "dazwischen". Sobald sich die beiden Mengen überschneiden, liegt die Lösung in der Schnittmenge. Weitere Beispiele können beide Mengen ganz zur Deckung bringen. Dann wurde die Lösung gefunden.

Algorithmus 12: Versionenraum Lernverfahren. Initialisiere die Menge G mit den generellsten Begriffen und die Menge S mit den speziellsten.
Solange die Mengen G und S disjunkt sind, lies ein Beispiel e ein und

falls $e \in N$ und e von G abgedeckt wird,
entferne all $s \in S$, die e abdecken,
spezialisiere G bis e nicht mehr abgedeckt wird,
entferne alle $g \in G$, die echt spezieller sind als ein anderes $g' \in G$

falls $e \in P$ und e von S nicht abgedeckt wird,
entferne als $g \in G$, die g nicht abdecken,
generalisiere S bis e abgedeckt wird,
entferne alle $s \in S$, die echt allgemeiner sind als ein anderes $s' \in S$.

Dann entferne alle $g \in G$, für die es kein $s \in S$ gibt, das spezieller ist,
entferne alls $d \in S$, für die es kein $g \in G$ gibt, das genereller ist.

Sobald G und S gleich sind und nur noch eine Hypothese enthalten,
gib diese aus und halte an.

Dabei sind die e_i allgemein Beispiele, p_i und n_i sind die bisher dem System gezeigten positiven und negativen Beispiele. Die Mengen G und S sind also folgendermaßen definiert:

Definition 13: Menge der generellsten Spezialisierungen. *Die Menge G der generellsten Spezialisierungen ist definiert als*

G=$\{g$, *so daß* $\forall p_i \in P, \forall n_i \in N$ *gilt*
$covers(g, p_i)$, $\neg covers(g, n_i)$,
$\neg\exists g'$, *das genereller ist als* g *und alle* p_i *abdeckt und kein* $n_i\}$

Definition 14: Menge der speziellsten Generalsisierungen. *Die Menge S der speziellsten Generalisierungen ist definiert als*

S=$\{s$, *so daß* $\forall p_i \in P, \forall n_i \in N$ *gilt*
$covers(s, p_i)$, $\neg covers(s, n_i)$,
$\neg\exists s'$, *das spezieller ist als* s *und alle* p_i *abdeckt und kein* $n_i\}$

Definition 15: Vollständigkeit einer Charakterisierung. *Eine Charakterisierung c ist vollständig, wenn sie alle positiven Beispiele abdeckt, also gilt:*

$$\forall p_i \in P : covers(c, p_i).$$

Definition 16: Korrektheit einer Charakterisierung. *Eine Charakterisierung c ist korrekt, wenn sie keine negativen Beispiele abdeckt, also gilt:*

$$\forall n_i \in N : \neg covers(c, n_i).$$

Ein Beispiel soll die Strukturierung des Suchraums illustrieren. Nehmen wir an, LH enthielte zwei Merkmale mit hierarchisch angeordneten Werten, wobei die allgemeineren Werte oben, die spezielleren unten stehen.

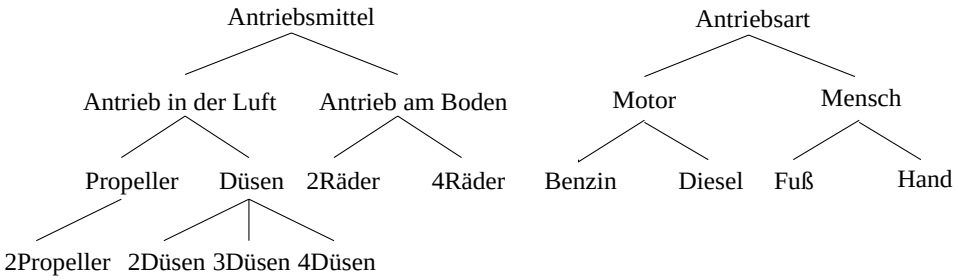


Abbildung1. Merkmalsbäume

Die Blätter der Merkmalsbäume werden verwendet, um Beispiele, die darüber gelegenen Merkmale, um Charakterisierungen anzugeben. Beispiele sehen dann so aus:

P: $\{$ $[2Räder, Benzin]$, $[4Räder, Diesel]$, $[2Räder, Fuß]\}$

N: $\{$ $[3Düsen, Hand]$ $\}$

Der geordnete Suchraum stellt alle Kombinationen der beiden Merkmale in der Anordnung von generelleren Charakterisierungen (oben) zu spezielleren (unten) dar. Das Abgleichsprädikat *covers* für die zwei Merkmale verhält sich folgendermaßen:

$covers([a,b],[c,d])$ gdw. $covers(a,c)$ und $covers(b,d)$.

Dabei sind die Beschreibungen a und b für das Beispiel und c und d für den entstehenden Begriff. Im Suchraum können verschiedene generellere Beschreibungen dieselbe Spezialisierung haben. Da die beiden Merkmalsbäume unterschiedlich tief sind, liegen nicht alle Beispielbeschreibungen (Blätter) auf derselben Ebene des Suchraums. Ein Ausschnitt:

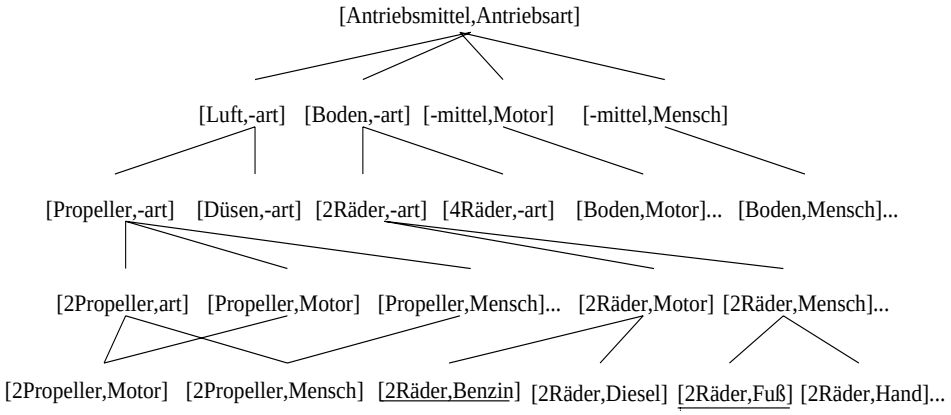


Abbildung2. Ausschnitt aus dem Versionenraum

Ein Versionenraum-Programm verhält sich etwa so:

Programm: Was ist ein positives Beispiel?

Benutzer: $[2Räder, Benzin]$

Programm: G: $[Antriebsmittel, Antriebsart]$ S: $[2Räder, Benzin]$
Ein weiteres Beispiel?

Benutzer: $[3Düsen, Hand]$ n

Programm: G: $[Boden, Antriebsart]$, $[Antriebsmittel, Motor]$ S: $[2Räder, Benzin]$
Ein weiteres Beispiel?

Benutzer: $[4Räder, Diesel]$ p

Programm: G: $[Boden, Antriebsart]$, $[Antriebsmittel, Motor]$ S: $[Boden, Motor]$
Ein weiteres Beispiel?

Benutzer: $[2Räder, Fuß]$ p

Programm: G: $[Boden, Antriebsart]$ S: $[Boden, Antriebsart]$
LÖSUNG: $[Boden, Antriebsart]$

In diesem Beispiel kann man das Abgleichsprädikat durch die Vorgängerrelation zwischen Knoten im Merkmalsbaum definieren. Der strukturierte Suchraum entsteht dann beim Abgleichen. Obendrein muß das schrittweise Generalisieren und Spezialisieren formuliert werden sowie der globale Ablauf. In Prolog läßt sich das leicht machen.

3 Lernen als Funktionsapproximation

Viele Lernprobleme lassen sich als Funktionsapproximation auffassen (z.B. auch das Begriffslernen). Das Ziel ist es, eine Hypothese zu finden, die zur Vorhersage von zukünftigen Ereignissen benutzt werden kann. Gegeben sind Beispiele von der Funktion, die gelernt werden soll. Gesucht ist die Hypothese, die diese Funktion möglichst gut approximiert. Eine Hypothese approximiert die Zielfunktion genau dann gut, wenn ihre Vorhersagen möglichst häufig tatsächlich eintreten. Ein Beispiel hierfür ist Kreditwürdigkeitsprüfung. Das Lernproblem ist hier, anhand von z. B. der Kreditgeschichte, Kontostand etc. vorherzusagen, ob der Bankkunde einen Kredit ordnungsgemäß zurückzahlen wird oder nicht. Beispiele sind die Erfahrungen mit anderen Kunden (positive und negative). Gesucht ist eine Hypothese, welche die Zahlungsmoral eines neuen Kunden möglichst genau vorhersagen kann (d. h. sich möglichst selten irrt). Dieses Modell läßt sich formal wie folgt aufschreiben:

Definition 17: Lernen als Funktionsapproximation. *Sei das folgende gegeben:*

- *Ein Generator G, der Beispielbeschreibungen $x_i \in E$ anhand einer Wahrscheinlichkeitsverteilung $P(x_i)$ erzeugt.*
- *Ein Orakel O, das jeder von G erzeugten Beispielbeschreibung x_i einen Wert $y_i = t(x_i)$ zuweist.*
- *Eine Hypothesensprache LH.*

Ziel des Lernens als Funktionsapproximation ist es, die Hypothese $h \in LH$ zu finden, welche den folgenden Ausdruck (genannt den zu erwartenden Fehler oder auch das zu erwartende Risiko) minimiert:

$$R(h) = \sum_{i=1}^{|E|} Q(x_i, h) \cdot P(x_i)$$

$P(x_i)$ ist die Wahrscheinlichkeit, daß das Beipiel x_i aus der Beispielbeschreibungsprache gezogen wird. Es ist also wichtig, auf wahrscheinlich auftretenden Beipielen x_i weniger Fehler zu machen als auf unwahrscheinlichen Beispielen. $Q(x_i, h)$ ist eine Fehlerfunktion (sog. *Loss-Function*). Sie beschreibt die Qualität der Vorhersage von Hypothese h für Beispiel x_i. Anhand der Form von Q unterscheidet man u. a. die folgenden zwei Aufgaben:

Klassifikation: Einteilung von Beispielen in eine feste und vorgegebene Anzahl von Klassen (z. B. Klassifikation von Bankkunden in die Klassen *kreditwürdig* und *nicht kreditwürdig*). Hier wird normalerweise die folgende Fehlerfunktion verwendet. Sie liefert den Wert 1, wenn die Vorhersage $h(x_i)$ falsch ist.

$$Q(x_i, h) = \begin{cases} 1 \; h(x_i) \neq t(x_i) \\ 0 \; h(x_i) = t(x_i) \end{cases}$$

Regression: Approximation einer reellwertigen Funktion (z. B. Vorhersage von Aktienkursen). Häufig ist Q hier die quadrierte Abweichung des vorhergesagten Wertes $h(x)$ vom Sollwert $t(x)$.

$$Q(x_i, h) = (t(x_i) \Leftrightarrow h(x_i))^2$$

Die direkte Minimierung des zu erwartenden Fehlers $R(h)$ ist nicht möglich, da wir weder $P(x_i)$ noch $t(x_i)$ für alle i kennen. Allerdings haben wir Beispiele, die vom Generator anhand von $P(x_i)$ gezogen wurden und für die wir $t(x_i)$ kennen. Diese Beispiele werden dazu benutzt, den zu erwartenden Fehler $R(h)$ mit dem beobachteten Fehler $R_{emp}(h)$ zu approximieren.

Definition 18: Beobachteter Fehler. *Der beobachtete Fehler für eine Menge von Beispielen* $(x_1, t(x_1)), ..., (x_n, t(x_n))$ *und eine Hypothese* h *berechnet sich als:*

$$R_{emp}(h) = \frac{1}{n} \cdot \sum_{i=1}^{n} Q(x_i, h)$$

Daraus läßt sich das folgende Lernproblem formulieren. Diese Methode des Lernens wird *empirische Risikominimierung* (ERM) genannt.

Definition 19: Empirische Risikominimierung (ERM). *Sei das folgende gegeben:*

- *Eine Menge von Beispielen* $(x_1, t(x_1)), ..., (x_n, t(x_n))$ *und*
- *eine Hypothesensprache LH*

Man spricht von empirischer Risikominimierung (ERM), wenn ein Lernalgorithmus die Hypothese $h \in LH$ *sucht (und findet), für die der beobachtete Fehler* $R_{emp}(h)$ *minimal ist.*

Mit dieser Methode finden wir also die Hypothese h mit dem kleinsten empirischen Fehler $R_{emp}(h)$ auf den Beispielen. Wie groß ist aber der tatsächliche Fehler $R(h)$ dieser Hypothese? Es stellt sich heraus, das $R_{emp}(h)$ in der Regel kleiner ist als $R(h)$, d.h. obwohl h auf den Beispielen nur wenige Fehler gemacht hat, wird die Performanz auf ungesehenen Beispielen schlechter sein. Die Theorie des PAC-Lernens (s. Abschnitt 4) gibt an, wie groß die Differenz zwischen $R_{emp}(h)$ und $R(h)$ ist. Sie hängt im wesentlichen von der Größe des Hypothesenraumes und der Anzahl der Trainingsbeispiele ab. Eine Lernprinzip, das auf der Theorie des PAC-Lernens aufbaut, ist die *strukturelle Risikominimierung* [Vapnik, 1995]. Im Gegensatz zur empirischen Risikominimierung wägt man hierbei die Größe des Hypothesenraumes gegen die Anzahl der Beispiele und den empirischen Fehler $R_{emp}(h)$ ab.

Meist wird anhand einer ausgewählten Teilmenge von klassifizierten Daten (Lernmenge) eine Menge von Regeln oder ein Entscheidungsbaum induziert. Das Lernergebnis wird dann anhand einer anderen Teilmenge der klassifizierten Daten (Testmenge) geprüft. Dabei wird die Testmenge ohne die vorgegebene Klassifikation mit dem Lernergebnis klassifiziert. Wenn die Klassifikation durch das Lernergebnis mit der benutzergegebenen Klassifikation übereinstimmt, ist es korrekt. Damit nicht durch die besonders unglückliche Aufteilung in Lern- und Testmenge das Ergebnis verfälscht wird, werden oft n Lernläufe mit verschiedenen Lern- und Testmengen durchgeführt und als Korrektheit des Lernergebnisses der Mittelwert der Ergebnisse auf den Testbeispielen angegeben.

Algorithmus 20: Cross Validation. Man teile alle verfügbaren Beispiele in n Mengen auf. Für i=1 bis i=n:

- Wähle die i-te Menge als Testmenge und die restlichen $n \Leftrightarrow 1$ Mengen als Lernmenge.
- Bestimme die Korrektheit des Lernergebnisses auf der Testmenge.

Bilde das Mittel der Testergebnisse aller n Lernläufe. Dies gibt die Qualität des Lernergebnisses an.

3.1 Lernen in neuronalen Netzen

Wir können das Lernen in neuronalen Netzen als empirische Risikominimierung auffassen. Die Lernaufgabe ist:

Gegeben: Eine Menge von Beispielen in einer Attribut-Wert-Repräsentation mit binären Attributen.
Die Struktur des neuronalen Netzes.

Ziel: Ein neuronales Netz, das für neue Beispiele die Zielfunktion mit möglichst geringem Fehler vorhersagt.

Der im weiteren vorgestellte Backprop-Algorithmus ist in der Lage, sowohl Klassifikations- als auch Regressionsprobleme zu bearbeiten. Der Algorithmus stellt nur eine Methode aus dem Bereich der neuronalen Netze dar und wird hier exemplarisch behandelt. Im folgenden werden neuronale Netze aus der Sicht des maschinellen Lernens behandelt und nicht versucht, biologische Vorgänge im Gehirn zu modellieren. Es wird allgemein bezweifelt, daß im Gehirn Prozesse ablaufen, die mit dem Backprop-Algorithmus vergleichbar sind.

3.1.1 Neuronale Netze

Neuronale Netze setzen sich aus Neuronen zusammen. Jedes Neuron berechnet eine relativ einfache Funktion und erst ihr Zusammenwirken erlaubt die Repräsentation von komplexen Begriffen. Der Backprop-Algorithmus benutzt Neuronen des in Abb. 3 dargestellten Typs (genannt *sigmoides Neuron*).

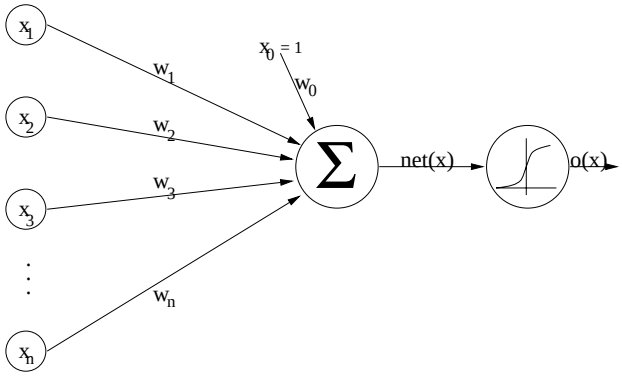


Abbildung3. Darstellung eines sigmoiden Neurons

Ein Neuron besitzt einen Vektor von aktuellen Eingabewerten $\vec{x} = (x_1, \ldots, x_n)$ mit $x_i \in \Re$ (x_0 ist immer gleich 1 und realisiert einen Schwellwert). Für jede Dimension i dieses Vektors existiert ein Gewichtungsfaktor w_i , der in der Lernphase durch den Algorithmus verändert werden kann. Aus den Eingabewerten und den Gewichten berechnet das Neuron die Summe

$$net(\vec{x}) = \sum_{i=0}^{n} w_i \cdot x_i$$

Um die Ausgabe $o(x_i)$ des Neurons für den Eingabevektor $\vec{x}$ zu berechnen, wird die Summe $net(\vec{x})$ durch eine sigmoide Funktion $\sigma(x)$ auf den Wertebereich von 0 bis 1 normiert. Man wählt zur Normierung eine sigmoide Funktion, da diese differenzierbar ist und ihre erste Ableitung beim Lernen benutzt werden kann, um die Gewichte angemessen zu verändern. Durch die sigmoide Form verhält sich das Neuron aber weitestgehend wie ein Neuron mit der Vorzeichenfunktion $sign(x)$, die den Wert 0 annimmt, wenn $x < 0$ und 1 wird, wenn $x > 0$.

Eine nichtlineare Ausgabefunktion (wie die Vorzeichen- oder eine sigmoide Funktion) ist wichtig, da sie beim Zusammenschluß von mehreren Neuronen zu neuronalen Netzen die Ausdruckskraft des Netzes erhöht. Wählt man eine lineare Ausgabefunktion, kann das Netz auch nur lineare Funktionen lernen.

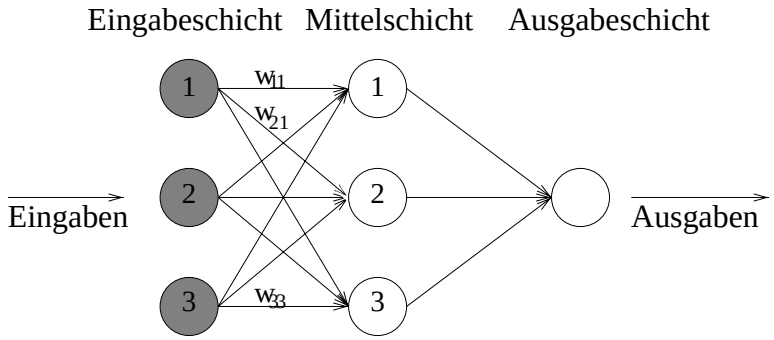


Abbildung4. Dreilagiges Netz in Feedforward-Architektur

Sehr häufig werden Netze der *feedforward* Struktur (Abb. 4) verwendet. Mit genügender Anzahl von (sigmoiden) Neuronen in der Mittel-Schicht, können Netze dieser Struktur eine große Klasse von Funktionen beliebig genau approximieren (z. B. beliebige Boolesche Funktionen und beschränkte stetige Funktionen). Das Netz besteht aus drei Schichten: Einer Eingabe-Schicht, einer Mittel- Schicht (auch *hidden layer* genannt) und einer Ausgabe-Schicht. Die Ausgaben einer Schicht sind die Eingaben zur jeweils nachgelagerten Schicht. Jedes Neuron der nachgelagerten Schicht ist mit jedem Ausgang der vorgelagerten Schicht verbunden. An der Eingabe-Schicht wird die Beschreibung eines Beispiels in das Netz eingegeben. In dieser Schicht befinden sich keine eigentlichen Neuronen, sondern sie dient nur als Schnittstelle für die Eingaben. Die Anzahl der Neuronen in der Mittel-Schicht ist frei wählbar und bestimmt die Komplexität der Funktionen, die das Netz lernen kann. Die Ausgabe der Ausgabeschicht ist die Klassifikation des Beispiels. Vielfach besteht die Ausgabeschicht auch aus mehreren Neuronen und gibt somit einen Vektor aus.

Man unterscheidet zwei Durchlaufrichtungen durch das Netz: vorwärtsgerichtet und rückwärtsgerichtet. Beim **Vorwärtsdurchlauf** werden ausgehend von den Eingaben der Eingabeschicht die Neuronen der Mittelschicht berechnet und danach die der Ausgabeschicht. Die Ausgabe ist eine Zahl im Bereich $[0 \ldots 1]$ und kann durch einen Schwellwert in eine binäre Ausgabe umgewandelt werden. Auf diese Weise berechnet das Netz seine Vorhersage für die Klassifikation eines Beispiels.

Der **Rückwärtsdurchlauf** wird in der Lernphase verwendet. Lernen bedeutet, daß die Gewichte der Neuronen mit Hilfe von Trainingsbeispielen angepaßt werden. Anhand einer Menge von Beispielen E werden die Gewichte der Neuronen so eingestellt, daß die

folgende Fehlerfunktion F minimiert wird.

$$F = \frac{1}{2} \sum_{e \in E} (t(e) \Leftrightarrow o(e))^2 \tag{1}$$

$t(e)$ ist die gegebene Klassifikation des Beispiels e und $o(e)$ ist die Ausgabe des Netzes (berechnete Klassifikation). Wir können F für eine Lernmenge berechnen, wenn wir alle Gewichte w_{ji} festgelegt haben (Hypothesentest). Eine Hypothese ist hier also die Menge aller Werte für alle Gewichte w_{ji}. Wie generieren wir solche Hypothesen in einer sinnvollen Reihenfolge? Woher können wir wissen, wann wir die Suche im Hypothesenraum abbrechen dürfen? Eine Ordnung nach Allgemeinheit läßt sich hier nicht angeben. Die Rolle, die diese Allgemeinheitsordnung bei dem Lernen im Versionenraum spielte, war, dem Verfahren eine Richtung anzugeben, in die die Hypothese verändert werden soll (generalisieren oder spezialisieren). Entscheidend war dabei, daß bei jedem Beispiel und gegebenen Hypothesen *lokal* über die Richtung entschieden werden konnte. Auch hier wollen wir bei einer Beispielmenge und jedem Gewicht zumindest entscheiden können, ob wir das Gewicht größer oder kleiner machen sollen. Die erste Ableitung bezüglich einer Beispielmenge E und eines Gewichts w_{ji}, d.h. der Gradient $\frac{\partial F_E}{\partial w_{ji}}$, liefert uns genau diese Richtungsangabe.

3.1.2 Gradientenabstieg und Deltaregel

Das Prinzip des **Gradientenabstiegs** ist bei nur einem Neuron leicht zu verstehen. Deshalb wird hier der Zusammenhang von Gradienten und Gewichtsveränderung für ein Neuron hergeleitet, bevor wir das Prinzip auf die Gewichtsveränderungen bei mehrschichtigen neuronalen Netzen anwenden. Ein Neuron erhält gewichtete Eingabewerte. Uns interessieren die Gewichte. Eine Kombination von Werten aller Gewichte schreiben wir als einen Vektor $\vec{w}$. Den Gradienten des Fehlers bezüglich $\vec{w}$ schreiben wir $\bigtriangledown F$ und definieren ihn als

$$\bigtriangledown F(\vec{w}) = \left[\frac{\partial F}{\partial w_0}, \frac{\partial F}{\partial w_1}, \ldots, \frac{\partial F}{\partial w_n} \right] \tag{2}$$

Dieser Gradient gibt die Richtung an, in der der Fehler wächst. Also gibt die Negation die Richtung an, in der der Fehler abnimmt. Wir geben auch noch eine Schrittweite η an, mit der die Gewichte verändert werden sollen. Dies ergibt den Term, um den ein bestimmtes Gewicht verändert werden soll:

$$\vec{w} := \vec{w} + \Delta\vec{w} \tag{3}$$

$$\Delta\vec{w} = \Leftrightarrow\eta \bigtriangledown F(\vec{w}) \tag{4}$$

Wir können dies analog zu 2 auch für jedes einzelne Gewicht schreiben:

$$w_i := w_i + \Delta w_i, \tag{5}$$

$$\Delta w_i = \Leftrightarrow\eta \frac{\partial F}{\partial w_i} \tag{6}$$

Jetzt müssen wir nur noch Δw_i in jedem Schritt berechnen. Wenn wir nun in Gleichung 6 für F den quadratischen Fehler (wie in 1 definiert) einsetzen und geeignet umformen, erhalten wir

$$\frac{\partial F}{\partial w_i} = \sum_{e \in E} (t(e) \Leftrightarrow o(e)) \cdot (\Leftrightarrow x_i) \tag{7}$$

Dabei ist x_i der $i$te Wert des Eingabevektors aus Beispiel e. Dies ergibt, wenn wir es in 6 einsetzen die **Deltaregel**:

$$\Delta w_i = \eta \sum_{e \in E} (t(e) \Leftrightarrow o(e)) x_i \tag{8}$$

Die Deltaregel berechnet den Fehler über allen Beispielen der Lernmenge und verändert dann die Gewichte, berechnet wieder den Fehler über allen Beispielen, verändert die Gewichte, ... bis der Fehler klein genug ist. Die inkrementelle Version der Deltaregel verändert die Gewichte nach jedem Beispiel:

$$\Delta w_i = \eta (t \Leftrightarrow o) x_i \tag{9}$$

Die Deltaregel verwendet den Fehler einer Hypothese, um die nächste Hypothese zu generieren. Ein Neuron lernt mit dieser Regel in der folgenden Weise. Gegeben sind die Beispiele als Paare $(\vec{x}, t)$, wobei $\vec{x}$ der Vektor der numerischen Eingabewerte und t der tatsächliche Ausgabewert ist. Die Gewichte w_i werden mit irgendwelchen kleinen Werten initialisiert, die Lernrate η ebenso.

Algorithmus 21: Lernen mit der Deltaregel. Für alle Beispiele, bis die Haltebedingung erfüllt ist:

- Berechne für die Eingabe $\vec{x}$ den Ausgabewert o.
- Für jedes Gewicht w_i: $w_i := w_i + \eta (t \Leftrightarrow o) x_i$

3.1.3 Backprop

Das Verfahren, das wir eben für nur ein Neuron kennengelernt haben, läßt sich leicht erweitern für ein neuronales Netz mit mehreren Schichten. Es müssen jetzt lediglich die Schichten gesondert behandelt werden. Die Gewichte, die zur Ausgabeschicht führen, werden fast so wie eben angepaßt. Wir müssen nur beachten, daß es jetzt einen Vektor als Ausgabe gibt und nicht nur einen Wert. Bei der Mittelschicht gibt es das Problem, daß es keine vorgegebenen Werte gibt, mit denen die berechneten Werte verglichen werden können. Allerdings wissen wir, zu welchem Fehler in der Ausgabeschicht ein Neuron der Mittelschicht beigetragen hat. Und das Gewicht, das von diesem Mittelschichtneuron zu dem mit einem bestimmten Fehler behafteten Ausgabeneuron führt, kennen wir auch. Es gibt das Ausmaß an, in dem das Mittelschichtneuron am Fehler des Ausgabeneurons *schuld* hat. Wenn also die Verbindung von Neuron h zu Ausgabeneuronen k gewichtet ist mit w_{kh}, so propagieren wir den mit w_{kh} gewichteten Fehler von k an h. Dies tun wir für alle Verbindungen zwischen Neuronen der Mittelschicht und der Ausgabeschicht. Damit haben wir die Fehler aller Mittelschichtneuronen und können sie an weitere Schichten in derselben Weise propagieren, bis die Eingabeschicht erreicht ist, für die keine Fehler bestimmt werden müssen.

Der folgende Algorithmus beschreibt die Lernphase eines Netzes mit einer Mittelschicht und die damit verbundene Veränderung der Gewichte. Es ist klar, wie der Algorithmus für weitere Mittelschichten zu schreiben ist – es wird dann lediglich unübersichtlicher, nicht schwieriger. Gegeben sind die Beispiele als Paare $(\vec{x}, \vec{t})$, wobei $\vec{x}$ der Vektor der numerischen Eingabewerte und $\vec{t}$ der Vektor der tatsächlichen Ausgabewerte ist. Die Eingabe, die von Neuron i an Neuron j gegeben wird, wird x_{ji} geschrieben. Das Gewicht der Verbindung von Neuron i nach Neuron$j$ wird w_{ji} geschrieben. Den Fehler eines Neurons i notieren wir δ_i. Die Gewichte w_{ji} werden mit irgendwelchen kleinen Werten – etwa zwischen $\Leftrightarrow 0,05$ und $0,05$ – initialisiert, die Lernrate η ebenso.

Algorithmus 22: Backprop. Bis der Fehler *klein genug* ist, wiederhole für jedes Beispiel e:

- Vorwärtsdurchlauf: Berechne $o(e)$ mit einem Vorwärtsdurchlauf.
- Rückwärtsdurchlauf: Verändere die Gewichte in einem Rückwärtsdurchlauf:
  - Für alle Neuronen $k \in Ausgabeschicht$:
    $\delta_k := o_k(e) \cdot (1 \Leftrightarrow o_k(e)) \cdot (t_k(e) \Leftrightarrow o_k(e))$
  - Für alle Neuronen $h \in Mittelschicht$:
    $\delta_h := o_h(e) \cdot (1 \Leftrightarrow o_h(e)) \cdot \sum_{k \in Ausgabeschicht} w_{kh} \cdot \delta_k$
  - Für alle Gewichte w_{ji}:
    $w_{ji} := w_{ji} + \Delta w_{ji}$, wobei
    $\Delta w_{ji} = \eta \cdot \delta_j \cdot x_{ji}$

Eine ausführliche Herleitung dieses Algorithmus findet sich in [Mitchell, 1997a]. Bei Vladimir Vapnik werden neuronale Netze als Spezialfall seiner statistischen Lerntheorie eingeordnet und es wird insbesondere kritisiert, daß das Lernen mit der Backprop-Regel in neuronalen Netzen nur schwer kontrollierbar ist [Vapnik, 1995]. Insbesondere die Netzstruktur muß vom Benutzer vorgegeben werden und beeinflußt die Qualität des Lernergebnisses stark. Dies hat auch Achim Hoffmann mithilfe der Kolmogorov-Komplexität formal bestätigt: entweder man gibt eine geeignete Netzstruktur vor, oder man braucht unglaublich viele Beispiele, bis ein Begriff aus einer reichen Begriffsklasse gelernt werden kann [Hoffmann, 1997].

4 Wahrscheinlich annähernd korrektes Lernen

Seit Leslie Valiants Artikel hat sich die komplexitätstheoretische Herangehensweise an das Problem der Lernbarkeit von Begriffen aus Beispielen durchgesetzt [Valiant, 1984]. Das von ihm eingeführte Paradigma des wahrscheinlich annähernd korrekten Lernens (*probably approximately correct learning* – **PAC-learning**) geht von der Überlegung aus, daß es völlig aussichtslos ist, ein korrektes und vollständiges Lernverfahren zu fordern, das nach einer bestimmten Menge von Eingaben sicher und prompt das richtige Ergebnis abliefert und dann anhält. Beim *PAC-learning* schwächt man die Anforderung an die Korrektheit des Lernergebnisses ab. Das Lernergebnis ist nur noch mit einer bestimmten Wahrscheinlichkeit von $1 \Leftrightarrow \delta$ mit einem Fehler von höchstens ϵ richtig. Es wird also nur approximiert, nicht mehr identifiziert. Der Abschwächung bei der Korrektheit stehen aber zwei schwierige Anforderung an das Lernen gegenüber: Das Lernen soll in polynomiell beschränkter Rechenzeit zum Ergebnis kommen, nachdem es eine beschränkte Zahl von Beispielen gesehen hat. Die Beispiele sind in genau der Wahrscheinlichkeitsverteilung, in der tatsächlich Instanzen und Nicht-Instanzen des zu lernenden Begriffs vorkommen. Es wird also eine Stichprobe gegeben. Das Lernergebnis soll die Begriffsdefinition oder Erkennungsfunktion für den Begriff sein.

Hier wird das Szenario des *PAC-learning* kurz vorgestellt. Eine kurze, übersichtliche Einführung bietet [Hoffmann, 1991], eine ausführliche Behandlung des Bereiches bieten [Kearns, 1990][Kearns und Vazirani, 1994a], [Anthony und Biggs, 1992], [Natarajan, 1991].

Ein Lernalgorithmus für Begriffe einer Repräsentationsklasse (z.B. Boolsche Funktionen oder Formeln in einer Normalform mit k Termen) erhält Beispiele für einen bestimmten

Begriff c aus dieser Repräsentationsklasse. Die Beispiele werden zufällig gewählt, entsprechen aber der „wirklichen“, unbekannten Wahrscheinlichkeitsverteilung, d.h. jedes Beispiel wird mit genau der Wahrscheinlichkeit gezogen, welche die Wahrscheinlichkeitsverteilung angibt. Der Lernalgorithmus erhält außerdem die Parameter δ und ϵ, $\delta < 1$, $\epsilon < 1$. δ gibt an, mit welcher Wahrscheinlichkeit der Algorithmus den Begriff lernt. ϵ gibt an, wie nahe das Lernergebnis h dem tatsächlichen Begriff c ist, d.h. wieviele Instanzen oder Nicht-Instanzen falsch klassifiziert werden. h klassifiziert Beispiele annähernd korrekt, wenn die Wahrscheinlichkeit $e\Leftrightarrow$, daß ein negatives Beispiel als Instanz des Begriffs klassifiziert wird, und die Wahrscheinlichkeit $e+$, daß ein positives Beispiel als Nicht-Instanz des Begriffs klassifiziert wird kleiner ist als ϵ. Die beiden Parameter schwächen also die Anforderung an die Korrektheit einer gelernten Begriffsdefinition h ab.

Gegeben eine Menge $X \in LE$ von Objekten (alle möglichen Beispiele), ist ein Begriff c eine Teilmenge von X (nämlich alle Objekte, die der Begriff abdeckt) und eine Begriffsklasse C eine Teilmenge der Potenzmenge von X. Meist wird ein Begriff mit der Klassifikation durch den Begriff gleichgesetzt. c ist dann eine Funktion, die X auf 1 (positives Beispiel) oder 0 (negatives Beispiel) abbildet. Analog zur Repräsentationsklasse C des Zielbegriffs c wird die Repräsentationsklasse H des Lernergebnisses h definiert. H ist der Hypothesenraum, h eine Hypothese. Wenn $h(x) = c(x)$ für alle $x \in X$ einer Stichprobe ist, so ist die Hypothese h konsistent mit den gegebenen Beispielen.

Definition 23: PAC-Lernbarkeit. *Eine Begriffsklasse C ist (PAC-)lernbar durch einen Hypothesenraum H aus einer Menge von Beispielen X mit $x \in X$, wenn es einen Algorithmus $L(\delta, \epsilon)$ gibt, der*

- *bei allen beliebigen aber festen Wahrscheinlichkeitsverteilungen über X*
- *für alle $c \in C$*
- *in einer Zeit, die durch ein Polynom über $1/\epsilon, 1/\delta, \mid c \mid, \mid x \mid$ begrenzt ist*
- *mit einer Wahrscheinlichkeit von mindestens $1 \Leftrightarrow \delta$*

eine Hypothese $h \in H$ ausgibt, deren Fehler nicht größer ist als ϵ.

Dabei ist $0 \leq \epsilon \leq 1/2$ und $0 \leq \delta \leq 1/2$. Die Repräsentationsgröße $\mid c \mid$ wird unterschiedlich angegeben. Es kann z.B. die Länge einer Repräsentation bei effizienter Codierung sein. Entsprechend ist $\mid c \mid$ die Länge des größten Beispiels bei effizienter Codierung. L liefert also wahrscheinlich ein annähernd korrektes Ergebnis in polynomieller Zeit ab. Damit das möglich ist, muß L aus nur polynomiell vielen Beispielen lernen können, wobei die Verarbeitung jedes Beispiels nur polynomielle Zeit benötigt.

In diesem Szenario kann man nun für bekannte Sprachklassen bzw. ihre Automaten die prinzipielle Lernbarkeit von Begriffen, die in dieser Sprache ausgedrückt sind, untersuchen. In den letzten Jahren ist eine Fülle von Beweisen zur Lernbarkeit bestimmter Repräsentationsklassen erarbeitet worden. Interessanterweise ist die Hierarchie der Lernbarkeit nicht dieselbe wie die übliche Komplexitätshierarchie. Das bekannte Beispiel hierfür sind k-KNF und k-Term-DNF.

Definition 24: k-KNF. *Eine Formel $C_1 \wedge ... \wedge C_l$, bei der jedes C_i eine Disjunktion von höchstens k Literalen über Boolschen Variablen ist, ist in k-konjunktiver Normalform (k-KNF).*

Definition 25: k-Term-DNF. *Eine Formel $T_1 \wedge ... \wedge T_k$, bei der jeder Term T_i eine Konjunktion über höchstens n Boolschen Variablen ist, ist in k-Term disjunktiover Normalform (k-Term-DNF).*

Natürlich ist k-KNF ausdrucksstärker als k-Term-DNF, denn man kann jeden Ausdruck in k-Term-DNF in k-KNF schreiben, aber nicht umgekehrt. Die Repräsentationsklasse $C = k \Leftrightarrow KNF$ ist lernbar durch $H = k \Leftrightarrow KNF$. Die Repräsentationsklasse $C = k \Leftrightarrow Term \Leftrightarrow DNF$ ist nicht lernbar durch $H = k \Leftrightarrow Term \Leftrightarrow DNF$, wohl aber lernbar durch $H = k \Leftrightarrow KNF$. Der mächtigere Hypothesenraum, der immer noch polynomiell lernbar ist, erleichtert das Lernen. Der 'kleinere' Hypothesenraum macht das Lernen schwieriger, weil je Beispiel mehr Rechenzeit benötigt wird. Die Lernbarkeit ergibt sich ja einerseits aus der Anzahl der benötigten Beispiele, andererseits aus der Rechenzeit pro Beispiel. Obwohl das Lernen eines Begriffs in der Klasse der k-DNF-Formeln nur polynomiell viele Beispiele braucht, ist er nicht lernbar, weil er im schlimmsten Falle zu lange für die Verarbeitung eines Beispiels braucht. Man kann die Anzahl der Beispiele abschätzen, die man dem Algorithmus geben muß, damit er lernen kann (Stichprobenkomplexität). Damit hat man dann den ersten Schritt eines PAC-Lernbarkeitsbeweises geschafft.

4.1 Stichprobenkomplexität

Nehmen wir an, ein Lernalgorithmus gibt nur die Hypothesen aus, die alle positiven Beispiele und kein negatives Beispiel abdecken. Wenn wir weiterhin annehmen, daß der Zielbegriff in der Hypothesensprache enthalten ist, dann können wir recht einfach die Stichprobenkomplexität berechnen, nach denen der Algorithmus spätestens den Zielbegriff wahrscheinlich annähernd korrekt ermittelt hat.

Definition 26: Stichprobenkomplexität. *Die Stichprobenkomplexität eines Algorithmus für gegebenes H, C, ϵ und δ ist die Anzahl von Beispielen m, nach denen der Algorithmus spätestens ein beliebiges $c \in C$ mit Wahrscheinlichkeit δ bis auf einen Fehler von ϵ approximiert (PAC-gelernt) hat.*

Zunächst benutzen wir einen Hypothesenraum mit endlicher Größe $\mid H \mid$.

Theorem 27: Stichprobenkomplexität endlicher Hypothesenräume. *Die Stichprobenkomplexität m eines endlichen Hypothesenraumes H mit $H = C$ ist begrenzt durch*

$$m \geq \frac{1}{\epsilon}(ln \mid H \mid + ln(1/\delta)) \tag{10}$$

das kleinste m, für das die Ungleichung gilt, gibt die Anzahl von Beispielen an, nach denen spätestens jedes $c \in C$ PAC-gelernt werden kann.

Beweis. Nach m Beispielen wird jeder Lernalgorithmus, der nur Hypothesen behält, die konsistent sind mit den Beispielen, und in endlichen Hypothesenräumen arbeitet, mit einer Wahrscheinlichkeit $1 \Leftrightarrow \delta$ eine Hypothese ausgeben, deren Fehler höchstens ϵ beträgt. Die Anzahl der Beispiele hängt also linear von $1/\epsilon$ und logarithmisch von $\mid H \mid$ und $1/\delta$ ab. David Haussler hat die Formel 10 anhand des Versionenraums bewiesen [Haussler, 1988]. Der Versionenraum enthält ja zu jedem Zeitpunkt nur mit den bisher gesehenen Beispielen konsistente Hypothesen und der Hypothesenraum ist endlich. Damit erfüllt er genau die Annahmen. Der Beweis betrachtet den schlimmsten Fall: bezüglich der bisher gesehenen Beispiele ist eine Hypothese im Versionenraum zwar korrekt, tatsächlich ist ihr Fehler

aber größer als ϵ. Die Wahrscheinlichkeit, daß eine beliebige Hypothese mit Fehler größer als ϵ ein zufällig gezogenes Beispiel richtig klassifiziert, ist also höchstens $1 \Leftrightarrow \epsilon$. Bei m Beispielen beträgt die Wahrscheinlichkeit also höchstens $(1 \Leftrightarrow \epsilon)^m$. Bei k Hypothesen ist die Wahrscheinlichkeit, daß eine von ihnen schlecht ist (d. h. einen Fehler größer als ϵ hat), höchstens $k(1 \Leftrightarrow \epsilon)^m$. k kann natürlich nicht größer sein als der Hypothesenraum. Im schlimmsten Fall ist die Wahrscheinlichkeit, daß eine der Hypothesen im Versionenraum einen Fehler größer als ϵ hat, also $\mid H \mid (1 \Leftrightarrow \epsilon)^m$. Dies beschränkt die Wahrscheinlichkeit, daß m Beispiele nicht ausreichten, um die schlechten Hypothesen aus dem Versionenraum zu tilgen. Da ϵ zwischen 0 und 1 liegt, können wir sie ganz grob abschätzen als $\mid H \mid e^{-\epsilon m}$ (oft additive Chernoff-Schranke genannt). Damit ein Begriff PAC-gelernt wird, muß diese Wahrscheinlichkeit kleiner sein als δ:

$$\mid H \mid e^{-\epsilon m} \leq \delta \tag{11}$$

Eine Umformung von 11 ergibt genau 10. □

Ein wichtiges Konzept für den Nachweis der wahrscheinlich annähernd korrekten Lernbarkeit ist die Vapnik-Chervonenkis-Dimension. Die **Vapnik-Chervonenkis-Dimension** soll die Ausdrucksstärke einer Repräsentationsklasse angeben. Damit kann die Komplexität ("Größe") eines Hypothesenraums auch für unendliche Hypothesenräume angegeben werden. Und dies kann wiederum verwendet werden, um die Stichprobenkomplexität auszurechnen.

Definition 28: Zerschmettern einer Beispielmenge. *Sei H der Hypothesenraum über X und S eine m-elementige Teilmenge von X. S wird von H zerschmettert (shattered), falls es für alle $S' \subseteq S$ eine Hypothese $h_{S'} \in H$ gibt, die S' abdeckt, d.h. $S \cap h_{S'} = S'$.*

Alle Teilmengen von S werden also durch Hypothesen in H erkannt.

Definition 29: Vapnik-Chervonenkis-Dimension (VC-Dimension). *Die Vapnik-Chervonenkis-Dimension von H, $VCdim(H)$, ist die Anzahl der Elemente von der größten Menge S, wobei S von H zerschmettert wird.*

$$VCdim(H) = max\{m : \exists S \subseteq X, \mid S \mid = m, H \text{ zerschmettert } S\}$$

Sie gibt also an, wieviele Unterschiede H machen kann. Wenn es kein Maximum der Kardinalität von S gibt, ist $VCdim$ unendlich.

Theorem 30: VC-Dimension endlicher Hypothesenräume. *Wenn der Hypothesenraum H endlich ist, dann ist $VCdim(H) \leq log_2(\mid H \mid)$.*

Beweis. Um eine Menge der Größe m zu zerschmettern, sind mindestens 2^m verschiedene Hypothesen nötig, weil es ja 2^m verschiedene Teilmengen gibt. □

Wenn wir umgekehrt die größte Menge wissen wollen, die ein Hypothesenraum zerschmettern kann, so müssen wir m bestimmen, also $log_2(\mid H \mid)$ (d.h. $log_2(2^m)$).

Als einfaches Beispiel zur Illustration nehmen wir

- für X Punkte in einer Ebene, dargestellt durch (x_i, y_i);
- für H nehmen wir ein Perzeptron mit zwei Eingängen, das in einem Zustand, der durch zwei Gewichte w_1 und w_2 und einen Schwellwert t gegeben ist, die folgende Boolsche Funktion berechnet: $h(x_i, y_i) = 1$ gdw. $t \leq w_1 x_i + w_2 y_i$.

Wenn wir eine 3-elementige Teilmenge S von X haben, wobei für keinen der drei Punkte $x_i = x_j$ oder $y_i = y_j$, so gibt es 2^3 Möglichkeiten, die Elemente von S in positive und negative Beispiele zu klassifizieren. Die 8 verschiedenen Hypothesen sind:

S	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
(x_1, y_1)	-	+	-	+	-	+	-	+
(x_2, y_2)	-	-	+	+	-	-	+	+
(x_3, y_3)	-	-	-	-	+	+	+	+

Man kann sich eine Hypothesen als trennende Linie zwischen positiven und negativen Beispielen vorstellen. $VCdim(H)$ ist also mindestens schon einmal 3. Aber könnte H nicht auch eine 4-elementige Teilmenge von X zerschmettern? Das Bild 5 zeigt, daß im ersten Fall $\{(x_1, y_1), (x_3, y_3)\}$ und $\{(x_2, y_2), (x_4, y_4)\}$ nicht durch eine Linie getrennt werden können und im zweiten Fall $\{(x_4, y_4)\}$ nicht von den anderen drei Beispielen getrennt werden kann. $VCdim(H)$ ist also nicht nur mindestens 3, sondern genau 3.

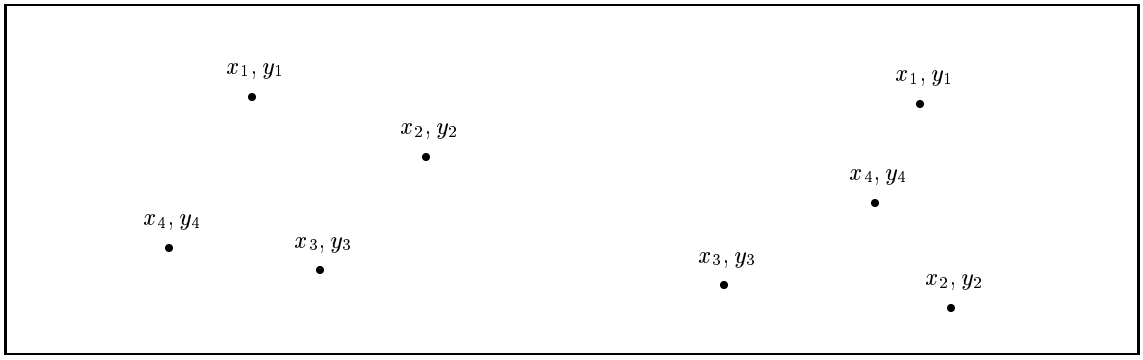


Abbildung5. 4-elementige Teilmengen von X

Es ist oft sehr schwierig, die $VCdim$ genau zu bestimmen. Oft werden nur Abschätzungen gefunden. Den Zusammenhang zwischen der $VCdim$ einer Begriffsklasse und ihrer wahrsheinlich annähernd korrekten Lernbarkeit geben die folgenden Ergebnisse an:

Theorem 31: *C ist PAC-lernbar gdw. $VCdim(C)$ endlich ist [Blumer et al., 1990].*

Analog zu der Bestimmung der höchstens nötigen Beispiele über die Größe des Hypothesenraums (Gleichung 10), kann man die Anzahl der nötigen Beispiele über die VCdim abschätzen.

Theorem 32: Stichprobenkomplexität unendlicher Hypothesenräume – obere Schranke. *Die Stichproblenkomplexität m eines unendlichen Hypothesenraumes H mit $H = C$ ist begrenzt durch*

$$m \geq \frac{1}{\epsilon}(4log_2(2/\delta) + 8VCdim(H)log_2(13/\epsilon)) \tag{12}$$

Wir wissen damit, welche Anzahl von Beispielen hinreicht, um einen Begriff zu lernen, wenn er denn PAC-lernbar ist. Allerdings hilft uns dies nur, wenn wir sicherstellen können, daß wir tatsächlich eine Stichprobe ziehen und nicht einfach einen Datensatz vor uns haben, der eventuell nach einer ganz anderen Wahrscheinlichkeitsverteilung zustande kam als der tatsächlichen.

4.2 Einige Ergebnisse zu neuronalen Netzen

4.2.1 Ausdruckskraft

Die Klasse der mehrschichtigen neuronalen Netze kann zwar jede beliebige Funktion über einer Menge von Attributwerten ausdrücken, aber bei einem speziellen Netzwerk kann es sein, daß die Anzahl der Mittelschichtsknoten nicht ausreicht. Für Boolsche Funktionen von n Eingabewerten braucht man $2^n/n$ Mittelschichtknoten. Es gibt $O(2^n)$ Gewichte und wir brauchen mindestens 2^n bits, um eine Boolsche Funktion zu spezifizieren[Russell und Norvig, 1995].

Jede Boolsche Funktion kann durch ein Netz mit zwei Schichten ausgedrückt werden, wobei aber die Anzahl der Mittelschichtknoten exponentiell bzgl. der Eingabeknoten wächst. Für jeden Eingabevektor kann man einen Knoten in der Mittelschicht konstruieren, dessen Gewichte genau bei diesem Eingabevektor und sonst nie so hoch sind, daß der Knoten aktiv ist[Mitchell, 1997b].

Beliebige Funktionen können approximiert werden mit beliebiger Genauigkeit bei dreischichtigen Netzwerken mit sigmoiden Neuronen in den Mittelschichten[Mitchell, 1997b].

4.2.2 Hypothesenraum

Der Hypothesenraum besteht bei Backpropagation in den Gewichten, ist also bei n Gewichten ein n-dimensionaler Euklidischer Raum. Dieser Hypothesenraum ist kontinuierlich, nicht diskret. Statt einer Generalisierungsbeziehung als Ordnung im Hypothesenraum gilt hier die Ableitung bzw. der Fehlergradient als Struktur [Mitchell, 1997b]. Als Hypothesenraum wird auch die Klasse von Funktionen bezeichnet, die ein neuronales Netz ausdrücken kann.

4.2.3 VCdim

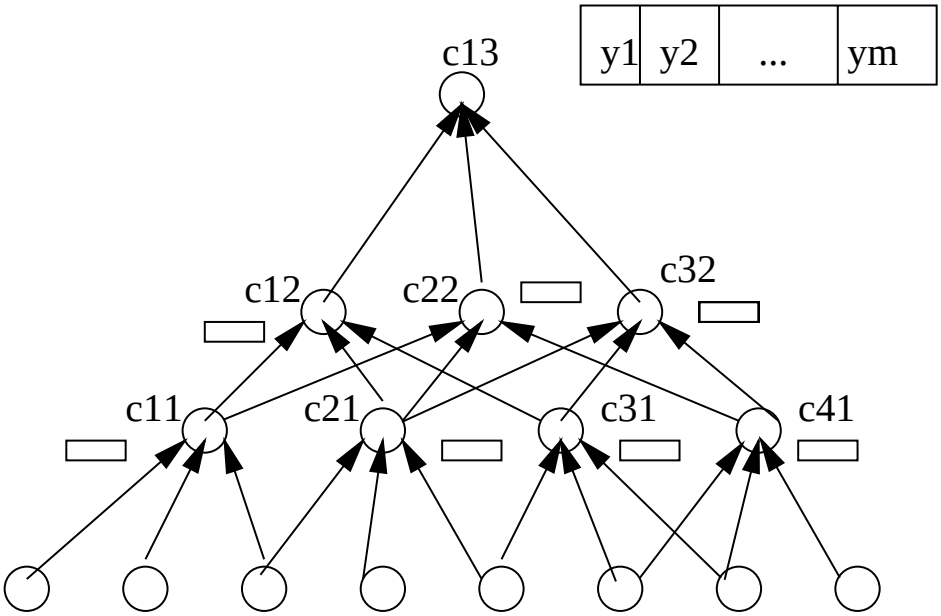
Die VCdim wird errechnet, indem das neuronale Netz als geschichteter, gerichteter, azyklischer Graph aufgefaßt wird, und indem für die einzelnen Knoten in diesem Graphen eine VCdim errechnet wird. Jedes Neuron N_i stellt eine Boolsche Funktion $c_i : \mathcal{R}^r \rightarrow \{0,1\}$ dar, wobei die c_i aus einer Klasse von Funktionen C stammen. Bei r Eingängen in das Neuron sei C eine Klasse über dem r-dimensionalen Euklidischen Raum $\mathcal{R}^r$. Wenn es sich bei den Neuronen um Perzeptronen handelt, die Boolsche Funktionen über $\{0,1\}^r$ darstellen, dann ist ihre VCdim $r+1$.

Das gesamte Netz ist nun ein gerichteter azyklischer Graph G, dessen interne Knoten beschriftet sind mit $c_i \in C$. Als *G-Komposition von C* werden alle Funktionen bezeichnet, die der Graph G ausdrücken kann, wenn die einzelnen Neuronen jeweils eine Funktion $c_i \in C$ verwenden. Die G-Komposition von C ist also der Hypothesenraum von G. Wenn wir m Beispiele haben, r Eingänge in die internen Knoten und nur einen Ausgabeknoten, dann sieht der Graph so aus wie in Abbildung 6.

Die VCdim des Netzwerks mit Perzeptronen ist kleiner gleich $2(r+1)s \cdot log(e \cdot s)$.

Theorem 3.6 in [Kearns und Vazirani, 1994b]

> Sei G ein geschichteter, gerichteter, azyklischer Graph mit n Eingabeknoten und $s \geq 2$ internen Knoten, jeder mit r Eingängen.



x11	x21	x31	x41	x51	x61	x71	xn1
x12	x22	x32	x42	x52	x62	x72	xn2
...							
x1m	x2m	x3m	x4m	x5m	x6m	x7m	xnm

Abbildung6. Berechnungen der G-Komposition von C für m Beispiele

Sei C eine Begriffsklasse über $\mathcal{R}^r$ mit der VCdim d und GC die G-Komposition von C.

Dann ist VCdim$(GC) \leq 2dslog(es)$, wobei e die Basis des Logarithmus naturalis ist.

Beweis: Sei S die Menge von m Eingabevektoren der Länge n (Beispiele): $x_1, ..., x_m \in \mathcal{R}^n$.

Für diese Menge beschriften wir jeden Knoten N_i in G mithilfe c_i, so daß für jeden Eingabevektor der binäre Ausgabewert festgelegt ist. An jedem Knoten stehen nun also für alle m Eingabevektoren (Beispiele) die jeweils berechneten Werte, so daß wir Vektoren der Länge m mit binären Ausgabewerten an jedem Knoten haben. Den Graphen mit all diesen Vektoren nennen wir eine *Berechnung von G über S* (s. Abbildung 6).

Betrachten wir jetzt diejenigen Ausgabewerte, die zu positiven Beispielen gehören. Dieser Graph drückt die voneinander echt verschiedenen positiven Beispiele aus. Kearns und Vazirani nennen diesen Graphen $\Pi_{CG}(S)$. Zwei Beispiele sind dann echt verschiedenen, wenn es einen Knoten gibt, an dem sie für einen Eingabevektor unterschiedliche Ausgabewerte berechnen. Natürlich kann es nicht mehr verschiedene positive Beispiele geben als es verschiedene Berechnungen von G über S gibt. Wir

haben also einmal alle möglichen Berechnungen von G über C, $T_{CG}(S)$, und dann die Berechnungen für positive Beispiele.

$\Pi_{CG}(S) \leq T_{CG}(S)$

Wir wollen aber die Berechnungen von G über C für m Beispiele in Abhängigkeit von der VCdim und den Beispielen abschätzen. Dazu wird die Wachstumsfunktion $\Pi_C(m)$ verwendet.

$\Pi_C(m) = max\{\mid \Pi_C(S) \mid : \mid S \mid = m\}$.

Die Wachstumsfunktion soll angeben, wie schnell aufgrund neuer Beispiele mehr unterschiedliche Funktionen im Netz bzw. Graphen dargestellt werden können. Wenn die VCdim nicht beschränkt ist, sind bei m Beispielen 2^m verschiedene Berechnungen möglich. Falls das Netz aber gar nicht alle Unterschiede ausdrücken kann, weil seine VCdim eben nicht unendlich ist, so gibt es weniger als 2^m mögliche Bereechnungen. Wenn deren Anzahl als Polynom über der VCdim und der Anzahl der Beispiele angegeben wird, haben wir eine genauere Schranke gefunden! Es gibt nun einen Beweis, dass

$\Pi_C(m) \leq (\frac{em}{d})^d$, wobei d die VCdim und e die Basis des natürlichen Logarithmus ist [Kearns und Vazirani, 1994b]. [4]

Betrachten wir noch einmal den beschrifteten Graphen von Abbildung 6. Mit einem induktiven Argument zeigen wir, dass weniger als 2^m verschiedene Ausgabevektoren von den Knoten erzeugt werden. Wenn wir den Graphen um einen Knoten, zunächst den Ausgabeknoten, verkürzen und wir kennen die tatsächlichen Belegungen, die die m Beispiele bewirkt haben, dann können wir die Belegungen, die seine untergeordneten Knoten produzieren, um alle Belegungen reduzieren, die nicht zu den gegebenen Belegungen des Oberknoten passen. Dies tun wir für alle s internen Knoten. Wir haben also

$\Pi_C(m)^s \leq (\frac{em}{d})^{ds}$, wobei d die VCdim ist.

Damit GC m Punkte zerschmettern kann, muß die Ungleichung $(em/d)^{ds} \geq 2^m$ gelten.

Umgekehrt gilt: Wenn

$(em/d)^{ds} < 2^m$ für irgendein m, dann ist m eine obere Schranke für die VCdim(GC).

Diese Ungleichung ist erfüllt für $m = 2ds \cdot log(es)$, wenn $s \geq 2$. Also VCdim(GC) $\leq 2ds \cdot log(es)$, q.e.d.

4.2.4 Stichprobengröße

Wenn wir das Theorem 3.6 so hinnehmen, können wir die VCdim(GC) in die Formel 12 aus dem Skript zur Stichprobengröße einsetzen. Wenn wir als die VCdim an jedem Neuron $d = r + 1$ annehmen, erhalten wir:

Anzahl nötiger Beispiele Wir schätzen die Stichprobengröße m ab:

$m \geq 1/\epsilon(4log(2/\delta) + 8VCdim(GC)log(13/\epsilon))$

[4] Man kommt auf den natürlichen Logarithmus, weil man eigentlich rechnen müsste $\frac{\left(1+\frac{d}{m}\right)^{m\,d}}{\left(\frac{d}{m}\right)}$. Wir nutzen aus, dass $e^d = lim_{m\to\infty}\left(1+\frac{d}{m}\right)^m$, erhalten $\frac{e^d}{\left(\frac{d}{m}\right)^d} = \left(\frac{e}{\frac{d}{m}}\right)^d = \left(\frac{me}{d}\right)^d$.

d.h.:$1/\epsilon(4log(2/\delta) + 8 \cdot 2(r+1)s \cdot log(es)log(13/\epsilon))$

Wenn wir ein neuronales Netz mit einem Ausgabe-, 4 Mittelschichtknoten und 960 Eingabeknoten haben, so ist dessen VCdim $= 2(r+1)slog(es) = 25072,49$. Das Netz kann also sehr viele Unterschiede machen und daher werden wir vermutlich sehr viele Beispiele benötigen.

Beispiel mit $\epsilon = 0,2, \delta = 0,1, r = 30 \cdot 32 = 960, s = 5$:
$5(4log(20) + 8 \cdot 2 \cdot 961 \cdot 5 \cdot log(e \cdot 5)log(65))$
$5(5,2 + 8 \cdot 25072,49 \cdot 1,8128)$
$= 1.818.082,4$

Wie man deutlich sieht, müssen die Schranken viel feiner gesetzt werden! Die PAC-Theorie reflektiert (noch) nicht unsere Erfahrung.

5 Logik-orientiertes induktives Lernen

Wie oben schon dargestellt, ist mit Lernen meist ein induktiver Schluß gemeint. Diesen induktiven Schluß für die Prädikatenlogik konstruktiv zu formalisieren, so daß für eine gegebene Menge von Daten (und eine Theorie) die speziellste (oder generellste) Verallgemeinerung gefunden werden kann, ist in jüngster Zeit ein wieder lebhaft diskutiertes Thema geworden. Es geht darum, gegebene Daten (Beispiele, Beobachtungen) zusammenfassend zu beschreiben. Diese Beschreibung der Daten soll verständlich und übersichtlich sein. Untersucht wird die Beziehung zwischen den Daten und ihrer zusammenfassenden Beschreibung. Es sollen also nicht irgendwie Regeln gefunden werden, die bei den Beispielen oder Beobachtungen gelten, sondern es sollen Verhältnisse zwischen Beschreibung und Beschriebenem zugesichert werden: generellste Spezialisierung bezüglich Beispielen und Hintergrundwissen oder speziellste Generalisierung von Beispielen bezüglich des Hintergrundwissens. Insofern unterscheidet sich dies Paradigma ganz wesentlich von dem der Funktionsapproximation.

Definition 33: Begriffslernen $BL(\vdash, \succeq, LH, LE, LB)$. *Die Lernaufgabe* **Begriffslernen** *der induktiven logischen Programmierung ist:*

Gegeben: *positive und negative Beispiele E in einer Sprache LE,*
Hintergrundwissen B in einer Sprache LB, wobei $B \cup E \not\models \Box$ und $B \not\models E$.

Ziel: *eine Hypothese H in einer Sprache LH, so daß folgendes gilt:*

Konsistenz: $B, H, E \not\models \Box$,

Vollständigkeit: $B, H \models E^+$,

Korrektheit: $\forall e \in E^- : B, H \not\models e$

Meist besteht sowohl B als auch E aus variablenfreien Fakten. LH ist eine eingeschränkte Prädikatenlogik. Die logische Folgerung wird durch eine Ableitungsrelation $\vdash$ modelliert. Zur Auswahl aus Hypothesen, die korrekt und vollständig sind, wird oft eine Präferenzrelation $\succeq$ angegeben, die z.B. die allgemeinsten Hypothesen oder die speziellsten auswählt.

Definition 34: Konsistenzproblem $K(\vdash, LH, LE, LB)$. *Das Problem, zu entscheiden, ob es eine konsistente, vollständige und korrekte Hypothese gibt, wird* **Konsistenzproblem** *genannt.*

Wichtig ist die kompakte und übersichtliche Darstellung von Relationen, die mit der Attribut–Werte–Repräsentation nicht möglich ist. Dort können zum Beispiel Graphen nicht so repräsentiert werden, daß der Begriff *Pfad* oder *beidseitig verbunden* gelernt werden könnte.

5.1 Die Generalisierungsbeziehung bei Klauseln

Bei den logik-basierten Verfahren geht es darum, genau anzugeben, wann ein Literal oder eine Klausel eine Generalisierung (bzw. Spezialisierung) eines anderen Literals bzw. einer anderen Klausel darstellt [5]. Durch die partielle Ordnung der Generalisierung wird auch die Äquivalenz von Klauseln definiert. Aus dem Äquivalenzbegriff kann dann ein Redundanzbegriff entwickelt werden. Eine Klausel ist **redundant**, wenn Literale aus ihr gestrichen werden können, und die verkürzte Klausel äquivalent zu der ursprünglichen Klausel ist. Eine nichtredundante Klausel wird auch **reduziert** genannt. Das Erkennen von Redundanzen ist für die induktive Logikprogrammierung wichtig, um möglichst einfache Programme erzeugen zu können.

5.1.1 Implikation

Eine Möglichkeit, die Generalisierung zu beschreiben, verwendet die **Implikation**:

Definition 35: Genereller-als Beziehung unter Implikation. *Eine Klausel $C1$ ist genereller als eine andere Klausel $C2$, wenn gilt: $C1 \rightarrow C2$.*

Die Generalisierungsrelation der Implikation schließt das Hintergrundwissen in die Definition ein.

Definition 36: Genereller-als Beziehung unter Implikation mit Hintergrundwissen. *Sei B eine Konjunktion von Hornklauseln. Eine Klausel $C1$ ist genereller als eine Klausel $C2$ bezüglich B gdw. $B, C1 \models C2$.*

Ein Beispiel:

> Sei B gegeben durch: $append([], C, C)$. Sei das zu generalisierende Beispiel $C2$: $append([1,2],[3],[1,2,3])$. Eine Generalisierung von $C2$ relativ zu B unter der Implikation ist $C1$: $append([A \mid B], C, [A \mid E]) \leftarrow append(B, C, E)$. $C1$ und B bilden gemeinsam das klassische Programm für die Listenkonkatenation. Damit kann $C2$ bewiesen werden.

Die Äquivalenz von Klauseln wird in der folgenden Definition formalisiert.

Definition 37: Äquivalenz von Klauseln unter Implikation. *Sei B eine Konjunktion von Hornklauseln. Zwei Klauseln $C1$ und $C2$ sind* logisch äquivalent *bezüglich B, $C1 \equiv C2$, gdw.*

$$B, C1 \models C2 \text{ und } B, C2 \models C1.$$

Die folgende Definition erklärt die Redundanz von Literalen in einer Klausel.

Definition 38: Redundanz von Literalen unter Implikation. *Ein Literal L ist* logisch redundant *in einer Klausel C für eine Konjunktion von Hornklauseln B, wenn C und $(C \setminus \{L\})$ bezüglich B logisch äquivalent sind.*

[5] Dieser Abschnitt ist stark an [Jung, 1992] angelehnt, die Beispiele sind direkt übernommen.

Ein Beispiel:

Sei B gegeben durch:

$E : member(X, [X \mid Y])$.
$C : member(X, [Y \mid Z]) \leftarrow member(X, Z), member(Y, [Y \mid Z])$.
Das zweite Literal im Klauselkörper von C ist redundant, da es eine Relation beschreibt, die durch die Klausel E unmittelbar gegeben ist. Das redundante Literal kann gestrichen werden und es ergibt sich ein vereinfachtes Logikprogramm B' mit den Klauseln:
$E : member(X, [X \mid Y])$.
$C' : member(X, [Y \mid Z]) \leftarrow member(X, Z)$.

Der Nachteil bei der Verwendung der Implikation als Generalisierungsrelation ist, daß es den Hypothesenraum bei der Anwendung der einzelnen Generalisierungsoperationen nicht verkleinert. Es gibt eine unendliche Kette von immer spezielleren Hypothesen, die alle die Beispiele abdecken. Der Hypothesenraum hat nicht die Verbandseigenschft, daß es zu zwei Klauseln genau eine speziellste Generalisierung gibt. Außerdem ist seine Anwendung zu aufwendig. Um eine Generalisierung zu finden, müssen wir die Klausel finden, die die gegebenen Beispiele impliziert. Dies ist schwierig, weil es darauf hinausläuft, die logische Folgerung zwischen Klauseln als Grundlage zu nehmen, die nicht im allgemeinen Fall entscheidbar ist.

5.1.2 θ-Subsumtion

Um die logische Folgerung bei der generalisierungsbeziehung von Klauseln zu umgehen, wird die schwächere **Subsumtionsbeziehung** bevorzugt. Immer, wenn die Subsumtionsbeziehung gilt, gilt auch die logische Folgerung – aber nicht umgekehrt!

Definition 39: Genereller-als Beziehung unter θ-Subsumtion. *Eine Klausel $C1$ ist genereller als eine andere Klausel $C2$, wenn gilt: $C1$ subsumiert $C2$.*

Bei Literalen ist das einfach.

Definition 40: θ-Subsumtion von Literalen. *Ein Literal $L1$ subsumiert ein anderes Literal $L2$, wenn es eine Substitution σ gibt, so daß $L1\sigma = L2$.*

Damit wird Substitution zur Grundlage der Formalisierung von Induktion.

Wir können uns Generalisierung für Klauseln einmal (semantisch) an den Objekten (Daten, Beispielen) einer logischen Struktur deutlich machen. Eine Klausel $C1$ ist genereller als eine andere Klausel $C2$ (geschrieben: $C1 \geq C2$),wenn sie mehr Objekte abdeckt. So ist zum Beispiel

$$säugetier(X) \leftarrow tier(X) \geq säugetier(rex) \leftarrow tier(rex)$$

und

$$säugetier(X) \leftarrow tier(X) \geq säugetier(X) \leftarrow tier(X), im\_haus(X)$$

Die Säugetiere umfassen einmal nur *rex* (und vielleicht noch andere Objekte), einmal mindestens die Schnittmenge der beiden Objektmengen, schließlich sogar mindestens alle Tiere. An dem Beispiel ist auch deutlich zu sehen, daß neben der Subsumtion, auch die

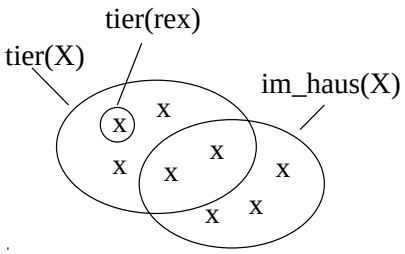


Abbildung7. Objektmengen

logische Folgerung gilt. Zum anderen können wir uns die Generalisierung aber auch (syntaktisch) an der Gestalt der Klauseln klarmachen. Wenn wir sie als Mengen schreiben, so ist die generellere Klausel eine Teilmenge der spezielleren.

$$\{\neg tier(X), \neg im\_haus(X), säugetier(X)\} \supseteq \{\neg tier(X), säugetier(X)\}.$$

Natürlich müssen die Terme so substituiert werden, daß es paßt.

Definition 41: θ-Subsumtion von Klauseln. *Eine Klausel $C1$ θ-subsumiert eine Klausel $C2$ ($C1 \geq_\theta C2$), gdw. es eine Substitution θ gibt, so daß*

$$C2 \supseteq C1\theta.$$

Sollen nur Hornklauseln betrachtet werden, kann man die Definition umformulieren. Bei Hornklauseln nennen wir das positive Literal den Klauselkopf, die negativen Literale den Klauselkörper. Wir beziehen uns auf den Klauselkopf der Klausel $C1$ mit $C1_{kopf}$, entsprechend schreiben wir $C1_{körper}$ für den Klauselkörper von $C1$. Bei Hornklauseln besteht also der Klauselkopf immer aus genau einem positiven Literal und der Klauselkörper aus einer Disjunktion von negativen Literalen.

Definition 42: θ-Subsumtion von Hornklauseln. *Eine Hornklausel $C1 = C1_{kopf} \leftarrow C1_{körper}$ θ-subsumiert eine Hornklausel $C2 = C2_{kopf} \leftarrow C2_{körper}$, $C1 \geq_\theta C2$, gdw. es eine Substitution θ gibt, so daß*
$C1_{kopf}\theta = C2_{kopf}$, *und*
$C1_{körper}\theta \subseteq C2_{körper}$.

Ein Beispiel:

Seien zwei Klauseln $C1$ und $C2$ wie folgt gegeben:
$C1 : member(X, [Y \mid Z]) \leftarrow member(X, Z).$
$C2 : member(3, [1, 2, 3]) \leftarrow member(3, [2, 3]), member(3, [3]).$

$C1$ θ-subsumiert $C2$, da nach der Anwendung der Substitution
$\theta = \{X/3, Y/1, Z/[2, 3]\}$ auf $C1$ gilt:
$C1_{kopf}\theta = member(3, [1, 2, 3]) = C2_{kopf}$, und
$C1_{körper}\theta = member(3, [2, 3]) \subseteq C2_{körper}$.

Das Generalisierungsmodell der θ-Subsumtion schränkt den Hypothesenraum bei der Generalisierung ein. Als Begründung hierfür kann das obige Beispiel (mit *append*) für

die Generalisierung durch Implikation gesehen werden. Die dort erzeugte Generalisierung unter logischer Konsequenz ist keine Generalisierung unter θ-Subsumtion.

Analog zur Generalisierung unter logischer Konsequenz führen wir auch für die θ-Subsumtion den Begriff der Äquivalenz ein.

Definition 43: Äquivalenz von Klauseln unter θ-Subsumtion. *Zwei Klauseln $C1$ und $C2$ sind äquivalent unter θ-Subsumtion, $C1 \equiv_\theta C2$, gdw.*

$$C1 \geq_\theta C2 \text{ und } C2 \geq_\theta C1$$

Nun können wir auch definieren, was wir unter *"echt genereller"* verstehen.

Definition 44: Echte genereller-als Beziehung unter θ-Subsumtion. *Eine Klausel $C1$ ist echt genereller unter θ-Subsumtion als $C2$ gdw.*

$$C1 \geq_\theta C2 \text{ und nicht } C1 \equiv_\theta C2$$

Die folgende Definition führt die Redundanz unter θ-Subsumtion ein.

Definition 45: Redundanz von Literalen unter θ-Subsumtion. *Ein Literal L einer Klausel C ist redundant unter θ-Subsumtion, wenn*

$$C \geq_\theta C \setminus \{L\}$$

Definition 46: Reduziertheit einer Klausel unter θ-Subsumtion. *Eine Klausel ist reduziert, wenn sie keine redundanten Literale enthält.*

In der folgenden Klausel ist das letzte Literal redundant unter θ-Subsumtion.

$C1' : member(X, [Y \mid Z]) \leftarrow member(X, Z), member(X, U).$

Wird das Literal $member(X, U)$ aus $C1'$ gestrichen, ergibt sich:
$C1 : member(X, [Y \mid Z]) \leftarrow member(X, Z).$

Es gilt $C1' \geq_\theta C1$, da nach Anwendung der Substitution $\theta = \{U/Z\}$ gilt, daß $C1'\theta = C1$.

Theorem 47: *Nach Plotkin ist die reduzierte Form einer Klausel (bis auf alphabetische Varianten) eindeutig [Plotkin, 1970].*

Ein Nachteil bei der Generalisierung durch θ-Subsumtion ist, daß kein Hintergrundwissen in den Generalisierungsschritt eingebunden wird. Wir werden aber in Abschnitt 5.1.3 ein Verfahren sehen, das Hintergrundwissen einbezieht.

5.1.3 Generalisierte θ-Subsumtion

Die generalisierte θ-Subsumtion generalisiert zwei funktionsfreie Hornklauseln bezüglich gegebenem Hintergrundwissen. Die Substitution θ ist hier derart, daß für alle Variablen der zu generalisierenden Klausel neue Konstante eingeführt werden (*Skolemsubstitution*). Wir belassen σ als Bezeichner für (irgend)eine Substitution von Termen. Dann definiert Buntine die Generalisierung folgendermaßen [Buntine, 1988][6] :

[6] Der Existenzquantor in der Definition dient lediglich dem Binden von freien Variablen.

Definition 48: Generalisierte θ-Subsumtion. *Nach Buntine θ-subsumiert eine Hornklausel $C1$ eine Hornklausel $C2$ mit Hintergrundwissen B ($C1 \geq_B C2$) gdw.*

$$\exists\sigma,\ so\ daß\ C1_{kopf}\sigma = C2_{kopf}\ und\ B, \neg C2_{körper}\theta \models \exists(\neg C1_{körper}\sigma\theta)$$

Wir müssen also die generellere Klausel durch Substitutionen erst einschränken, damit sie aus der spezielleren folgt[7].

Die Definition operationalisiert Buntine, indem er für jede Klausel C_i der generelleren Klauselmenge zeigt, daß sie zurückgeführt werden kann auf eine Klausel der spezielleren Klauselmenge[8], indem

- Variablen aus C_i in Konstante oder andere Terme überführt werden,
- Atome dem Klauselkörper von C_i hinzugefügt werden, oder
- der Klauselkörper von C_i im Hinblick auf das Hintergrundwissen B teilweise ausgewertet wird, d.h. ein Atom aus C_i wird mit einer Klausel aus B resolviert.

Wie sieht es bei den folgenden Klauseln aus, ist $C2 \geq_B C1$ bezüglich B?

B: $haustier(X) \leftarrow tier(X), im\_haus(X)$
$C1$:$kuscheltier(Z) \leftarrow flauschig(Z), tier(Z), im\_haus(Z)$
$C2$:$kuscheltier(Y) \leftarrow flauschig(Y), haustier(Y)$

Die Klauselköpfe werden durch $\sigma = \{Y/Z\}$ unifiziert. Wir erhalten als B, $\neg C1_{körper}\theta$ mit $\theta = \{Z/b\}$:

$haustier(X) \leftarrow tier(X), im\_haus(X).$
$flauschig(b).tier(b).im\_haus(b).$

und folgern

$flauschig(b).haustier(b).$

Das ist gerade $\neg C2_{körper}\sigma\theta$.

Es gilt also $C2 \geq_B C1$ bezüglich B.

Das Verfahren von [Buntine, 1988] zählt nicht nur die Literale durch, sondern berücksichtigt, daß einige Literale (hier: $tier(Z)$ und $im\_haus(Z)$) durch das Hintergrundwissen verbunden sind mit einem anderen Literal (hier: $haustier(X)$). Die Faustregel ist jetzt mit den Substitutionen σ und θ und der Einbeziehung des Hintergrundwissens so formalisiert, daß sie Ergebnisse liefert, die unserer Intuition entsprechen. Ihre Operationalisierung verwendet die Resolution, um alle Literale herauszuschneiden, die sowieso durch B gegeben sind. Die Substitutionen sorgen dafür, daß keine unzusammenhängenden Objekte einbezogen werden. Daher ist $C1$ auch genereller als $C3$:

[7] Es sei noch einmal daran erinnert, daß in diesem Skript der Körper einer Hornklauseln die Disjunktion der negativen Literale der Klausel ist. Die Negation des Körpers ist also eine Konjunktion von positiven Literalen. Dies wird in der Literatur manchmal auch anders definiert.

[8] Die Generalisierungsbeziehung kann auf Klauselmengen erweitert werden: es werden dann alle Klauseln der generelleren Menge mit denen der spezielleren Menge verglichen.

$C3 : kuscheltier(Y) \leftarrow flauschig(Y), tier(Y), im\_haus(Y), zahm(Y)$
Denn aus
$flauschig(b).tier(b).im\_haus(b).zahm(b).$ folgt logisch
$flauschig(b).tier(b).im\_haus(b).$ und dies ist
$\neg C1_{körper}\sigma\theta$ mit $\sigma = \{Z/Y\}$ und $\theta = \{Y/b\}$.

Betrachten wir nun wieder unser Beispiel mit $member/2$. An diesem Beispiel kann man ein Problem bei der Benutzung der generalisierten θ-Subsumtion erkennen.

Theorem 49: Unentscheidbarkeit der generalisierten θ-Subsumtion. *Die generalisierte θ-Subsumtion von Hornklauseln ist unentscheidbar.*

Anstelle des Beweises wird hier an einem Beispiel das Problem der generalisierten θ-Subsumtion deutlich gemacht.

Es sei $C2$ als ein Beispiel gegeben, für das eine Generalisierung gefunden werden soll:
$C2 : member(2, [1, 2]).$
Und bestehe das Hintergrundwissen nur aus dem Basisfall von $member/2$:
$B : member(A, [A \mid B]).$
Als Generalisierung von $C2$ unter θ-Subsumtion ist zum Beispiel $C1_1$ eine plausible Erklärung des Beispiels:
$C1_1 : member(A, [B, A]).$
Unter generalisierter Subsumtion kann dagegen das Hintergrundwissen bei der Generalisierung miteinbezogen werden. Mögliche Generalisierungen sind unter generalisierter Subsumtion:

$$C1_2 : member(2, [1, 2]) \leftarrow member(2, [2]). \qquad \text{oder}$$
$$C1_3 : member(A, [B \mid C]) \leftarrow member(A, [C]).$$

Bemerkenswerterweise ist unter θ-Subsumtion $C1_2$ eine Spezialisierung von $C2$, unter generalisierter Subsumtion sind die beiden Klauseln jedoch äquivalent. $C1_3$ beschreibt den Rekursionsfall von $member/2$. Wird $C1_3$ in das Logikprogramm B aufgenommen, enthält es eine vollständige und korrekte Definition für dieses Prädikat. Leider sind aber auch alle folgenden $C1_i$ gemäß der generalisierten Subsumtion Generalisierungen von $C2$:

$C1_4 : member(2, [1, 2]) \leftarrow member(2, [2]), member(2, [2, 2]).$
$C1_5 : member(2, [1, 2]) \leftarrow member(2, [2]), member(2, [2, 3, 4]).$
$C_6 : member(2, [1, 2]) \leftarrow member(2, [2]), member(2, [2, 2]), member(2, [2, 2, 2]).$
$\vdots$

Schließlich kann das Hintergrundwissen beliebig häufig verwendet werden und eine Konjunktion gilt, wenn man jedes Literal einzeln ableiten kann! Es kann also unendlich viele Generalisierungen geben[9]. Weil es im allgemeinen unendlich viele Generalisierungen einer Klausel gibt und zudem die Folgerung Teil der Definition ist (die logische Folgerung ist im

[9] Auch bei der θ-Subsumtion gibt es unendlich viele Generalisierungen, aber mit der Reduktion werden sie auf genau eine Klausel zusammengeführt. Eine analoge Reduktion für Buntines generalisierte Subsumtion ist vorstellbar, fehlt aber noch.

allgemeinen nur semientscheidbar), ist die generalisierte Subsumtion unentscheidbar. Es gibt aber eingeschränkte Sprachklassen, für die die generalisierte Subsumtion entscheidbar ist. Wenn zum Beispiel die Länge der Klauseln auf k Literale im Klauselkörper beschränkt ist, können nicht mehr unendlich viele Generalisierungen erzeugt werden.

5.2 Zuverlässige Generalisierungsoperatoren

Die Generalisierungsbeziehung zwischen Klauseln strukturiert uns den Hypothesenraum, in dem wir eine Lösung für das Begriffslernproblem suchen. Wenn man die Generalisierungsbeziehung so formalisieren kann, daß es für zwei Klauseln genau eine Generalisierung (bzw. Spezialisierung) gibt, besteht das Lernen darin, Beispiele so lange zu generalisieren, bis alle positiven Beispiele bezüglich des Hintergrundwissens abgedeckt sind bzw. allgemeine Hypothesen so lange zu spezialisieren, daß keine negativen Beispiele mehr bezüglich des Hintergrundwissens abgedeckt sind. Die Generalisierungsbeziehung ordnet den Hypothesenraum so, daß der induktive Schluß als durch Beispiele gesteuerte Bewegung in diesem Verband von Hypothesen realisiert werden kann. Die Generalisierungsbeziehung ermöglicht obendrein das sichere Beschneiden des Hypothesenraums:

- Wenn beim Generalisieren bereits die Hypothese $C1$ ein negatives Beispiel abdeckt, so wird auch jede Generalisierung von $C1$ mindestens dieses negativen Beispiel abdecken – $C1$ braucht daher nicht weiter generalisiert zu werden.
- Wenn beim Spezialisieren bereits die Hypothese $C1$ ein positives Beispiel nicht abdeckt, so wird auch jede Spezialisierung von $C1$ mindestens dies positive Beispiel nicht abdecken – $C1$ braucht daher nicht weiter spezialisiert zu werden.

Zuverlässige Generalisierungsoperatoren sind solche, die als Ergebnis die speziellste Generalisierung oder die generellste Spezialisierung abliefern. Sie können dem Benutzer also Zusicherungen über das Lernergebnis machen. Im Gegensatz dazu gibt es auch heuristische Verfahren, die irgendeine Lösung des Begriffslernproblems abliefern. In Abschnitt 5.4 lernen wir ein solches heuristisches Verfahren kennen.

5.2.1 Speziellste Generalisierung unter θ-Subsumtion

Gordon Plotkin stellt ein Verfahren vor, wie in der uneingeschränkten Prädikatenlogik erster Stufe die speziellste Generalisierung (*least general generalization* – lgg oder *most specific generalization* – MSG) gefunden werden kann [Plotkin, 1970]. Dabei stellt er zunächst ein Verfahren vor, das für Literale und Terme (zusammengefaßt unter der Bezeichnung *Wort*) eine speziellste Generalisierung findet.

Definition 50: Genereller-als Beziehung von Wörtern unter θ-Subsumtion. *Wort* $W1$ *ist genereller als Wort* $W1$ *unter* θ*-Subsumtion* ($W1 \geq_\theta W2$)*, wenn*

$$W1\sigma = W2$$

Zum Beispiel ist $p(X, X, g(g(Y))) \geq_\theta p(l(3), l(3), f(g(X)))$, mit $\sigma = \{X/l(3), Y/X\}$. Wieder ist $\geq_\theta$ (wie schon bei Lernen als Suche besprochen) eine Halbordnung (also reflexiv und transitiv).

Definition 51: Speziellste Generalisierung von Wörtern unter θ-Subsumtion. *Für eine Menge von Wörtern* K *ist* W *die speziellste Generalisierung, gdw.:*

- *Für jedes Wort V aus K gilt $W \geq V$ und*
- *wenn für jedes V aus K gilt $W1 \geq V$, dann gilt auch $W1 \geq W$.*

Mit der zweiten Bedingung wird gewährleistet, daß es sich bei W um die speziellste Generalisierung handelt. Wenn obendrein $W \geq W1$ gilt, dann sind W1 und W äquivalent.

Im ersten Schritt führt Plotkins Verfahren eine sog. Selektion durch.

Definition 52: Selektion. *Die Selektion sucht Wörter aus, die dasselbe Prädikatensymbol mit demselben Vorzeichen (negiert oder nicht negiert) haben.*

Dann muß er eine Substitution für die Terme finden, die an derselben Stelle des Prädikats auftreten, aber verschieden sind. Er wählt eine sonst nicht vorkommende Variable, setzt sie für die Terme ein und schreibt die Substitutionen in die Substitutionslisten der Wörter. Dies wird für alle verschiedenen Terme an selber Argumentstelle so gemacht. Wenn es keine solchen Terme (mehr) gibt, so ist das durch die Substitutionen entstandene Wort die speziellste Generalisierung der Wörter.

Beispiel:

$V1 : p(f(a(), g(Y)), X, g(Y))$
$V2 : p(h(a(), g(X)), X, g(X))$

Y und X haben nehmen dieselbe Argumentstelle ein und werden durch die neue Variable Z ersetzt. Das ergibt
$V1 : p(f(a(), g(Z)), X, g(Z))$
$V2 : p(h(a(), g(Z)), X, g(Z))$

und die Substitutionsliste für $V1$ ist $\{Z/Y\}$, die für $V2$ ist $\{Z/X\}$.

Als nächstes sind $f(a(), g(Z))$ und $h(a(), g(Z))$ verschiedene Terme an derselben Argumentstelle. Sie werden durch eine neue Variable U ersetzt.
$V1 = V2 = p(U, X, g(Z))$ mit den Substitutionslisten:
$\sigma 1 = \{U/f(a(), g(Y)), Z/Y\}$ für $V1$ und
$\sigma 2 = \{U/h(a(), g(X)), Z/X\}$ für $V2$.
$p(U, X, g(Z))$ ist die speziellste Generalisierung für $V1$ und $V2$.

Durch Anwendung der Substitutionen auf die Generalisierung erhält man wieder $V1$ respektive $V2$ [10].

Um dieses Verfahren auf Klauseln übertragen zu können, müssen zunächst alle Literale miteinander kombiniert werden. So macht man aus einer Menge eine äquivalente Liste. Wenn etwa die Klausel $C1$ die Literale $L11$, $L12$, $L13$ enthält und die Klausel $C2$ die Literale $L21$, $L22$, $L23$, so erhält man die Listen

$C1 : [L11, L11, L11, L12, L12, L12, L13, L13, L13]$
$C2 : [L21, L22, L23, L21, L22, L23, L21, L22, L23]$

Diese Listen können nun einfach elementweise gegeneinander abgeglichen werden. Dabei kann man alle übereinander stehenden Literalpaare streichen, die nicht dasselbe Prädikatssymbol haben. Für die resultierenden Listen findet Plotkin Generalisierungen mit Hilfe der θ-Subsumtion.

[10] Statt beim Einfügen einer Ersetzung in die Substitutionsliste die bisherigen Substitutionen auszuführen, kann man auf die Ersetzung in eingebetteten Termen verzichten wie es die Antiunifikation in Prolog tut.

Plotkins Übertragung des Verfahrens für Wörter auf Klauseln ist einfach: er formt Klauseln so um, daß neue Funktionssymbole anstelle der Prädikatssymbole stehen, kann die Prädikatensymbole dann weglassen und verfährt mit den Funktionen wie oben beschrieben. Man behandelt die Menge von Literalen einer Klausel einfach als Terme eines Prädikats!

Algorithmus 53: LGG (nach Plotkin). .

- Die Menge der Literale in den Klauseln werden mit zwei Indizes versehen, eines für die Klausel (i oder j), eines für die Literale selbst $(1, .., l, .., n)$. Jedes Literal L_{jl} hat ja die Form $(\pm)p_l(t_1, \ldots, t_k)$.
- Bei n Literalen einer Klausel wird ein neues, n-stelliges Prädikat q konstruiert, das für jedes k-stellige Prädikat der Klausel eine neue k-stellige Funktion als Argument bekommt:

$$q(f_1(t_{j11}, .., t_{jk1}), \ldots, f_n(t_{j1n}, \ldots, t_{jkn}))$$

- Damit erhält man für zwei Klauseln zwei solche q-Literale, deren Terme generalisiert werden. Das Ergebnis hat dann die Form:

$$q(f_1(u_{11}, .., u_{k1}), \ldots, f_n(u_{1n}, \ldots, u_{kn}))$$

- Dann ist $\{(\pm)p_l(u_{11}, .., u_{k1}), \ldots, (\pm)p_n(u_{1n}, \ldots, u_{kn})\}$ die speziellste Generalisierung der zwei Klauseln.

Wir haben gesehen, daß die Listen groß werden. Die Reduktion entfernt redundante (siehe Def. 45) Literale aus Klauseln. Alle solche L aus einer Klausel E werden entfernt, für die es eine Substitution gibt, so daß $E \setminus \{L\} \supseteq E\sigma$. Auch ohne L umfaßt E immer noch $E\sigma$. Da aber die θ-Subsumtion, die für jedes Literal bei jedem Test auf Redundanz ausgeführt wird, exponentiell ist, ist die Reduktion exponentiell.

Theorem 54: Eigenschaften des LGG. *Der lgg-Algorithmus ist kommutativ, assoziativ und idempotent:*

$$lgg(e_1, e_2) = lgg(e_2, e_1)$$
$$lgg(lgg(e_1, e_2), e_3) = lgg(e_1, lgg(e_2, e_3))$$
$$lgg(e, e) = e$$

Der lgg-Algorithmus ist also reihenfolgeunabhängig:

$$lgg(C_1, C_2, ..., C_n) = lgg(...lgg(lgg(C_1, C_2), C_3), ...C_n)$$

Das Lernergebnis ist eindeutig, d.h. die lggs sind äquivalent. Die lggs bilden einen Verband: in einem Verband der Äquivalenzklassen von Klauseln kann für zwei Klauseln eindeutig ihr Supremum bestimmt werden – es ist ihr lgg.

Theorem 55: Aufwand zur Berechnung des LGG. *Der Aufwand des Algorithmus sowie die Länge der speziellsten Generalisierungen ist linear zur Anzahl der Literale mit gleichem Prädikatensymbol in den Klauseln. Hat die längste Klausel m Literale und gibt es in E^+ n Klauseln, so kann es höchstens m^n Selektionen (gleiche Prädikatensymbole) geben und nur diese werden ja bearbeitet.*

Hat die längste Klausel m Literale und gibt es in E^+ n Klauseln, so kann es höchstens m^n Selektionen (gleiche Prädikatensymbole) geben und nur diese werden ja bearbeitet.

Diese Formalisierung hat zwei Schwächen: sie berücksichtigt kein Hintergrundwissen und sie ist ineffizient. Zur Einbeziehung von Hintergrundwissen hat Wray Buntine eine generalisierte θ-Subsumtion eingeführt, mit der die Induktion einer Klausel aus einer anderen Klausel bezüglich einer Theorie beschrieben werden kann [Buntine, 1988] (siehe Abschnitt 5.1.3). Ein Verfahren dazu ist die Induktion als inverse Resolution (Abschnitt 5.2.2). Auf die Effizienz kommen wir in Abschnitt 5.3 zu sprechen.

5.2.2 Induktion als inverse Resolution

Ein naheliegender Gedanke ist es, die Induktion als umgekehrte Deduktion aufzufassen. Die Resolution hat sich als operationales Verfahren für die Deduktion bewährt. Wäre eine inverse Resolution dann vielleicht ein operationales Verfahren für die Induktion? Diese Frage untersuchten zeitgleich Rüdiger Wirth für die Vervollständigung einer Menge von Syntaxregeln für ein natürlichsprachliches System [Wirth, 1989] und Stephen Muggleton und Wray Buntine [Muggleton und Buntine, 1988].

Eine kurze Rekapitulation der Resolution:

Definition 56: Resolution von Klauseln. *Seien $C1$ und $C2$ zwei Klauseln, $\neg R1$ ein negatives Literal aus $C1$ und $R2$ ein positives Literal aus $C2$. Wenn es einen generellsten Unifikator σ gibt mit $R1\sigma = R2\sigma$, dann sind $R1$ und $R2$ Resolutionsliterale und*

$$C = (C1 \setminus \neg R1)\sigma \cup (C2 \setminus R2)\sigma$$

der Resolvent.

Ein Beispiel zeigt die Abbildung 8.

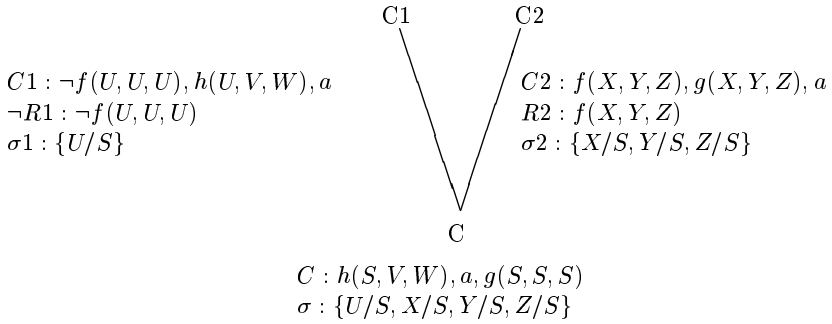


Abbildung8. Resolutionsschritt

Mit diesen Ausdrücken können wir die **inverse Resolution** als Verfahren angeben:

Definition 57: Inverse Resolution. *Sei das folgende gegeben:*

- *ein Resolutionsliteral $C1$*

– *ein Resolvent C*

Ziel der inversen Resolution ist es, Resolutionsliterale $C2$ zu $C1$ und C zu konstruieren.

Dabei gibt es zwei Probleme: die Umkehrung der Substitution und die Entscheidung, welcher nicht resolvierte Rest auf beiden Seiten gleichermaßen vorhanden ist. So ist im Beispiel nicht festzustellen, wenn wir nur $C1$ und C kennen, ob h und a auch in $C2$ vorkommen, oder nur eines von beiden. Wenn es keinen nicht-resolvierten Rest gibt, so ist $C2$:

$$(C \setminus C1\sigma1 \cup \neg R1\sigma1)\sigma^{-1}$$

Dabei ist σ^{-1} die inverse Substitution. Rüdiger Wirth nimmt als Umkehrung der Substitution einfach neue Variablen für die verschiedenen Argumentstellen [Wirth, 1989].

Im allgemeinen Fall, wenn es einen nicht-resolvierten Rest gibt, so muß die Potenzmenge dieses Restes, $pot(C1 \setminus \neg R1)$, gebildet werden. Damit gibt es im allgemeinen Fall bei n Literalen in $C1$ schlimmstenfalls $2^n \Leftrightarrow 1$ Lösungen für $C2$. In unserem Beispiel sind die möglichen Lösungen:

$g(X1, X2, X3), f(X4, X5, X6)$ oder
$g(X1, X2, X3), f(X4, X5, X6), a$ oder
$g(X1, X2, X3), f(X4, X5, X6), h(S, V, W)$ oder
$g(X1, X2, X3), f(X4, X5, X6), h(S, V, W), a$

Die inverse Resolution ist also ein exponentielles Verfahren. Außerdem erfordert es die Vorgabe des Resolventen, was in der Praxis selten möglich ist.

Stephen Muggleton verwendet die speziellste inverse Substitution $\sigma^{-1} = \emptyset$, um so die speziellste inverse Resolution zu erhalten. Für ein Beispiel e und Hintergrundwissen B, wobei LE und LB auf Fakten beschränkt sind, konstruiert sein System GOLEM mit einer Folge von speziellsten inversen Resolutionen einen speziellsten inversen linearen Ableitungsbaum [Muggleton, 1991]. Damit konstruiert GOLEM die Klausel e_{neu}, die eingegeben werden müßte, um mit ihr und B das Beispiel e abzuleiten. Um die speziellste Generalisierung zweier Klauseln, $e1$ und $e2$, bezüglich B zu erhalten, erstellt GOLEM zunächst für jede der beiden Klauseln bezüglich B einen speziellsten inversen linearen Ableitungsbaum. Auf das Ergebnis, $e1_{neu}$ und $e2_{neu}$, wendet er Plotkins Verfahren an.

Die inverse Resolution dient hier also dazu, das relevante Hintergrundwissen in die Beispiele hineinzurechnen, so daß dann Plotkins Verfahren als Generalisierung ohne Hintergrundwissen angewandt werden kann. Liegen Hintergrundwissen und Beispiele in Form von Fakten vor, können die Beispiele auch auf folgende Weise um das Hintergrundwissen erweitert werden:

$\forall e \in E : e_{neu} = e \leftarrow K$, wobei K die Konjunktion aller Fakten aus B ist.

Natürlich kann man K auch auf solche Fakten beschränken, die einen Term mit e gemeinsam haben, so daß nicht irrelevantes Hintergrundwissen e_{neu} unnötig verlängert. Auch kann K so gewählt werden, daß die Klausel $e \leftarrow K$ einer der Sprachbeschränkungen wie sie in Abschnitt 5.3 beschrieben sind, genügt.

5.3 Lernbarkeit in Prädikatenlogik

Um das Lernproblem handhabbar zu machen, werden Beschränkungen der Prädikatenlogik für die Repräsentation der Beispiele (LE), die des Hintergrundwissens (LB) und die der Hypothesen (LH) eingeführt. Hornklauseln werden noch weiter beschränkt.

Definition 58: Generative Klausel. *Eine Klausel ist generativ, wenn jede Variable aus dem Klauselkopf auch im Klauselkörper vorkommt. In der Literatur zu deduktiven Datenbanken heißen diese Klauseln* bereichsbeschränkt.

Definition 59: Deterministische Klausel. *Eine Klausel ist bezüglich B deterministisch, wenn für jedes Literal im Klauselkörper die bereits durch Konstante substituierten Variablen und das Hintergrundwissen eine eindeutige Substitution für die restlichen Variablen des Literals festlegen. Gibt es mehr als eine Substitution, so ist die Klausel indeterministisch.*

Definition 60: Beschränkte Klausel. *Eine Klausel ist beschränkt (constrained), wenn jede Variable aus dem Klauselkörper auch im Klauselkopf vorkommt.*

Solche Beschränkungen erlauben die effiziente Lernbarkeit. Wenn LB auf eine feste höchste Stelligkeit der Prädikate eingeschränkt wird und LH aus generativen Klauseln besteht, so können alle Ableitungen in einer Zeit polynomiell zur Anzahl der Klauseln in B durchgeführt werden. Dies bedeutet allerdings noch nicht, daß sie auch polynomiell lernbar sind. Die kombinatorische Explosion des lgg kann an folgendem Beispiel deutlich gemacht werden.

$$\begin{aligned} E^+ = \{\{+(e_1),\ & \neg teil(e_1,a), \neg klein(a), \neg rot(a),\\ & \neg teil(e_1,b), \neg gross(b), \neg blau(b)\}\\ \{+(e_2),\ & \neg teil(e_2,c), \neg klein(c), \neg blau(c),\\ & \neg teil(e_2,d), \neg gross(d), \neg rot(d)\}\}\\ lgg(E^+) = \{+(E),\ & \neg teil(E,X_1), \neg klein(X_1),\\ & \neg teil(E,X_2), \neg rot(X_2),\\ & \neg teil(E,X_3), \neg gross(X_3),\\ & \neg teil(E,X_4), \neg blau(X_4)\} \end{aligned}$$

Die zwei Objekte, die in den n Beispielen vorkommen, können ja auf 2^n Arten aufeinander abgebildet werden. Jede Abbildung liefert eine neue existentielle Variable. Es geht also darum, die Anzahl der existentiellen Variablen zu beschränken.

Theorem 61: *Wenn beschränkte Klauseln als LH gewählt werden, so ist das Lernproblem wahrscheinlich annähernd korrekt lösbar [Page und Frisch, 1992].*

Diese Aussage verknüpft das logik-basierte Lernen mit der Theorie des wahrscheinlich annähernd korrekten Lernens (PAC-Lernen, siehe Kapitel 4).

Theorem 62: *Der Aufwand bei deterministischen Klauseln ist exponentiell in der Stelligkeit der Prädikate und der deterministischen Tiefe von Termen. Damit sind sie nicht PAC-lernbar.*

Was ist nun die **deterministische Tiefe** eines Terms?

Definition 63: Deterministische Tiefe eines Terms. *Betrachten wir hierzu zunächt die deterministische Verbundenheit von Termen mit dem Klauselkopf.*

Ein Term aus dem Klauselkopf K ist mit einer Kette der Länge 0 deterministisch verbunden.

Für den Klauselkörper nehmen wir an, daß die Literale nach ihrer Verbundenheit mit dem Klauselkopf geordnet sind: $\{\neg L_1, ..., \neg L_m, \neg L_{m+1}, ..., \neg L_n\}$. *Ein Term t aus* L_{m+1} *ist genau dann durch eine deterministische Kette der Länge* $d+1$ *verbunden, wenn*

- *alle Terme im Klauselkopf und in* $\{\neg L_1, ..., \neg L_m\}$ *verbunden sind durch deterministische Ketten, die höchstens d lang sind, und*
- *es für jede Substitution* θ*, die K mit einem Beispiel und die ersten Literale mit dem Hintergrundwissen unifiziert – also* $K\theta \in E^+$ *und* $\{\{L_1\}, ..., \{L_m\}\}\theta \subseteq B$ *– genau eine eindeutige Substitution* σ *gibt, so daß* $L_{m+1}\theta\sigma \in B$.

Die deterministische Tiefe eines Terms ist die minimale Länge der deterministische verbindenden Ketten.

Nimmt man ein konstantes i für die maximale deterministische Tiefe und ein konstantes j für die maximale Stelligkeit der Prädikate, erhält man ij-deterministische Klauseln. Und diese sind polynomiell lernbar. Beschränkte Klauseln sind ein Spezialfall von ij-deterministischen Klauseln ($i = 0$).

Definition 64: ij-deterministische Klausel. *Eine Klausel ist* **ij-deterministisch**, *wenn sie*

- *deterministisch ist und*
- *bezüglich B die maximale (deterministische) Tiefe ihrer Terme i ist,*
- *die maximale Stelligkeit ihrer Prädikate j ist.*

Theorem 65: Lernbarkeit von ij-deterministische Klauseln. *ij-deterministische Klauseln können in polynomieller Zeit gelernt werden, wenn LE variablenfreie Fakten und LB variablenfreie Klauseln sind [Muggleton und Feng, 1992].*

Im schlimmsten Falle enthält das Lernergebnis $(Crt)^{ij}$ Literale, wobei C die Anzahl der Prädikate im Hintergrundwissen B, r die höchste Stelligkeit eines Prädikats in B und t die Anzahl von Selektionen im $lgg(E^+)$ ist. Ihr System GOLEM nimmt feste i und j als Parameter und ist dann polynomiell in der Größe von Hintergrundwissen und Beispielen beschränkt. Die Verbindung zum wahrscheinlich annähernd korrekten Lernen haben Saso Dzeroski, Stephen Muggleton und Stuart Russell hergestellt

Theorem 66: PAC-Lernbarkeit von ij-deterministische Klauseln. *Für einfache Verteilungen der positiven und negativen Beispiele sind k funktionsfreie, nicht rekursive ij-deterministische Klauseln wahrscheinlich annähernd korrekt lernbar [Džeroski et al., 1992].*

LH ist dann eigentlich nicht ausdrucksstärker als Aussagenlogik. Aber in vielen Fällen ist die Repräsentation dann leichter verständlich.

Eine weitere Beschränkung von Hornklauseln sind k-lokale Klauseln. Sie müssen aber noch weiter differenziert werden, um die Grenze der Lernbarkeit festzulegen.

Definition 67: k-lokale Hornklauseln. *Sei D eine Hornklausel* $D : D_0 \leftarrow D_{DET}, D_{NONDET}$ *und vars eine Funktion, die die Menge der Variablen einer Klausel berechnet.* $LOC_i \subseteq D_{NONDET}$ *ist ein lokaler Teil von D genau dann, wenn* $(vars(LOC_i) \backslash vars(\{D_0, D_{DET}\})) \cap vars(D_{NONDET} \setminus LOC_i) = \emptyset$ *und es kein* $LOC_j \subset LOC_i$ *gibt, welches auch ein lokaler Teil von D ist. Ein lokaler Teil* LOC_i *ist für eine Konstante k:*

k-vlokaler Teil *gdw.* $k \geq |\ (vars(LOC) \setminus vars(\{D_0, D_{DET}\})\ |$,

k-llokaler Teil *gdw.* $k \geq |\ LOC\ |$.

Eine Hornklausel D heißt k-llokal *bzw.* k-vlokal *genau dann, wenn jeder lokale Teil von D ein k-llokaler bzw. k-vlokaler Teil ist.*

Es sind gerade die k-llokalen Klauseln, die polynomiell lernbar sind. Die k-vlokalen Klauseln sind es nicht.

Den positiven Ergebnissen stehen negative gegenüber. Jörg-Uwe Kietz zeigte, daß die Komplexität des Lernproblems bereits bei nicht tiefenbeschränkten maximal 2-stelligen deterministischen Hornklauseln – d.h. LH enthält n2-deterministische Klauseln – mindestens in PSPACE liegt [Kietz, 1993]. Bei indeterministischen Klauseln nützt selbst die Beschränkung der Tiefe auf 1 und die der Stelligkeit auf 2 nichts: das Problem der Erfüllbarkeit einer Formelmenge (SAT) kann als Lernen von 12-indeterministischen Klauseln formuliert werden. Damit ist das Lernen von 12-indeterministischen Klauseln NP-schwierig [Kietz, 1993]. 12-indeterministische Klauseln sind nicht wahrscheinlich annähernd korrekt lernbar, es sei denn $RP = NP$. Bereits 1-vlokale Klauseln zu lernen, ist NP-schwierig. Um nun ein Gefühl für die Forschungen im Bereich der induktiven logischen Programmierung zu bekommen, wird in Abschnitt 5.3.1 der Beweis von Jörg-Uwe Kietz wiedergegeben, der die Schwierigkeit des Lernens für deterministische Hornklauseln zeigt. Solche negativen Beweise sind insofern praktisch, als man zwei Dinge damit tun kann:

- Man grenzt gerade die schwierigen Stellen, die für den Beweis verwendet wurden, aus den verwendeten Sprachen LE, LB, LH aus. Der Beweis zeigt, welche Sprachbeschränkung gewählt werden sollte.

- Man entwirft einen Algorithmus, der bei einer gegebenen Lernaufgabe die polynomiellen Teile erkennt und in polynomieller Zeit löst. Die exponentiellen Teile werden erkannt, der Benutzer wird gewarnt und kann dann doch den Algorithmus laufen lassen – vielleicht ist ja auch die exponentielle Zeit noch zu verkraften!

5.3.1 Lernen von deterministischen Hornklauseln ist PSPACE-schwierig — J.-U. Kietz

Die Folgerung ist für nicht-rekursive deterministische Hornklauseln mit polynomialem Aufwand entscheidbar[11]. Wie sieht es mit der Lernbarkeit deterministischer Hornklauseln aus? Dabei werden wir zeigen, daß das Konsistenzproblem $K(\vdash_\theta, DL^1_{DET}, DL_{GDET}, \emptyset)$ PSPACE-schwierig ist. Dies ist das Problem, für deterministische, funktionsfreie und variablenfreie Hornklauseln (DL_{GDET}) als Beispielsprache LE ohne Hintergrundwissen ($\emptyset$) unter Verwendung der θ-Subsumtion als Ableitungsrelation ($\vdash_\theta$) zu entscheiden, ob es eine Hypothese in Form einer deterministischen, funktionsfreien Hornklausel (DL^1_{DET}) gibt, die konsistent, vollständig und korrekt ist[12].

Um dieses zu beweisen, werden wir uns mit dem Zusammenhang zwischen Hornklauseln und endlichen Automaten beschäftigen, und das PSPACE-vollständige Problem des Schnitts deterministischer endlicher Automaten (DFA Intersection [Garey und Johnson, 1979]) auf das Konsistenzproblem zurückführen.

[11] Dieser Abschnitt ist ein nur geringfügig modifizierter Auszug aus der Arbeit [Kietz, 1996].

[12] DL: Datalog, i.e. funktionsfreie Formeln; DET: deterministisch; G: Grund-, i.e. variablenfrei; das Superskript bezeichnet die erlaubte Anzahl von Klauseln in einer Hypothese.

Definition 68: DFA Intersection Problem. *Gegeben ein Alphabet Σ und die DFAs $A_1, \ldots, A_n$ mit Eingabealphabet Σ. Gibt es ein Wort $w \in \Sigma^*$, das von allen DFAs $A_i, 1 \leq i \leq n$ erkannt wird?*

Ein DFA A wird durch einen 5-Tuple $(Q, \Sigma, \delta, s, F)$ beschrieben, wobei Q eine Menge von Zuständen ist, Σ ist das Eingabealphabet, $s \in Q$ ist der Startzustand, $F \subseteq Q$ ist eine Menge von Zielzuständen, und δ ist die Zustandsübergangsfunktion $Q \times \Sigma \Leftrightarrow\!\!\!\rightarrow Q$. Die Zustandsübergangsfunktion δ wird wie folgt auf Wörter aus Σ^* erweitert: $\hat{\delta}(q, \epsilon) = q$, $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$ wobei ϵ das leere Wort ist, $a \in \Sigma$, $w \in \Sigma^*$. Ein DFA erkennt ein Wort $x \in \Sigma^*$, gdw. $\hat{\delta}(s, x) \in F$. Die Menge der Wörter (die Sprache), die von A erkannt wird, wird mit $S(A)$ bezeichnet

Nach diesen Grundlagen über DFAs können wir jetzt eine Kodierung von DFAs und Wörtern als 2-deterministische Hornklauseln geben, die die Basis unseres Beweises bildet.

Definition 69: Kodierung von DFA's und Wörtern als Hornklauseln. *Sei Γ eine Funktion von $\Sigma^* \cup (Q, \Sigma, \delta, s, F)$ nach Hornklauseln, die wie folgt definiert ist:*
$\Gamma_{word}(x_1 x_2 \ldots x_n) = h(Q_0) \leftarrow x_1(Q_0, Q_1), x_2(Q_1, Q_2), \ldots, x_n(Q_{n-1}, Q_n), f(Q_n)$
$\Gamma_{DFA}(A) = \{h(s)\} \cup \{\neg x(q, p) | x \in \Sigma, p, q \in Q, \delta(q, x) = p\} \cup \{\neg f(q) | q \in F\}$

Es ist leicht zu sehen, daß DFAs auf variablenfreie Hornklauseln abgebildet werden, das heißt: sie werden auf Elemente unserer Beispielsprache LE abgebildet. Wörter werden auf Hornklauseln mit Variablen abgebildet, d.h. auf Elemente unseres Hypothesenraumes LH. Die Nützlichkeit dieser Abbildung wird durch die folgenden Lemmata belegt.

Lemma 70: *$\hat{\delta}(s, x) \in F \Leftrightarrow \Gamma(w) \vdash_\theta \Gamma(A)$, d.h. ein Wort w wird durch einen DFA A erkannt, gdw. das kodierte Wort den kodierten DFA θ-subsumiert.*

Beweis. $\hat{\delta}(s, w) \in F \Rightarrow \Gamma(w) \vdash_\theta \Gamma(A)$
Sei $w = x_1 \ldots x_m$, Dann gibt es eine Reihe von Zuständen (die Berechnung von A bezüglich w) $q_0, \ldots, q_m$, so daß $q_0 = s, q_m \in F$, und $\delta(q_{i-1}, x_i) = q_i$. Sei θ be $\{Q_i/q_i\}$, für $0 \leq i \leq m$, aus der Definition von Γ folgt, daß $\Gamma(w)\theta \subseteq \Gamma(A)$.
$\Gamma(w) \vdash_\theta \Gamma(A) \Rightarrow \hat{\delta}(s, w) \in F$
Sei θ be $\{Q_i/q_i\}$, for $0 \leq i \leq m$, aus der Definition von Γ folgt, daß $q_0 = s, q_m \in F$, und daß $\delta(q_{i-1}, x_i) = q_i$, d.h. die Folge $q_0 \ldots, q_m$ ist eine w akzeptierende Berechnung von A. □

Wenn wir Γ für DFAs so verändern, daß die Zustände nicht auf Konstanten sondern auf Variablen abgebildet werden $\Gamma'_{DFA}(A) = \{h(S)\} \cup \{\neg x(Q, P) | x \in \Sigma, p, q \in Q, \delta(q, x) = p\} \cup \{\neg f(Q) | q \in F\}$ können wir dieses Lemma noch verstärken:

Lemma 71: *$\Gamma'(A_1) \vdash_\theta \Gamma'(A_2) \Rightarrow S(A1) \subseteq S(A2)$, d.h. wenn die Kodierung eines DFAs A_1 die Kodierung eines DFAs A_2 subsumiert, dann erkennt A_1 eine Teilsprache von A_2.*

Beweis. Das folgt direkt aus der Transitivität von $\vdash_\theta$ und dem Lemma 70. □

Die Umkehrung dieses Lemmas gilt natürlich nicht, da es selbst für die gleiche Sprache viele verschiedene erkennende Automaten gibt. Auch stellt nicht jede Hornklausel, die die Kodierung eines DFAs subsumiert, selbst wieder einen kodierten DFA dar.

Nachdem wir den Zusammenhang zwischen den Berechnungen von DFAs und und der Subsumtion der mit Γ erzeugten Hornklauseln geklärt haben, wollen wir nun ein Lernproblem finden, das das DFA intersection Problem löst.

Theorem 72: *Seien $A_1 \ldots, A_n$ mit dem Alphabet $\Sigma = \{x_1, \ldots, x_m\}$ DFAs, bei denen der Startzustand kein Zielzustand ist*[13]*. Es existiert ein nicht leeres Wort $x \in \Sigma^+$ daß von jedem der $A_1 \ldots, A_n$ akzeptiert wird, gdw. es eine Hornklausel gibt, die eine Lösung des Konsistenzproblems mit den folgenden Beispielen ist:*

$$
\begin{aligned}
E^+ &= \{\Gamma(A_i) | 1 \leq i \leq n\} \\
E^- &= \{\ h(s_1) \leftarrow \qquad\quad x_1(s_1,s_1), \ldots, x_m(s_1,s_1). \\
&\qquad h(s_2) \leftarrow f(s_2),\ x_1(s_2,q_1), \ldots, x_m(s_2,q_1), \\
&\qquad\qquad\qquad\quad x_1(q_1,q_1), \ldots, x_m(q_1,q_1), \\
&\qquad\qquad\quad f(q_2),\ x_1(q_2,q_2), \ldots, x_m(q_2,q_2), \\
&\qquad\qquad\qquad\quad x_1(q_2,s_2), \ldots, x_m(q_2,s_2), \\
&\qquad\qquad\qquad\quad x_1(q_2,q_1), \ldots, x_m(q_2,q_1). \\
&\quad\ \}
\end{aligned}
$$

Wenn solch eine mit den Beispielen konsistente Hornklausel existiert, dann existiert auch eine mit den Beispielen konsistente verbundene deterministische Hornklausel.

Beweis. Wir werden den Beweis in drei Teilen führen, die zusammen unser Theorem ergeben.

(I) Die Kodierung eines von allen DFAs akzeptierten Wortes ist eine mit den Beispielen konsistente deterministische Hornklausel.

(II) Jede Hornklausel, die die Kodierung aller DFAs $A_1, \ldots, A_n$ subsumiert, besteht nur aus Literalen der Form $h(X_h)$, $\neg f(X_f)$ und $\neg x(X_1, X_2), x \in \Sigma$. Alle kodierten Wörter die diese Klausel subsumieren, werden von allen Automaten erkannt[14].

(III) Jede Hornklausel, die nur aus Literalen der Form $h(X_h)$, $\neg f(X_f)$ und $\neg x(X_1, X_2), x \in \Sigma$ besteht, subsumiert keins der negativen Beispiele, gdw. es ein nicht leeres Wort gibt, dessen Kodierung diese Hornklausel subsumiert[15].

(I): Sei w das von $A_1, \ldots, A_n$ akzeptierte Wort. Durch Lemma 70 wissen wir, das $\Gamma(w)$ alle $\Gamma(A_i) \in E^+$ subsumiert. Da w nicht das leere Wort ist, ist $\Gamma(w)$ von der Form $h(Q_0) \leftarrow x_1(Q_0, Q_1), \ldots, x_n(Q_{n-1}, Q_n), f(Q_n)$, mit $x_1, \ldots, x_n \in \Sigma$. $\Gamma(w)$ subsumiert nicht das erste negative Beispiel, da e_1 kein f-Literal enthält. $\Gamma(w)$ subsumiert nicht das zweite negative Beispiel, da $h(Q_0)$ nur $h(s_2)$ subsumiert, dementsprechend kann $x_1(Q_0, Q_1)$ nur $x_1(s_2, q_1)$ subsumieren und alle weiteren $\neg x_i(X_3, X_4)$ Literale von $\Gamma(w)$ müssen auf $\neg x_i(q_1, q_1)$ Literale von e_2 abgebildet werden. Damit enthält θ die Substitution Q_n/q_1 und das Literal $f(Q_n)\theta = f(q_1)$ aus $\Gamma(w)$ kann nicht abgebildet werden.

(II): Sei C eine Hornklausel, die $\Gamma(A_1), \ldots, \Gamma(A_n)$ subsumiert. Wenn sie ein Literal enthalten würde, das nicht von der Form $h(X_h)$, $\neg f(X_f)$ und $\neg x(X_1, X_2), x \in \Sigma$ ist, würde dieses Literal diese Subsumtionen verhindern. Da C alle $\Gamma(A_i)$ subsumiert, subsumieren alle Klauseln, die C subsumieren, auch alle $\Gamma(A_1), \ldots, \Gamma(A_n)$. Dies gilt insbesondere auch

[13] Dies verändert die Komplexität der Aufgabe nicht, da wir jeden Automaten durch Einführung eines neuen Zustands als Startzustand, der die gleiche Transitionsfunktion wie der ursprünglich Startzustand erhält, in einen Automaten umwandeln können, der bis auf das leere Wort dieselbe Sprache erkennt.

[14] Jede Hornklausel, die nur aus $h(X_h)$, $\neg f(X_f)$ und $\neg x(X_1, X_2), x \in \Sigma$ Literalen besteht, kann als Kodierung eines NFAs (nicht-deterministischen endlichen Automaten) interpretiert werden, der die Wörter erkennt, die die Hornklausel subsumieren. Diese Hornklausel kann deshalb als Kodierung eines NFAs aufgefaßt werden, der eine gemeinsame Teilsprache aller $A_1, \ldots, A_n$ erkennt.

[15] Sie repräsentiert damit einen NFA, der mindestens ein nicht leeres Wort akzeptiert.

für die kodierten Wörter und nach Lemma 70 werden diese Wörter auch von allen DFAs erkannt.

(III): Wir haben in (I) bereits gesehen, daß kein kodiertes Wort ein negatives Beispiel subsumiert und damit subsumiert ein kodiertes Wort auch keine Klausel, die ein negatives Beispiel subsumiert. Wir müssen also nur noch zeigen, daß jede Klausel, die kein negatives Beispiel subsumiert und aus $h(X_h)$, $\neg f(X_f)$ und $\neg x(X_1, X_2), x \in \Sigma$ Literalen besteht (wegen (II)), von einem kodierten (nicht leeren) Wort subsumiert wird, d.h. ein kodiertes (nicht leeres) Wort als Teilmenge enthält.

Sei C diese Hornklausel, die kein negatives Beispiel subsumiert. C muß genau ein $h(X_h)$ Literal enthalten, da es sonst keine Hornklausel wäre. Wenn C nur $h(X_h)$ und $\neg x(X_1, X_2), x \in \Sigma$ Literale enthalten würde, würde C das erste negative Beispiel mit $\theta = \{X_h/s_1, X_i/s_1\}$ subsumieren.

Wenn C nur $h(X_h)$ und $\neg f(X_f)$ als Literale enthalten würde, würde C das zweite negative Beispiel mit $\theta = \{X_h/s_2, X_f/s_2\}$ subsumieren. C muß also alle drei Formen von Literalen enthalten, und die Literale müssen verbunden sein, damit sie sich gegenseitig beeinflussen können. Nicht minimale Verbindungsketten brauchen wir nicht weiter zu betrachten, da alle daran beteiligten $\neg x(X_1, X_2), x \in \Sigma$ Literale entweder auf die $\neg x(q_1, q_1), x \in \Sigma$ oder $\neg x(q_2, q_2), x \in \Sigma$ Literale im zweiten negativen Beispiel zusammengefaltet werden können (es sei denn sie sind Bestandteil einer anderen minimalen Verbindungskette). Sie können daher nicht dabei helfen, das zweite negative Beispiel nicht zu subsumieren. Für die minimale Verbindung von $h(X_h)$ und $\neg f(X_f)$ gibt es zwei prinzipielle Möglichkeiten. Entweder ist $X_h = X_f$, dann verhindern diese Literale jedoch nicht, daß das zweite negative Beispiel subsumiert wird, oder es gibt eine minimale Verbindungskette von $\neg x_1(X_{11}, X_{21}), \ldots \neg x_n(X_{1n}, X_{2n}) | x_i \in \Sigma$ Literalen, so daß das erste $\neg x_1(X_{11}, X_{21})$ Literal die Variable X_h enthält ($X_{11} = X_h$ oder $X_{12} = X_h$), das n-te Literal $\neg x_n(X_{1n}, X_{2n})$, die Variable X_f, enthält ($X_{1n} = X_f$ oder $X_{2n} = X_f$), und für alle $1 \leq i < n$ gilt $\neg x_i(X_{1i}, X_{2i})$ und $\neg x_{i+1}(X_{1(i+1)}, X_{2(i+1)})$ das i-te und das $i+1$-te Literal haben eine Variable gemeinsam ($X_{1i} = X_{1(i+1)}$ oder $X_{1i} = X_{2(i+1)}$ oder $X_{2i} = X_{1(i+1)}$ oder $X_{2i} = X_{2(i+1)}$). Wenn sie mehr als eine Variable gemeinsam haben, ist eins der beiden Literale für die minimale Verbindung überflüssig. Auch wenn ein Literal dieselbe Variable mit dem vorherigen und dem nächsten Literal teilt, ist es für die minimale Verbindung überflüssig. Für jedes der $\neg x(X_1, X_2), x \in \Sigma$ Literale in der minimalen Verbindungskette gilt also: eine der beiden Variablen ist mit dem vorherigen Literal verbunden und die andere Variable mit dem nachfolgenden Literal. Wenn das erste $\neg x(X_1, X_2), x \in \Sigma$ Literal X_2 mit dem Kopf gemeinsam hat, und X_1 mit dem nachfolgenden Literal, dann kann es auf das $\neg x(q_2, s_2), x \in \Sigma$ Literal im zweiten negativen Beispiel abgebildet werden, und damit können alle nachfolgen Literale auf $\neg x(q_2, q_2), x \in \Sigma$ Literale und das $f(X_f)$ Literal auf das $f(q_2)$ Literal abgebildet werden, die Kette subsumiert damit das zweite negative Beispiel. Das erste $\neg x(X_1, X_2), x \in \Sigma$ Literal muß also X_1 mit dem Kopf gemeinsam haben, damit es die Subsumtion des negativen Beispiels verhindert. Dieselbe Argumentation gilt für alle nachfolgenden $\neg x(X_1, X_2), x \in \Sigma$ Literale, nur daß diese dann auf die entsprechenden $\neg x(q_2, q_1), x \in \Sigma$ Literale im zweiten negativen Beispiel abgebildet werden könen. Daraus folgt, daß die Verbindungskette die Form $h(X_h), \neg x_1(X_h, X_1), \neg x_2(X_1, X_2), \ldots, x_n(X_{n-1}, X_f), \neg f(X_f)$ haben muß, damit sie das zweite negative Beispiel nicht subsumiert. Dies ist aber genau die Kodierung eines Wortes. □

Wir haben bewiesen, daß das Konsistenzproblem für das Lernen einer deterministi-

schen Hornklausel aus variablenfreien Hornklauseln und ohne Hintergrundwissen nicht polyniomiell lösbar ist. Aus der nicht polynomialen Lösbarkeit des Konsistenzproblems folgt weiterhin, daß die entsprechenden Begriffslernprobleme (d.h. mit allen möglichen Präferenzordnungen) nicht polynomiell lösbar sind. Auch die entsprechenden Probleme der Wissensentdeckung (Abschnitt 5.5) sind nicht polynomiell lösbar.

5.4 Heuristische Beschränkung der Suche – FOIL

Ein Beispiel für die unvollständige Suche im Hypothesenraum ist das System FOIL [Quinlan, 1990]. Die Hypothesensprache LH umfaßt funktionsfreie Klauseln, deren Kopf aus genau einem positiven Literal besteht und deren Körper aus positiven oder negierten Literalen besteht. Terme müssen mit dem Klauselkopf verbunden sein. Beispiele werden in Form von variablen- und funktionsfreien Fakten gegeben. Wenn keine negativen Beispiele vorliegen, berechnet FOIL gemäß der *closed-world assumption* alle möglichen, nicht als positive Beispiele gegebenen Fakten als negative Beispiele. Das Hintergrundwissen liegt in Form von extensionalen Definitionen von Prädikaten in Form von variablen- und funktionsfreien Fakten vor. FOIL erzeugt die Hypothesen von der allgemeinsten zu den spezielleren. FOIL durchsucht den Hypothesenraum nur dort, wo ein Informationsgewinn erwartet wird. Dazu wird eine Heuristik verwendet, die auf der Entropie basiert.

FOIL erstellt aus den gegebenen Beispielen für ein k-stelliges Zielprädikat eine Menge von k-stelligen Tupeln, wobei jedes Tupel eine Wertebelegung der k Variablen angibt. Die Tupel sind als positive oder negative Beispiele klassifiziert. Die Tupelmenge ist vollständig, d.h. alle möglichen Wertebelegungen kommen entweder als positives oder als negatives Beispiel vor. Der Algorithmus besteht aus zwei Schleifen, einer äußeren und einer inneren. Die äußere Schleife findet für positive Beispiele bzw. Tupel T^+ eine Klausel, die sie abdeckt. Die abgedeckten positiven Beispiele werden aus der Tupelmenge entfernt und für die verbleibenden positiven Beispiele wird eine weitere Klausel gelernt. Es werden so viele Klauseln gelernt, daß insgesamt von der Menge der Klauseln alle positiven Beispiele abgedeckt werden.

Algorithmus 73: FOIL - äußere Schleife. Bis T^+ keine Tupel mehr enthält:

- finde eine Klausel, die positive Beispiele abdeckt,
- entferne alle Tupel aus T^+, die von dem Klauselkörper abgedeckt werden.

In der inneren Schleife wird der Körper einer Klausel durch Literale erweitert, die Klausel also spezialisiert. Ein zusätzliches Literal wird aufgrund des Informationsgewinns ausgewählt, den es für die Charakterisierung des Zielprädikats erbringt.

Algorithmus 74: FOIL - innere Schleife. Initialisiere T_i mit den Beispielen in Tupeldarstellung und setze $i = 1$. Solange T^- Tupel enthält:

- finde ein nützliches Literal L_i und füge es dem Klauselkörper hinzu,
- erstelle eine neue Tupelmenge T_{i+1}, die nur diejenigen Beispiele enthält, für die L_i gilt. Falls L_i eine neue Variable einführt, werden die Tupel um diese erweitert. Die Klassifikation der Tupel in T_{i+1} bleibt diejenige von T_i.
- setze i auf $i + 1$

Die Information, die man bei einer Tupelmenge T_i mit p_i positiven und n_i negativen Beispielen braucht, um zu entscheiden, ob ein Tupel positiv ist, ist

$$I(T_i) = \Leftrightarrow log_2 \frac{p_i}{p_i + n_i}$$

Ziehen wir den Informationsgehalt der neuen Tupelmenge von dem der alten Tupelmenge ab, so erhalten wir den Informationsgewinn durch das neu eingeführte Literal. Dieser Informationsgewinn wird durch die Anzahl der positiven Tupel, die durch das neue Literal abgedeckt werden, p_{i+1}, gewichtet. Der gewichtete Informationsgewinn durch ein neues Literal L_i bezogen auf die Tupelmenge T_i wird also durch die folgende Formel ausgedrückt:

$$Gain(L_i) = (p_{i+1} + n_{i+1})(I(T_i) \Leftrightarrow I(T_{i+1}))$$

Ein einfaches Beispiel ist das Lernen der Relation *tochter* aus den folgenden Beispielen, wobei vor dem Semikolon die positiven Tupel, nach dem Semikolon die negativen Tupel angegeben werden. Das Hintergrundwissen ist durch die Relationen *eltern* und *weibl* gegeben. Das Beispiel ist [Lavrač und Džeroski, 1994] entnommen.

tochter(<name>, <name>)	eltern(<name>, <name>)	weibl(<name>)
mary, ann	ann, mary	mary
eve, tom ;	ann, tom	ann
tom, ann	tom, eve	eve
eve, ann	tom, ian	

Die erste Menge T_1 enthält alle 4 Tupel, $p_1 = 2, n_1 = 2$. Ihr Informationsgehalt ist $\Leftrightarrow log_2 \frac{2}{4} = 1$. Die Klausel enthält nur das Zielprädikat $tochter(X, Y)$ als Klauselkopf. Wird nun *weibl* als erstes Literal in den Klauselkörper eingetragen, so enthält die neue Tupelmenge T_2 nicht mehr das negative Beispiel tom, ann, da $weibl(tom)$ nicht gilt. Damit ist die neue Tupelmenge geordneter: $I(T_2) = 0,58$. Alle positiven Beispiele bleiben abgedeckt. Der Informationsgewinn von *weibl* ist also:

$$Gain(weibl(X)) = 2(1 \Leftrightarrow 0,58) = 0,84$$

Wird nun noch $eltern(Y, X)$ als nächstes Literal hinzugefügt, so brauchen wir bei der resultierenden Tupelmenge $T3$ keine weitere Information für die Klassifikation: $I(T_3) = 0$. Der Informationsgewinn von *eltern* ist:

$$Gain(eltern(Y, X)) = 2(0,58 \Leftrightarrow 0) = 1,16$$

Die Klausel $tochter(X, Y) \leftarrow eltern(Y, X), weibl(X)$ deckt alle positiven Beispiele ab und kein negatives. Da in der Menge T_3 kein negatives Beispiel mehr enthalten ist, endet die innere Schleife. Damit ist der erste Schritt der äußeren Schleife abgeschlossen und es werden alle positiven Beispiele aus der Menge T_1 entfernt, die von der Klausel abgedeckt werden. Da dies alle positiven Beispiele sind, endet auch die äußere Schleife.

Da FOIL den Hypothesenraum gemäß der Heuristik des Informationsgewinns durchsucht, kann es eine richtige Lösung des Lernproblems verpassen. Dies ist der Fall, wenn eigentlich ein Literal mit geringem Informationsgewinn gewählt werden müßte, das eine neue Variable einführt, die für die Wahl geeigneter weiterer Literale nötig ist. In sehr vielen Fällen führt die Heuristik jedoch dazu, daß FOIL schnell die richtige Lösung findet.

5.5 Wissensentdeckung

In der induktiven logischen Programmierung wird zusätzlich zum Begriffslernen auch noch die schwierigere Lernaufgabe des Regellernens bzw. der Wissensentdeckung untersucht. Manchmal spricht man dann vom *non-monotonic setting of ILP*. Die formale Fassung der Lernaufgabe verwendet den logischen Begriff des **minimalen Modells**. Hier zur Erinnerung seine Definition:

Definition 75: Modell. *Eine Interpretation I ist ein* Modell *einer Theorie T, $\mathcal{M}(T)$, wenn sie für jede Aussage in T wahr ist.*

Definition 76: Minimales Modell. *Eine Interpretation I ist ein* minimales Modell *von T, geschrieben $\mathcal{M}^+(T)$, wenn I ein Modell von T ist und es keine Interpretation I' gibt, die auch ein Modell von T ist und $I' \subset I$.*

Definition 77: Wissensentdeckung $RL(\vdash, \succeq, LH, LE, LB)$. *Die Lernaufgabe* **Wissensentdeckung** *der induktiven logischen Programmierung ist:*
Gegeben:

- *Beobachtungen E in einer Sprache LE,*
- *Hintergrundwissen B in einer Sprache LB.*

Ziel: *Eine Menge von Hypothese H in einer Sprache LH, so daß gilt:*

- *Gültigkeit: $\mathcal{M}^+(B \cup E) \subseteq \mathcal{M}(H)$*
- *Notwendigkeit: für alle $h \in H$ gibt es mindestens ein $e \in E$ mit $B, E \setminus \{e\} \not\models e$ und $B, E \Leftrightarrow \{e\}, h \models e$*
- *Vollständigkeit: für alle $h \in LH$, die gültig und notwendig sind, gilt $H \models h$*
- *Minimalität: Es gibt keine echte Teilmenge G in H, die gültig und vollständig ist.*

Die Wissensentdeckung ist schwieriger als das Begriffslernen wegen der sehr starken Vollständigkeitsbedingung, die fordert, daß alle wahren und nicht redundanten Regeln in der Lösung vorhanden sind. Der Unterschied zwischen den beiden Lernaufgaben mag an einem einfachen Beispiel klar werden (Abbildung 9).

	Kunde		verheiratet	
Person	Einkommen	Kunde	Mann	Frau
Ann Smith	niedrig	ja	Jack Brown	Jane Brown
Joan Gray	hoch	ja	Bob Smith	Ann Smith
Mary Blythe	niedrig	nein		
Jane Brown	niedrig	ja		
Bob Smith	hoch	ja		
Jack Brown	hoch	ja		

Abbildung9. Datenbank mit zwei Tabellen

Ein Verfahren zur Wissensentdeckung findet vielleicht unter anderen die folgenden Regeln:

(i) $kunde\_ja(Partner) \leftarrow$
$verheiratet(Person, Partner), kunde\_ja(Person)$

(ii) $kunde\_ja(Partner) \leftarrow$
$verheiratet(Person, Partner), einkommen(Person, hoch)$

(iii) $kunde\_ja(Person) \leftarrow einkommen(Person, hoch)$

Diese Regeln können auch von einem Verfahren zum Begriffslernen gefunden werden, weil ja stets dasselbe Prädikat im Klauselkopf (in der Konklusion) steht. Die Wissensentdeckung findet auch über andere Prädikate etwas, z.B.:

$verheiratet(Partner, Person) \leftarrow$
$kunde\_ja(Person), einkommen(Person, niedrig)$

Der Unterschied wird aber auch anhand des Ausschnittes (Regeln i bis iii) aus dem Lernergebnis der Wissensentdeckung deutlich, wenn wir drei Fälle betrachten, in denen die Datenbank unvollständig ist:

Fall 1: Es ist unbekannt, ob *jane* Kundin ist.

Fall 2: Das Einkommen von *jack* ist unbekannt.

Fall 3: Beides, das Einkommen von *jack* und ob *jane* Kundin ist, ist unbekannt.

Im ersten Fall können Regeln (i) und (ii) nicht von der Wissensentdeckung akzeptiert werden, weil *jane* nicht in jedem minimalen Modell Kundin ist. Regel (iii) kann gefunden werden. Wenn wir *kunde_ja* als zu lernenden Begriff auffassen, kann das Begriffslernen im ersten Fall die Regeln (i) und (ii) sehr wohl lernen. Es wird dann vorausgesagt, daß *jane* Kundin ist. Je nach Präferenzkriterium liefert das Begriffslernen die Regeln (i) und (iii) oder (ii) und (iii) als Ergebnis. Jede dieser beiden Hypothesen erfüllt das Vollständigkeitskriterium des Begriffslernens. Wenn allerdings negative Beispiele durch die *closed world assumption* gewonnen werden, schlägt das Begriffslernen fehl. Die Korrektheitsbedingung ist nicht gewährleistet bei Regeln (i) und (ii) und Regel (iii) kann allein nicht alle positiven Beispiele ableiten. Wegen seiner Vollständigkeitsbedingung kann das Begriffslernen dann gar nichts lernen.

Im zweiten Fall findet die Wissensentdeckung alle drei Regeln. Die minimalen Modelle, in denen das Einkommen von *jack* hoch ist, widersprechen ebensowenig den Regeln wie diejenigen Modelle, in denen das Einkommen nicht hoch ist. Das Begriffslernen kann die Regeln (i) und (iii) finden und leitet dann ab, daß *jane*, *ann*, *bob*, *joan* und *jack* Kunden sind. Dies gelingt auch bei Anwendung der *closed world assumption*.

Im dritten Fall kann das Begriffslernen keine der Regeln akzeptieren, weil $kunde\_ja(jack)$ nicht abgeleitet werden kann. Auch ohne *closed world assumption* mißlingt das Begriffslernen. Die Wissensentdeckung hingegen findet auch in diesem Falle noch die Regel (iii). Die Wissensentdeckung findet also in allen Fällen etwas, während das Begriffslernen nur in einigen Fällen ein Ergebnis findet. Es gibt also Lernaufgaben, in denen die Wissensentdeckung gelingt, aber nicht das Begriffslernen. Sie löst also ein Problem, das (manchmal) für das Begriffslernen zu schwierig ist. Man kann das Begriffslernen durch die *closed world assumption* dahin bringen, daß sein Ergebnis die erste Bedingung der Wissensentdeckung erfüllt. Man kann es nicht dazu bringen, daß es die Vollständigkeitsbedingung der Wissensentdeckung erfüllt.

5.5.1 Wissensentdeckung mit deklarativer Sprachbeschränkung – RDT

Die anspruchsvolle Lernaufgabe der Wissensentdeckung macht eine Beschränkung an anderer Stelle zwingend erforderlich. Sprachbeschränkungen sind eine Möglichkeit, Lernen relationaler Begriffe effizient zu machen. Die Sprachbeschränkung kann vom System implizit vorgegeben oder deklarativ und veränderbar dargestellt sein. Im letzteren Fall spricht man vom *declarative bias*. Ein Lernverfahren, das eine explizite, syntaktische Sprachbeschränkung durch den Benutzer zuläßt, ist RDT [Kietz und Wrobel, 1992]. Das Lernverfahren ist Teil des Systems MOBAL [Morik et al., 1993].

Mengen syntaktisch gleicher Regeln kann man durch Regelschemata ausdrücken.

Definition 78: Regelschema. *Ein Regelschema RS hat die Form einer Regel R, in der mindestens ein Prädikatensymbol durch eine Prädikatenvariable ersetzt wurde.*

Alle Instanzen eines Regelschemas sind von der gleichen Form und unterscheiden sich nur durch die eingesetzten Prädikatensymbole. Diese Einsetzung von Prädikatensymbolen nennen wir Instantiierung des Regelschemas.

Definition 79: Instantiierung eines Regelschemas. *Eine Substitution Σ substituiert Prädikatenvariablen, ohne sie zu unifizieren. Die Anwendung von Σ auf ein Regelschema ergibt die Instantiierung des Schemas.*

Gemäß der Faustregel für die θ-Subsumtion ist ein längeres Regelschema auch ein spezielleres Regelschema, solange man nicht in den Prämissen dasselbe Prädikatensymbol für verschiedene Prädikatsvariablen einsetzt. Das Einsetzen des gleichen Prädikatensymbols für verschiedene Prädikatsvariable ist aber in obiger Definition explizit ausgeschlossen.

Definition 80: Genereller-als Beziehung für Regelschemata. *Ein Regelschema RS ist genereller als ein anderes RS', wenn es Substitutionen Σ und σ gibt, so daß $RS' \supseteq RS\Sigma\sigma$.*

Man kann also die Suche nach der generellsten Regel, die alle positiven Beispiele abdeckt und kein negatives, dadurch strukturieren, daß man zunächst alle möglichen Instanzen für das kürzeste Regelschema überprüft und dann – solange noch negative Beispiele abgedeckt werden – jeweils die nächst längeren Regelschemata mit allen möglichen Instanzen testet. Man braucht für eine Instanz keine weitere Spezialisierung zu versuchen, wenn schon für diese Instanz zu wenig Beispiele in den Fakten zu finden sind. Dadurch, daß die Spezialisierung abgebrochen wird, sobald eine konsistente generellste Spezialisierung gefunden ist, werden keine Regeln mit redundanten Literalen gelernt.

Damit nur generative Klauseln gelernt werden, wird die Instanziierung zusätzlich beschränkt. Die Prämissen werden danach geordnet, wie direkt ihre Argumente mit dem (oder den) Argument(en) der Konklusion verbunden sind. Dabei ist das Argument der Konklusion mit sich selbst am direktesten verbunden. Dasselbe Argument in einer mehrstelligen Prämisse kann ein anderes Argument mit der Konklusion verbinden. Dieses andere Argument kann dann in einer mehrstelligen Prämisse ein weiteres Argument einbinden, und so fort. Prädikate können nur so in ein Regelschema eingesetzt werden, daß ihre Argumente mit der Konklusion verbunden sind.

Die Regelschemata werden MOBAL vom Benutzer eingegeben und dann vom System auf ihre Generalisierungsbeziehung hin geordnet. Ein Beispiel für eine solche Anordnung von Regelschemata zeigt das Bild 10.

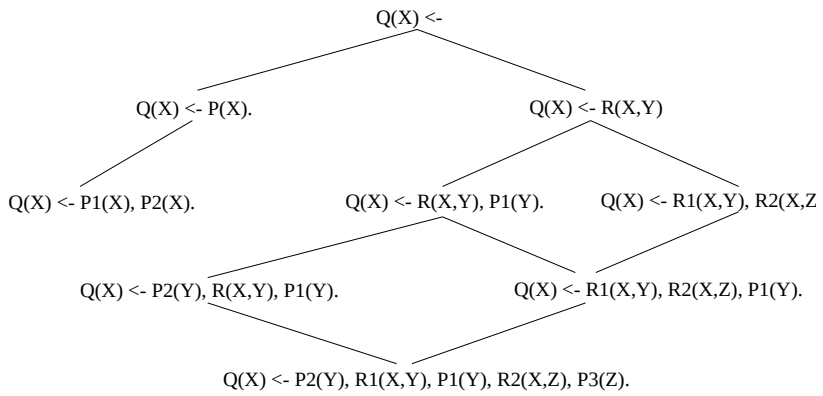


Abbildung10. Hierarchie von Regelschemata

Es kann verschiedene $\Sigma\sigma$ für eine Spezialisierung geben. Nehmen wir beispielsweise an, das Regelschema $Q(X) \leftarrow$ sei als $essbar(X)$ instanziiert. Dann seien als nächstes die beiden Spezialisierungen $essbar(X) \leftarrow pflanze(X)$ und $essbar(X) \leftarrow frisst(X, Y)$ getestet worden. Es gab aber nicht-eßbare Pflanzen und nicht-eßbare Objekte, die etwas anderes fressen. Es soll also weiter spezialisiert werden. Für die erste Hypothese $essbar(X) \leftarrow pflanze(X)$ kann das speziellere Regelschema

$$Q(X) \leftarrow P1(X), P2(X).$$

instanziiert werden. Das P des generelleren Schemas kann auf $P1$ oder $P2$ des ersten spezielleren Schemas abgebildet werden. In diesem Falle ändert das nicht viel. Es kann aber vorkommen, daß je nach Σ zusätzliche Prädikate über verschiedenen Objektbereichen gesucht werden. So zum Beispiel, wenn $essbar(X) \leftarrow frisst(X, Y)$ schon weiter spezialisiert wurde zu

$$essbar(X) \leftarrow frisst(X, Y), nachkomme(X, Z), pflanze(Y).$$

und nun weiter spezialisiert werden soll. Dann müssen für das nächst speziellere Regelschema den Prädikatsvariablen $P1, P2, P3, R1, R2$ die Prädikate $frisst$, $nachkomme$, $pflanze$, $essbar$ zugeordnet und die entsprechend noch nicht instanziierten Prädikatsvariablen mit zusätzlichen Prädikaten belegt werden. In das Regelschema

$$Q(X) \leftarrow P2(Y), R1(X, Y), P1(Y), R2(X, Z), P3(Z).$$

können die bereits ausgewählten Prädikate auf zwei verschiedene Weisen auf die Prädikatssymbole verteilt werden.

$$\Sigma 1 : \{R1/frisst, R2/nachkomme, P1/pflanze\}.$$

Dann müssen zwei neue Prädikate, eine über Pflanzen und eins über die Nachkommen gesucht werden, etwa $ungiftig$ und ei.

$$essbar(X) \leftarrow ungiftig(Y), frisst(X, Y), pflanze(Y), nachkomme(X, Z), ei(Z).$$

Eine andere Substitution ist

$$\Sigma 2 : \{R1/nachkomme, R2/frisst, P3/pflanze\}.$$

Dann müssen zwei neue Prädikate über die Nachkommen gesucht werden, etwa *gross* und *ei*.

$$essbar(X) \leftarrow ei(Y), nachkomme(X,Y), gross(Y), frisst(X,Z), pflanze(X).$$

Im ersten Fall wurde nach einer weiteren Eigenschaft für Pflanzen (*ungiftig*), im zweiten nach einer weiteren Eigenschaft für Nachkommen (*gross*) gesucht. Beide Hypothesen sind gleich speziell. Beide Hypothesen werden getestet, ob sie einem vom Benutzer gegebenen Bewertungskriterium genügen. Beim Testen werden die Argumentvariablen durch konstanten Terme substituiert, so daß sie positiven oder negierten Fakten entsprechen.

RDT durchsucht den durch die (Instanziierung von) Regelschemata gegebenen Suchraum vollständig. Die Suche wird nur dort beschnitten, wo mit Sicherheit keine Hypothesen mehr angenommen werden können. So braucht eine Hypothese nicht spezialisiert zu werden, die nicht genügend positive Beispiele abdeckt – sie wird durch Spezialisierung nicht mehr Beispiele abdecken. RDT lernt durch Spezialisierung die allgemeinste Generalisierung für eine Menge von Daten, die dem Bewertungskriterium genügt. Da der Benutzer auch ein Kriterium angeben kann, das sehr viel leichter zu erfüllen ist als die der Wissensentdeckungs-Lernaufgabe, ist RDT skalierbar für unterschiedliche Lernaufgaben.

RDT wurde so erweitert, daß es als Hypothesentest SQL-Anfragen an eine Datenbank stellt [Morik und Lindner, 1995]. Damit ist es als Werkzeug für die Wissensentdeckung in Datenbanken algemein einsetzbar. Bei einer Anwendung für die Daimler Benz AG konnte das Verfahren über einer Gigabyte großen Datenbank arbeiten. Allerdings lieferte erst das Zusammenspiel mit anderen Lernverfahren *interessante* Ergebnisse für die Anwender [Brockhausen und Morik, 1997].

5.6 Identifikation im Grenzwert – MIS

Das Model Inference System (MIS) von Ehud Shapiro war das erste Verfahren für Prädikatenlogik, das von einer allgemeinsten Hypothese ausgeht und diese Hypothese so lange bearbeitet, bis sie alle wahren und keine falschen Fakten abdeckt [Shapiro, 1981], [Shapiro, 1983]. Hier wird es angeführt, um neben dem Lernen als Funktionsapproximation und induktiver logischer Programmierung (Lernen als Datenmodellierung) auch das Paradigma der *Identifikation im Grenzwert* vorzustellen.

Die Lernaufgabe, die das MIS löst, besteht darin, aus variablenfreien Fakten, die in einem logischen Modell M gelten, als Hypothese ein Axiomensystem T in einer dazu passenden Signatur L zu inferieren. Dabei gibt ein Orakel (z.B. der Benutzer) an, ob ein Fakt wahr ist, also im Modell gilt, oder nicht.

Definition 81: Lernaufgabe MIS. *Sei das folgende gegeben:*

- *eine Beobachtungssprache LE und eine Hypothesensprache LH, die beide in der Signatur L sind*
- *ein Orakel, das zu jedem Fakt den Wahrheitswert liefert gemäß seiner Gültigkeit in M*
- *ein vollständiges deduktives Ableitungsverfahren;*

Das Ziel ist eine Theorie, dargestellt in LH, aus der alle wahren Fakten (also alle Fakten aus LE, die vom Orakel als wahr bewertet wurden) abgeleitet werden können, und keine falschen.

Das Ziel ist also eine bezüglich der Beobachtungssprache vollständige Axiomatisierung von M. Die Situation veranschaulicht Bild 11.

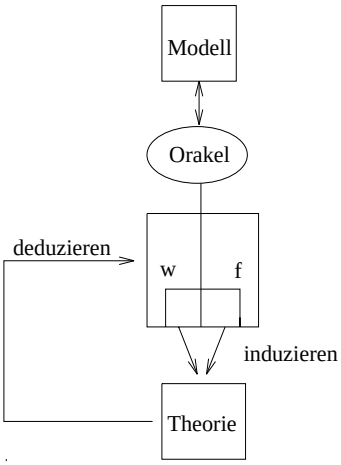


Abbildung11. Lernsituation von MIS

Dabei sollen LE und LH angemessen (*admissible*) sein, d.h. zusammen passen.

Definition 82: Angemessenheit von zwei Sprachen. *Angemessen sind zwei Sprachen einer Signatur, wenn für jedes Modell $\mathcal{M}$ der Signatur und jede Theorie, ausgedrückt in der Hypothesensprache LH, daraus, daß die wahren Fakten, ausgedrückt in LE, aus der Theorie ableitbar sind, folgt, daß die Theorie gültig ist in $\mathcal{M}$.*

Dies ist die Grundannahme, mit der eine solche Lernaufgabe überhaupt erst möglich wird. Wenn sie nämlich nicht gelten würde, dann hätten wir die paradoxe Situation, daß eine Theorie genau alle wahren Fakten ableitet und doch falsch ist. Diese Situation käme insbesondere dann vor, wenn wir einen für den Wahrheitswert wichtigen Unterschied in der Beobachtungssprache nicht ausdrücken können. Ehud Shapiro beruft sich auf die erkenntnistheoretische Forderung, daß Theorien durch Fakten falsifizierbar sein sollen. Zu falschen Theorien soll es auch Fakten geben, die vom Orakel als falsch bewertet werden, so daß die Theorie abgelehnt werden kann.

Eine weitere Beschränkung des Problems besteht in der Anzahl der Ableitungsschritte, die für die Ableitungen zugelassen wird. Dabei wird nicht einfach eine konstante Beschränkung gewählt, sondern eine Funktion h, die für das i-te Fakt die Zahl der Ableitungsschritte $h(i)$ festlegt. Die Funktion h sei vollständig berechenbar und rekursiv. Wir können aufgrund dieser Beschränkung nur solche Modelle axiomatisieren, bei denen die Theorie für jedes Fakt a_i nur $h(i)$ Ableitungsschritte benötigt, um es aus der Theorie abzuleiten. Wir nennen ein Modell, für das es so eine Theorie gibt, *h-leicht* (*h-easy*).

Definition 83: h-leichte Modelle. *Ein Modell ist h-leicht (h-easy), wenn es eine Theorie gibt, die für jedes Fakt a_i nur $h(i)$ Ableitungsschritte benötigt. Die Funktion h muß vollständig berechenbar und rekursiv sein.*

Der Algorithmus geht von der allgemeinsten Theorie als Hypothese aus. Zwei Mengen von Fakten, die wahren und die falschen, werden nach ihrer Eingabe gespeichert. Solange es einen falschen Fakt gibt, der aus der Theorie abgeleitet wird, oder einen wahren Fakt a_i

gibt, der nicht in $h(i)$ Schritten aus der Theorie abgeleitet werden kann, wird die nächste Theorie genommen und ausgegeben. Der Algorithmus hält nie an. Die Bedingungen für den Übergang zur nächsten Theorie sind dergestalt, daß der Algorithmus die Theorie im Grenzwert identifiziert: nur wenn Falsches abgeleitet wird oder Wahres nicht abgeleitet wird, geht der Algorithmus zur nächsten Theorie über, sonst nicht. Bei der Wahl der nächsten Hypothese (Theorie) werden die beiden Bedingungen für die Notwendigkeit, eine andere Theorie zu finden, benutzt. Eine Theorie ist zu stark, wenn sie (auch) Falsches ableitet. Dann muß sie abgeschwächt werden. Das heißt, die nächste Theorie sollte ähnlich wie die vorhergehende sein, nur schwächer. Eine Theorie ist zu schwach, wenn sie Wahres nicht ableiten kann. Dann sollte sie verstärkt werden. Man braucht also zwei Verfahren, eines zur Abschwächung und eines zur Verstärkung.

Shapiros Verfahren zur Abschwächung heißt **contradiction backtracing**. Es findet die falsche Hypothese aus der zu starken Theorie, die die Ableitung eines falschen Fakts ermöglichte. Dabei wird der Ableitungsbaum, der zum Widerspruch führt, ausgenutzt. Zum Beispiel könnte es den folgenden Ableitungsbaum geben, der den Widerspruch zwischen T und $\neg T$ darstellt (Abbildung 12).

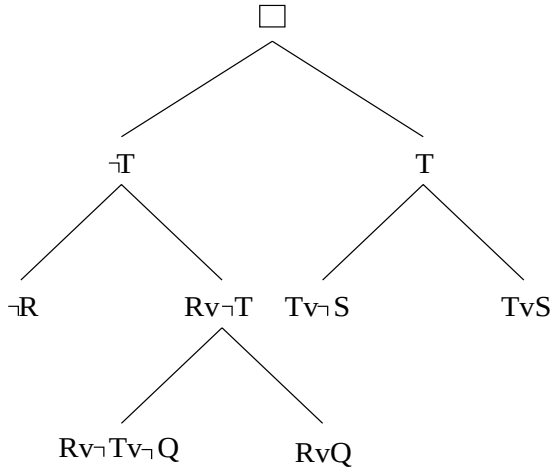


Abbildung12. Ableitungsbaum

Der Algorithmus beginnt an der Wurzel und befragt das Orakel, ob das jeweils resolvierte Atom im Modell M wahr ist oder nicht. Zuerst wird also gefragt, ob T gilt. Dabei sind die Fragen so formuliert, daß nach einem Grundatom (aus LE) gefragt wird. Das Orakel liefert nur Wahrheitswerte für Fakten, LE enthält ja nur Fakten. Substitutionen sind bereits beim Beweis mit dem Resolutionsprinzip entstanden. Falls aber ein resolviertes Atom nicht schon ein Grundbeispiel ist (was es ja nicht zu sein braucht), so wird die Substitution passend erweitert. Wenn das Orakel sagt, daß das Atom im Modell wahr ist, so kann der Fehler nicht in der rechten Seite sein, weil dort die nicht negierten Atome stehen. Also wird dann nach links der Kante gefolgt und beim nächsten resolvierten Atom gefragt, ob es wahr ist. Im Beispiel wird dann nach R gefragt. Wenn das Orakel sagt, daß das resolvierte Atom falsch ist, wird nach rechts weitergegangen. Es wird auf diese Weise gezielt gefragt, bis ein Blatt erreicht ist. Dieses Blatt ist die falsche Hypothese. Sie wird aus der Theorie gelöscht. Die neue Theorie ist aus der zu starken Theorie durch Löschen einer falschen Hypothese entstanden. Sie ist damit schwächer als die vorherige Theorie.

Das Verfahren läßt sich wie folgt zusammenfassen. Es wird mit dem Ableitungsbaumes, der den Widerspruch darstellt, aufgerufen.

Algorithmus 84: Contradiction Backtracing. Eingabe: Ableitungsbaum, bei dem links die negierten und rechts die nicht-negierten Fakten stehen.

- Wenn der Ableitungsbaum nur aus einem Blatt besteht, entferne dieses aus der Theorie und halte an. Sonst
- Frage Orakel nach Grundatom, das durch Resolution aus den beiden Teilbäumen herausgeschnitten wurde.
  - Wenn Orakel "wahr" antwortet, rufe Contradiction Backtracing rekursiv für den linken Teilbaum auf.
  - Wenn Orakel "falsch" antwortet, rufe Contradiction Backtracing rekursiv für den rechten Teilbaum auf.

Dieses Verfahren ist auch für praktische Anwendungen interessant. So wird es beispielsweise als Hilfe bei der Fehlersuche von Prolog-Systemen eingesetzt.

Die Verstärkung einer Theorie erreicht Shapiro mit einem Verfeinerungsoperator, der schrittweise Verfeinerungen (Spezialisierungen zurückgewiesener Hypothesen) in die zu schwache Theorie einfügt. Der Verfeinerungsoperator findet für jede Formel P eine umfangreichere Formel Q, die von P impliziert wird. Das ist so nicht berechenbar (alles, was von P impliziert wird). Ein konkreter Verfeinerungsoperator findet daher ein Q, das mit P in einer Verfeinerungsrelation sr steht.

Definition 85: Schrittweise Verfeinerungsrelation $sr(P,Q)$. *Für $P, Q \in LH$ gilt die Verfeinerungsrelation $sr(P,Q)$ gdw.*

1. *P ist ein Atom, das die Variablen U und V enthlt, dann ist Q dasselbe Atom mit angehängtem $U = V$; oder*
2. *P ist ein Atom, dann ist Q dasselbe Atom mit einer n-stelligen Funktion über Variablen X_i, die in p nicht vorkommen (Substitution von U durch $f(X_1, \ldots, X_n), f \in LH$); oder*
3. *Q ist $P \cup \{Li\}$, wobei Li ein generellstes Literal ist bezüglich P, so daß Q reduziert ist* [16].

Q soll stets umfangreicher sein als P, zum Beispiel dadurch, daß es mehr Prämissen enthält als P.

Über der Verfeinerungsrelation ist eine partielle Ordnung gegeben. Ein Verfeinerungsoperator ist vollständig, wenn aus der leeren Aussage über schrittweises Verfeinern alle in LH bildbaren Sätze erzeugt werden können.[17] Auf die Unvollständigkeit des Verfeinerungsoperators sr wird in Abschnitt 5.7.1 eingegangen. Der Verfeinerungsoperator ist ein

[16] Zur Reduktion von Formeln siehe Abschnitt 5.1.2. Definition und Diskussion des generellsten Literals folgen im Abschnitt 5.7.1.

[17] Die Verfeinerung bei RDT (siehe 5.5.1) ist bezüglich der Hornlogik mit expliziter Negation nicht vollständig, sondern auf die vorhandenen Regelschemata eingeschränkt. Wenn man die Sprache ebenfalls auf die Schemata einschränkt, ist die Verfeinerung bei RDT vollständig.

konstruktives Verfahren. Der Verfeinerungsoperator geht entlang der Verfeinerungsrelation von einer Ebene von Sätzen zur nächsten Ebene von verfeinerten Sätzen. Man kann sich die Sätze also in einem gerichteten, azyklischen Graphen angeordnet vorstellen, bei dem die Kanten die Verfeinerungsrelation darstellen und die Knoten Hypothesen sind. Dabei ist der Wurzelknoten die leere Aussage. Obendrein gilt für einen Knoten dieses Graphen, daß, wenn die entsprechende Hypothese im Modell M gültig ist, so auch die aller Nachfolgerknoten und keine der Vorgängerknoten. Die richtige Theorie für das Modell kann man sich als Linie durch den Verfeinerungsgraphen vorstellen wie in Abbildung 13 schematisch dargestellt.

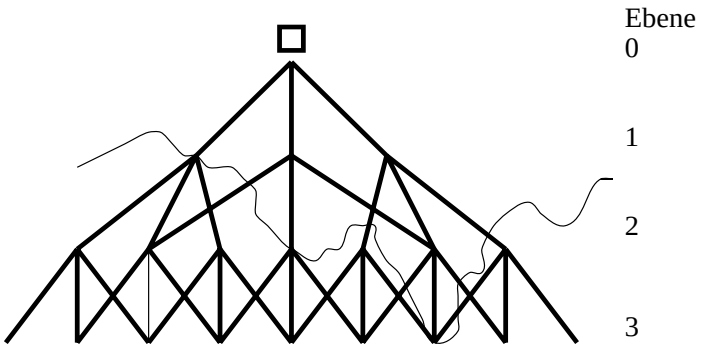


Abbildung13. Verfeinerungen

MIS besteht aus dem *contradiction backtracing* und dem Verfeinerungsoperator und arbeitet wie folgt:

Algorithmus 86: MIS. Wähle irgendeine Theorie als Ausgangspunkt

- repeat
  - lies den nächsten Fakt ein
  - repeat
    - wenn T zu stark ist, wende *contradiction backtracing* an;
    - wenn T zu schwach ist, wende den Verfeinerungsoperator an;
  - until T ist weder zu schwach noch zu stark bezüglich aller bisher gesehener Fakten;
  - drucke T aus.
- forever

Ein Beispiel für MIS aus dem Bereich natürlicher Zahlen ist:

LE:	0	interpretiert als 0
	X'	interpretiert als Nachfolger von X
	$X \leq Y$	interpretiert als X ist kleiner oder gleich Y
	$plus(X, Y, Z)$	interpretiert als $X + Y = Z$
LH:	Hornformeln	
Fakten:	$0 \leq 0$ ist wahr	
	$plus(0, 0', 0)$ ist falsch	

$plus(0', 0'', 0''')$ ist wahr
...
Theorie: $X \leq X$
$X \leq Y' \leftarrow X \leq Y$
$plus(X, 0, X)$
$plus(X, Y', Z') \leftarrow plus(X, Y, Z)$

Wichtig ist bei diesem Verfahren, daß es ganz offensichtlich – gerade so sind die Bedingungen im Algorithmus formuliert – eine falsche (zu starke oder zu schwache) Theorie verändert und eine einmal gefundene richtige Theorie nicht mehr verändert. Daß es eine falsche Theorie immer weiter verändert, liegt an der Angemessenheit der Beobachtungssprache für die Hypothesensprache. Daß es einen Grenzwert gibt, zu dem tatsächlich eine richtige Theorie gefunden wird, liegt an der Angemessenheit der Sprachen, der Monotonieeigenschaft der Logik und an der Vollständigkeit des Verfeinerungsoperators.

5.7 Spezialisierungsoperatoren

Die Generalisierungsbeziehung zwischen Klauseln strukturiert den Hypothesenraum. Wir haben nun Generalisierungsoperatoren kennengelernt, die von zwei Beispielen oder zwei spezielleren Hypothesen eine minimal generellere Hypothese erzeugen. Diese Generalisierungsoperatoren durchsuchen den Hypothesenraum *bottom-up*. Die Verfahren FOIL und RDT durchsuchen den Hypothesenraum *top-down*. Beide verwenden also Spezialisierungsoperationen. Das MIS bietet zwei Spezialisierungsoperationen an: eine bedarfsgestützte, die bei Widersprüchen eine falsche Aussage aus der Theorie löscht (*contradiction backtracing*), und eine Verfeinerungsoperation, die eine falsche und deshalb verworfene Aussage spezialisiert, bis die Spezialisierung korrekt ist. In diesem Abschnitt sollen Spezialisierungsoperatoren ausführlich behandelt werden. In Abschnitt 5.7.1 wird das Problem der Vollständigkeit eines Verfeinerungsoperators behandelt. Kann man wirklich von der leeren Klausel ausgehend alle Klauseln durch eine Folge von Operator-Anwendungen erreichen? Eine bedarfsgestützte minimale Spezialisierung einer Theorie wird in Abschnitt 5.7.2 beschrieben. Interessant ist dabei, daß hier die Spezialisierungsoperation mit dem Bereich des nicht-monotonen Schließens in Beziehung gesetzt wird.

5.7.1 Vollständige Spezialisierung unter θ-Subsumtion

Um effizienter zu verfeinern, hat Ehud Shapiro den Verfeinerungsoperator so definiert, daß er mit reduzierten Formeln arbeitet [Shapiro, 1981]. Unter θ-Subsumtion sind Formeln reduziert, wenn sie keine redundanten Literale enthalten (siehe Abschnitt 5.1). Als weitere Einschränkung war die Verfeinerung *sr* in MIS stets *umfangreicher* als das Verfeinerte. Insbesondere durfte die Anzahl der Literale bei der Verfeinerung nicht sinken. Was zunächst einleuchtend aussieht, führt leider zu einem Problem, das Patrick van der Laag und Shan-Hwei Nienhuys-Cheng entdeckten [van der Laag, 1995][Nienhuys-Cheng et al., 1993]. Wenn wir bei der Verfeinerung nur reduzierte Formeln zulassen und keine kleiner wird als ihr Vorgänger, gibt es zwischen einigen reduzierten Formeln keine Verfeinerungsbeziehung, obwohl die eine Formel die andere subsumiert. Um wirklich alle Formeln erreichen zu können, müssen entweder auch nicht-reduzierte Formeln in dem Verfeinerungsgraphen als Knoten vorkommen dürfen. Oder der Verfeinerungsoperator darf eine Formel verkürzen. Zwei Beispiele zeigen das Problem.

Seien P und Q zwei reduzierte Formeln, wobei P allgemeiner ist als Q.
$P : \{p(a, W), p(X, b), p(c, Y), p(Z, d)\}$
$Q : \{p(a, b), p(c, b), p(c, d), p(a, d)\}$

Es gibt einen Pfad zwischen P und Q, aber er enthält nicht reduzierte Formeln.
$P_1 : \{p(a, b), p(X, b), p(c, Y), p(Z, d)\}$
$P_2 : \{p(a, b), p(c, b), p(c, Y), p(Z, d)\}$
$P_3 : \{p(a, b), p(c, b), p(c, d), p(Z, d)\}$
$Q : \{p(a, b), p(c, b), p(c, d), p(a, d)\}$

Alle Substitutionen, die notwendig sind, um mit sr schrittweise von P nach Q zu kommen, führen über nicht-reduzierte Klauseln. Eine nicht-reduzierte Klausel θ-subsumiert einen Teil von sich selbst. So ist z.B. die reduzierte Fassung von P_1

$P_1' : \{p(a, b), p(c, Y), p(Z, d)\}$

Das Literal $p(X, b)$ ist redundant unter der θ-Subsumtion (siehe Abschnitt 5.1), weil $P_1\theta \subseteq P_1 \setminus \{p(X, b)\}$ mit $\theta = \{X/a\}$. Diese kürzere Fassung darf laut Shapiro nicht eingeführt werden. Tatsächlich ist am Ende Q nicht kürzer als P. Ohnehin ist eine Spezialisierung manchmal kürzer:

$\{p(X, Y), p(Y, X)\}$ subsumiert $\{p(X, X)\}$

In diesem speziellen Falle hatte Shapiro die Gleichheit als zusätzliches Literal angehängt:

$\{p(X, Y), p(Y, X), Y = X\}$

Das Problem hängt nicht nur mit der Länge von Klauseln zusammen, wie das nächste Beispiel zeigt, bei dem auf dem Weg zwischen einer kurzen, allgemeineren Klausel und ihrer längeren Spezialisierung eine nicht-reduzierte Klausel liegt.

$P : \{p(X), \neg q(X, a)\}$
$Q : \{p(X), \neg q(X, a), \neg q(Y, Z), \neg q(Z, Y)\}$

Nun ist einer der Zwischenschritte bei der Verfeinerung mit sr eine nicht-reduzierte Klausel, nämlich R.
$R : \{p(X), \neg q(X, a), \neg q(Y, Z)\}$

Mit $\theta = \{Y/X, Z/a\}$ fallen die beiden letzten Literale zusammen und $R\theta \subseteq R \setminus q(Y, Z)$, d.h.: R ist nicht-reduziert.

Das Beispiel ist gerade so konstruiert, daß die *beiden* letzten Literale in Q gemeinsam unmöglich machen, eines davon zu streichen.

Die Beispiele zeigen, daß der Verfeinerungsoperator von Ehud Shapiro unvollständig ist, weil er den Klauselgraph auf reduzierte und immer umfangreicher werdende Klauseln einschränkt. Andererseits sind seine Motive für die Einschränkung ja überaus einleuchtend – irgendwie muß die Verfeinerung eingeschränkt werden, um handhabbar zu sein. Wir müssen also aus den Gegenbeispielen zu sr lernen, wie die Verfeinerung zu verbessern ist. Dazu verwenden van der Laag und Nienhuys-Cheng zwei Hilfsmittel: $eq(P)$ und die Idee des *generellsten Literals*.

Definition 87: eq(P). *Die Funktion $eq(P)$ liefere alle zu P äquivalenten und höchstens um k Literale erweiterten Klauseln.*

Das *generellste Literal* hilft bei der Auswahl, welches Literal zu der Klausel bei der Spezialisierung hinzugefügt werden soll.

Definition 88: Generellstes Literal bezgl. einer reduzierten Klausel. *Für eine reduzierte Klausel C heißt ein Literal L das generellste Literal bezüglich C, wenn $C \cup \{L\}$ reduziert ist und für jedes andere Literal M, das echt genereller ist als L bezüglich C, ist $C \cup \{M\}$ nicht reduziert.*

Es geht bei dieser Definition um die sinnvolle Erweiterung einer Klausel bei der Spezialisierung. Die erweiterte Klausel soll im Raum der reduzierten Klauseln gerade "einen Schritt" unter der Ausgangsklausel liegen.

Ein Beispiel:

> Sei $C : q(X) \leftarrow p(X, Y)$. Wenn wir jetzt erweitern zu $C' : q(X) \leftarrow p(X, Y), p(U, X)$, so gibt es kein θ das die beiden Literale im Klauselkörper zusammenfallen läßt. Allerdings ist die Generalisierung von $p(U, X)$ bezüglich C zu $p(U, V)$ nicht mehr durch eine gemeinsame Variable mit dem Rest der Klausel beschränkt und $C'' : q(X) \leftarrow p(X, Y), p(U, V)$ kann mit $\theta = \{U/X, V/Y\}$ reduziert werden. Das Literal $L : p(U, X)$ ist also ein generellstes Literal bezüglich C, während das Literal $M : p(U, V)$ zu allgemein ist.

Nun haben wir oben gesehen, daß die Beschränkung auf den Raum der reduzierten Klauseln die Spezialisierung unvollständig macht. Deshalb beziehen van der Laag und Nienhuys-Cheng die äquivalenten, nicht-reduzierten Klauseln der Länge m mit ein. Diese äquivalenten Klauseln werden durch die *inverse Reduktion* gefunden. Es ist einigermaßen einfach, eine Reduktion rückgängig zu machen: an die reduzierte Klausel C muß mindestens ein Literal L angehängt werden mit einem Prädikatensymbol und Vorzeichen, das in C vorkommt. Die Argumente in L müssen teilweise disjunkt von den in C vorkommenden Variablen sein, damit sie bei der Reduktion via θ mit einer in C vorkommenden Variable ersetzt werden können. Dann können wir Varianten außer acht lassen und kommen zu allen möglichen θ-Substitutionen. Das einzige Problem bei der inversen Reduktion ist, daß wir beliebig viele Literale für jedes Literal in C anhängen könnten. Deshalb wird für die äquivalenten nicht-reduzierten Klauseln eine Längenbeschränkung eingeführt. Es dürfen nur k zusätzliche Literale angehängt werden.

Ein Beispiel:

> $C : \{p(X, X)\}$, $k = 1$
> $eq(C) : \{\{p(X, X), p(X, Y)\}, \{p(X, X), p(Y, X)\}, \{p(X, X), p(Y, Z)\}, \{p(X, X), p(Y, Y)\}\}$
>
> Hier wurden Y und Z als die nicht in C vorkommenden Variablen gewählt – jeder andere Bezeichner stellt nur eine Variante dar. Mit $\theta = \{Y/X\}$ bzw. $\theta = \{Y/X, Z/X\}$ wird jede Klausel in $eq(C)$ wieder reduziert.

Der neue Verfeinerungsoperator sr_τ unterscheidet sich jetzt von dem in Abschnitt 5.6 vorgestellten prinzipiell nur darin, daß auch auf die Klauseln in $eq(C)$ die Substitutionen des Verfeinerungsoperators angewandt werden. Der besseren Lesbarkeit halber ist die Substitution der Variablen in einem Atom aufgeteilt in die Substitution durch andere Variablen und durch Funktionen (einschließlich Konstanten). Die Bedingung, daß P die Spezialisierung Q $\theta\Leftrightarrow$subsumiert war in sr implizit.

Definition 89: Schrittweise Verfeinerungsrelation $sr_\tau(P, Q)$. *Für zwei Klauseln in der Sprache L gilt die Verfeinerungsrelation* $sr_\tau(P, Q)$ *gdw.*

1. P θ⇔*subsumiert echt* Q *und es gibt Klauseln* $P' \in eq(P)$ *und* $Q' \in eq(Q)$, *so daß* $Q' = P'\theta$, *wobei* $\theta = \{X/Y\}$, *und sowohl* X *als auch* Y *kommen in* P' *vor; oder*

2. P θ⇔*subsumiert echt* Q *und es gibt Klauseln* $P' \in eq(P)$ *und* $Q' \in eq(Q)$, *so daß* $Q' = P'\theta$, *wobei* $\theta = \{X/f(Y_1, \ldots, Y_n\}$ *und* X *kommt in* P' *vor und die* Y_i *sind verschiedene, nicht in* P' *vorkommende Variablen; oder*

3. $Q = P \cup \{M\}$, *wobei* M *das generellste Literal bezüglich* P *ist, das bezüglich Prädikatensymbol oder Vorzeichen von jedem Literal in* P *verschieden ist.*

Ein Beweis für die Vollständigkeit von sr_τ findet sich in [van der Laag, 1995]. Der dritte Fall erlaubt nur endlich viele generellste Literale, die angehängt werden können. Der erste und zweite Fall wendet Substitutionen auch auf die nicht-reduzierten Klauseln an, um die Verfeinerungsrelation zwischen ihren reduzierten Äquivalenten herzustellen. Wir können uns nun an den obigen Gegenbeispielen gegen sr klar machen, daß sr_τ nicht an ihnen scheitert.

$P : \{p(a, W), p(X, b), p(c, Y), p(Z, d)\}$
$Q : \{p(a, b), p(c, b), p(c, d), p(a, d)\}$

Im ersten Schritt wenden wir den zweiten Fall an, wobei
$P'_1 : \{p(a, b), p(X, b), p(c, Y), p(Z, d)\}, \theta = \{W/b\}$ und
$P_1 : \{p(a, b), p(c, Y), p(Z, d)\}$

Tatsächlich θ-subsumiert P echt P_1, d.h. mit $\theta = \{W/b, X/a\}$ gilt $P\theta \subseteq P_1$, wobei P und P_1 nicht äquivalent sind. Setzen wir $P' = P$ und nehmen $P'_1 \in eq(P_1)$, dann sehen wir, daß $P'_1 = P'\theta$ mit $\theta = \{W/b\}$, wobei W in P' vorkommt und die nullstellige Funktion b natürlich keine Variable aus P' enthält (siehe Fall 2). Wir stellen außerdem fest, daß P_1 auf dem richtigen Pfad liegt, denn es θ-subsumiert Q, d.h. z.B. mit $\theta = \{Y/b, Z/a\}$ gilt $P_1\theta \subseteq Q$.

Im nächsten Schritt wenden wir den zweiten Fall auf P'_1 an und erhalten mit $\theta = \{X/c\}$:
$P'_2 : \{p(a, b), p(c, b), p(c, Y), p(Z, d)\}$, wobei $P'_2 \in eq(P_2)$.
$P_2 : \{p(a, b), p(c, Y), p(Z, d)\}$

P_1 subsumiert P_2 und $P'_2 = P'_1\theta$, wobei X in P'_1 vorkommt und c nicht.

Es ergibt sich dieselbe Kette wie oben und alle reduzierten Klauseln liegen auf dem Pfad von P nach Q. Sie werden aber gefunden über ihre nicht-reduzierten Äquivalente.

Auch das zweite Gegenbeispiel gegen sr wird von sr_τ korrekt in eine Verfeinerungsrelation gestellt.

$P : \{p(X), \neg q(X, a)\}$
$Q : \{p(X), \neg q(X, a), \neg q(Y, Z), \neg q(Z, Y)\}$

Wir beginnen mit $P' \in eq(P)$:
$P' : \{p(X), \neg q(X, a), \neg q(Y, Z), \neg q(U, V)\}$

P' ist θ-äquivalent zu P. Nun wenden wir auf P' den ersten Fall von sr_τ an und erhalten mit der Substitution von U durch Z:
$P_1 : \{p(X), \neg q(X,a), \neg q(Y,Z), \neg q(Z,V)$

Hierauf wenden wir noch einmal den ersten Fall an mit $\theta = \{V/Y\}$ und sind dann richtig bei Q angekommen.

Obwohl die Pfade im Verfeinerungsgraphen stets endlich sind, ist der Graph *breiter* geworden: es müssen viel mehr Alternativen (nämlich alle äquivalenten Klauseln) untersucht werden. Setzen wir k zu klein, so ist die Vollständigkeit nicht gewährleistet (z.B. bei $k = 1$ im vorangegangenen Beispiel). Insofern ist sr_τ weniger von praktischem Wert als vielmehr erhellend für die Schwierigkeiten der Spezialisierung in Prädikatenlogik.

5.7.2 Minimale Spezialisierung

Wir haben eben die Spezialisierung *einer* Klausel betrachtet. Wie aber sieht es aus, wenn wir eine Menge von Klauseln spezialisieren wollen? Wie wir schon bei MIS in Abschnitt 5.6 gesehen haben, muß man dazu erst einmal diejenigen Klauseln in der Theorie finden, die Anlaß zu einer Spezialisierung geben. Es gilt also die Klauseln zu finden, die daran schuld sind, daß ein negatives Beispiel abgedeckt wird. Anders ausgedrückt: es gilt, die Klauseln zu finden, die für einen Widerspruch verantwortlich sind. Diese Umformulierung zeigt, warum der Spezialisierungsschritt, der im maschinellen Lernen vorgenommen wird, um negative Beispiele auszuschließen, so bedeutsam für viele Anwendungen ist. Es ist derselbe Schritt, der vorgenommen werden muß, wenn eine Wissensbasis inkonsistent geworden ist. Und das werden Wissensbasen im Laufe ihrer Verwendung immer: sei es, daß sich die Welt verändert, sei es, daß sich das Wissen über die Welt verändert! Es ist also gar nicht verwunderlich, daß es eine Fülle von Untersuchungen zum Thema der Wissensveränderung (Stichwort: *theory of change*) gibt – *knowledge in flux*, wie Peter Gärdenfors sein wichtiges Buch nannte. Bevor hier der Beitrag von Stefan Wrobel zur bedarfsgestützten minimalen Spezialisierung vorgestellt wird [Wrobel, 1993], sollen erst einmal die Schwierigkeiten der Wissensrevision klar werden.

Definition 90: Basis und Hülle einer Theorie. *Eine Theorie, in Prädikatenlogik dargestellt, besteht aus dem, was tatsächlich aufgeschrieben ist, der* Basis B*, und dem, was aus der Basis folgt, der inferentiellen Hülle* $Cn(B)$*.*

Definition 91: Abgeschlossenheit einer Theorie. *Eine Theorie ist abgeschlossen, wenn* $Cn(B) = B$*.*

Ein Beispiel:

$B : \{p(a),$
$\quad q(X) \leftarrow p(X)\}$

Wenn die Signatur nur a, b, p, q bereithält, dann ist
$Cn(B) : \{p(a), q(X) \leftarrow p(X), p(a) \vee p(b), p(a) \vee \neg p(b), \ldots\}$.

Das Problem der **Revision** besteht darin, unerwünschte Fakten (die hier ruhig allquantifizierte Variablen enthalten dürfen) auszuschließen:

Definition 92: Revision. *Sei das folgende gegeben:*

- *eine Theorie* B
- *eine Menge von Fakten* $F = \{f_1, \ldots f_n\}$

Ziel ist es, eine revidierte Theorie $B \frown F$ *zu finden, so daß* $Cn(B \frown F) \cap F = \emptyset$.

Mit der einleuchtenden Annahme, daß eine gute Revision so viel unangetastet läßt, wie es beim Ausschluß der unerwünschten Fakten möglich ist, kommt man zu der *maximal generellen und korrekten Spezialisierung* als Ziel der Revision [Muggleton und Bain, 1992].

Definition 93: Maximal generelle korrekte Spezialisierung. *Eine Theorie* B' *ist eine maximal generelle korrekte Spezialisierung von* B *bezüglich* F*, wenn*

- $Cn(B) \supseteq Cn(B')$,
- $F \notin Cn(B')$,
- *für alle anderen* B'' *gilt: wenn* $Cn(B) \supseteq Cn(B'') \supseteq Cn(B')$*, dann* $F \in Cn(B'')$.

Wenn eine Theorie also genereller ist als die maximal generelle, dann leitet sie F ab. Nun gibt es im Bereich des nicht-monotonen Schließens eine Reihe von Erkenntnissen über das Verhalten von abgeschlossenen Theorien. Die Postulate von Peter Gärdenfors [Gärdenfors, 1988] beschreiben eine Klasse von Revisionsoperatoren, über die man einiges weiß.

Gärdenfors Postulate: Die Spezialisierung bei abgeschlossenen Theorien soll den folgenden Postulaten gehorchen.

1. $B \frown f$ sei eine abgeschlossene Theorie.
2. $B \frown f \subseteq B$ (Inklusion)
3. Wenn $f \notin B$, dann $B \frown f = B$
4. Wenn $f \notin Cn(\emptyset)$, dann $f \notin B \frown f$ (Erfolg)
5. Wenn $Cn(f) = Cn(g)$, dann $B \frown f = B \frown g$ (Erhalt)
6. $B \subseteq Cn(B \frown f \cup \{f\})$ (Wiederherstellung)

Nun hat Bernhard Nebel festgestellt, daß alle Revisionsoperatoren, die den Gärdenfors-Postulaten gehorchen, mithilfe der maximalen korrekten Teilmengen einer abgeschlossenen Theorie definiert werden können [Nebel, 1989]. Diese Teilmengen sind die möglichen Spezialisierungen, wenn f ausgeschlossen werden soll. Es ist so eine Art Versionenraum für die Revision. Man spricht von diesen alternativen Möglichkeiten als B *ohne* f, geschrieben $B \downarrow f$.

Definition 94: Mögliche Spezialisierungen $B \downarrow f$, um f auszuschließen. *Sei* B *eine Theorie,* f *ein auszuschließendes Fakt, dann ist*

$B \downarrow f := \{B' \subseteq B \mid f \notin Cn(B')$ *und für alle* B'' *gilt:wenn* $B' \subset B'' \subseteq B$*, dann* $f \in Cn(B'')\}$

Für das Beispiel oben ist, wenn wir $f = q(a)$ entfernen wollen, $B \downarrow f$:

$$B \downarrow q(a) = \left\{ \begin{array}{l} \{p(a), p(a) \vee p(b), p(a) \vee \neg p(b), \ldots\} \\ \{q(X) \leftarrow p(X), p(a) \vee p(b), \ldots \quad\} \\ \{q(X) \leftarrow p(X), p(a) \vee \neg p(b), \ldots \quad\} \\ \ldots \end{array} \right\}$$

Nun können wir alle den Gärdenfors-Postulaten gehorchenden Revisionsoperatoren einheitlich darstellen, wobei sie jeweils unterschiedliche Auswahlen aus den maximalen Teilmengen treffen mögen.

Theorem 95: Revisionsoperation [Alchourron et al., 1985]. $\frown$ *ist eine Revisionsoperation, die die Gärdenfors-Postulate befolgt, gdw. es eine Selektionsfunktion* γ *gibt, die über den maximalen Teilmengen* $B \downarrow f$ *arbeitet, d.h.* $\gamma(B \downarrow f) \subseteq B \downarrow f$*, so daß*

$$B \frown f = \left\{ \begin{array}{ll} \bigcap \gamma(B \downarrow f) & \textit{wenn } f \notin Cn(\emptyset) \\ B & \textit{ansonsten.} \end{array} \right\}$$

Das Ergebnis der Revision ist also die Schnittmenge der von γ ausgewählten maximalen Teilmengen.

Definition 96: Maxi-choice contraction. *Wenn* γ *nur eine Menge auswählt, so handelt es sich um eine maxi-choice contraction.*

Diese Revisionsoperation entspricht ganz genau der Spezialisierungsoperation von Stephen Muggleton,[18] die eine maximal generelle korrekte Spezialisierung liefert. Das ist deshalb schade, weil gerade über die *maxi-choice contraction* Unangenehmes bekannt ist:

- Sie liefert eine abgeschlossene Theorie zurück. Damit ist sie für praktische Anwendungen nicht mehr relevant, denn nur bei Spielbeispielen im Skript kann man für eine abgeschlossene Theorie eine geeignete Basis finden (d.h. eine Axiomatisierung).
- Wenn wir mit abgeschlossenen Theorien arbeiten, wissen wir nicht mehr, was mal als Basis explizit angegeben war und was abgeleitet wurde. In unserem Beispiel könnten wir also, wenn wir $p(a)$ ungültig machen sollen, das abgeleitete $q(a)$ behalten. Wollen wir das?
- Als *Besserwisser-Effekt* wird bezeichnet, daß bei einer abgeschlossenen Theorie für jedes Fakt entweder das Fakt oder seine Negation gilt: für jede Aussage g gilt entweder $g \in Cn(B \frown f \cup \neg f)$ oder $\neg g \in Cn(B \frown f \cup \neg f)$. Wir wollten eigentlich nur f negieren und haben wundersamerweise neues Wissen über g in unserem Revisionsergebnis.

Wenn wir die Spezialisierung in einem Lernverfahren anwenden wollen, das negative Beispiele ausschließt, oder wenn wir sie einsetzen wollen, um eine Wissensbasis auf dem aktuellen Stand zu halten, dann müssen wir die *Basis* revidieren und die Spezialisierung so gestalten, daß sie die Basis möglichst wenig verändert. Natürlich müssen wir auch die Implikationen bedenken, aber nur in Bezug auf das vom Benutzer oder den bisherigen Lernschritten des Systems gegebene Wissen. Stefan Wrobel (a.a.O.) ersetzt deshalb das erste Gärdenfors-Postulat:

1. $B \frown f \subseteq B \cup \{g' \mid \exists g \in B$ so daß $g \geq g'\}$ (minimaler syntaktischer Abstand)

und setzt für B in den Gärdenfors-Postulaten, wo es noch nicht angegeben ist, $Cn(B)$.

Es geht nun darum, einen Revisionsoperator zu konstruieren, der die minimale Basisrevision vornimmt. Für diesen Operator brauchen wir die Buchführung, welche Fakten aus

[18] Der kurze Beweis in [Wrobel, 1993] zeigt, daß für eine nicht unbedingt abgeschlossene Theorie die maximal generelle Spezialisierung genau dem Ergebnis aller möglichen *maxi-choice contractions* von $Cn(B)$ entspricht.

welchen Regeln oder Fakten der Basis abgeleitet werden. Dann kann bestimmt werden, welche Regeln und Fakten der Basis an den auszuschließenden Fakten "beteiligt" waren (Anwendungsmenge). Um die Wiederherstellbarkeit zu gewährleisten, muß außerdem noch etwas hinzugefügt werden, das uns wieder f ableitet, wenn es denn doch in Ordnung war (Additionsmenge). Das Verfahren von Stefan Wrobel schränkt die Anwendbarkeit einer an der Ableitung von f beteiligten Regel so ein, daß f eben nicht mehr abgeleitet wird. Es fügt eine Regel ein, die mit f als Prämisse das wiederherstellt, was für den Ausschluß von f gelöscht wurde. Da der gesamte Beweisbaum für f berücksichtigt wird und nicht nur der letzte Resolutionsschritt, kann tatsächlich eine minimale Spezialisierung erreicht werden. Das Verfahren soll nun formal beschrieben werden. Für die Beweise, daß das Verfahren seinen Postulaten zur minimalen Basisrevision genügt und minimal spezialisiert, verweise ich auf [Wrobel, 1993, Wrobel, 1994].

Zuerst müssen die Schuldigen in einer (nicht abgeschlossenen) Theorie für die Ableitung eines unerwünschten Fakts f gefunden werden. Dazu werden alle Ableitungen von f hergenommen.

Definition 97: Unterstützungsmenge. *Die Unterstützungsmenge S von f ist:*

$$S := \{(C, \sigma, A) \mid \exists C \in B \wedge \exists g_1, \ldots, g_n \in Cn(B) \textit{so daß } C \textit{ mit } \{g_1, \ldots, g_n\} \textit{ unter Verwendung der Substitution } \sigma \textit{ resolviert und } f \textit{ ableitet.}\}$$

Die Ableitung der Prämissen A wird wiederum festgehalten.

Nehmen wir wieder $B : \{p(a), q(X) \leftarrow p(X)\}$ und $f : q(a)$. Es gibt in diesem Falle nur eine Ableitung von f, die in zwei Schritten erfolgt. Im ersten Schritt ist C die Regel $q(X) \leftarrow p(X)$ und $\sigma = \{X/a\}$. Die Prämisse A ist nun $p(a)$ und ist direkt in der Theorie zu finden, d.h. $\sigma = \{\emptyset\}$ und es gibt keine Prämissen. Wir erhalten als geschachtelte Tripel C, σ, A die Unterstützungsmenge für die Ableitung von f:

$$\{(q(X) \leftarrow p(X), \{X/a\}, (p(a), \emptyset, \emptyset))\}$$

Definition 98: Anwendungsmenge. *Die Anwendungsmenge $\Pi(f, B)$ ist die Menge, die alle Tupel (C, σ) aus der Unterstützungsmenge holt.*

In unserem Beispiel hat sie zwei Elemente:

$$\Pi(q(a), B) = \{P_1, P_2\}$$
$$P_1 = (p(a), \emptyset)$$
$$P_2 = (q(X) \leftarrow p(X), \{X/a\})$$

Die Klauseln in einem solchen P_i notieren wir $C(P_i)$

Die Anwendungsmenge enthält also die Schuldigen an der Ableitung von f. Es kann nun $B \downarrow_\Pi f$ für Anwendungsmengen statt für Theorien definiert werden.

Definition 99: Mögliche Spezialisierungen $B \downarrow_\Pi f$ für Anwendungsmengen. *Sei B eine Theorie, f ein auszuschließendes Fakt, dann ist*

$$B \downarrow_\Pi f := \{P \subseteq \Pi(f, B) \mid f \notin Cn(B_\Pi(P)) \textit{ und für alle } P' \textit{ gilt:}$$

$$P \subset P' \subseteq \Pi(f, B) \rightarrow f \in Cn(B_\Pi(P'))\}$$

Jedes P ist eine maximale Anwendungsmenge, d.h. es ist eine größte Teilmenge von $\Pi(f, B)$, die man beibehalten kann, ohne f einzuschließen.

Die Theorie B', die wir erhalten, wenn wir die maximale Anwendungsmenge P behalten und alle "unschuldigen" Aussagen natürlich auch, schreiben wir $B_{\Pi}(P)$.

Das Komplement der maximalen Anwendungsmenge ist das, was wir für die Praxis brauchen, nämlich die Aussagen, die wir löschen sollen, wenn wir f nicht mehr ableiten wollen. Das Komplement der maximalen Anwendungsmenge ist die **minimale Entfernungsmenge**:

Definition 100: Minimale Entfernungsmenge. *Die minimale Entfernungsmenge ist definiert als* $\{\Pi(f, B) \setminus P \mid P \in B \downarrow_{\Pi} f\}$

Ginge es nur um *irgendeine* Revision, wäre es damit getan: die minimale Entfernungsmenge wird aus der (meistens nicht abgeschlossenen) Theorie B gelöscht, f wird nicht mehr abgeleitet. Nun geht es aber um die *minimale* Revision. Dafür muß man sich die Substitutionen ansehen. Eine Klausel muß nicht gleich gelöscht werden. Es reicht, wenn ihre Anwendbarkeit eingeschränkt wird. Deshalb wurden die Substitutionen in Π aufgesammelt.

Definition 101: Instanzen einer Klausel. *Die Instanzen* $I(C, P)$ *der* $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$ *in der Klausel* C *in* P *sind die* $\{t_1, \ldots, t_n\}$.

In dem Beispiel gibt es in P_1 und P_2 nur jeweils eine Klausel. Die Instanzen sind $\emptyset$ und a. Für diese Instanzen soll die Klausel nicht mehr gelten. An den Anfang der Klausel wird das Literal gehängt:

$$vars(C) \notin I(C, P)$$

In unserem Beispiel kann die einzige Klausel in P_2 so eingeschränkt werden:

$$X \notin \{a\}, q(X) \leftarrow p(X)$$

Obendrein wurde noch die Wiederherstellbarkeit gefordert. Diese ist so noch nicht gewährleistet. Nehmen wir an, P_1 sei gelöscht worden. Wenn jetzt $q(a)$ wieder gelten soll, bekommen wir $p(a)$ nicht zurück. Deshalb fügt die minimale Basisrevision bei einer variablenfreien Klausel f in die Prämisse ein. Im Beispiel:

$$p(a) \leftarrow q(a)$$

In nicht variablenfreien Klauseln wird f als zusätzliche Prämisse eingefügt und die gesamte Prämisse so substituiert, wie es in der Anwendungsmenge festgehalten wurde. Bei P_2 würde also zusätzlich zu der spezialisierten Regel noch eingetragen:

$$q(a) \leftarrow p(a), q(a).$$

In diesem Beispiel ist dies redundant und gelangt nicht in die Theorie.

Algorithmus 102: Minimale Basisrevision. Die minimale Basisrevision revidiert eine (meist nicht abgeschlossene) Theorie B bezüglich f folgendermaßen:

- Ableitungen für f finden und daraus die Anwendungsmenge $\Pi(f, B)$ bestimmen;
- falls f nicht abgeleitet werden kann, ist B das Ergebnis der Revision; ansonsten:
  - die maximalen Anwendungsmengen $B \downarrow_{\Pi} f$ bestimmen und mit $\gamma(B \downarrow_{\Pi})$ daraus auswählen;

- das Komplement der ausgewählten maximalen Anwendungsmenge(n), die minimalen Entfernungsmengen, bestimmen;
- aus den minimalen Entfernungsmengen eine auswählen und für jede Aussage in der minimalen Entfernungsmenge die Spezialisierung durchführen:
  - wenn dies möglich ist, werden Klauseln durch zusätzliche Literale, die die Substitution einschränken, spezialisiert; wenn eine Klausel nicht durch Beschränkung erlaubter Substitutionen spezialisiert werden kann, wird sie gelöscht;
  - eine zusätzliche Regel wird eingefügt, die f als zusätzliche (oder einzige) Prämisse enthält – diese Regel ist substituiert mit σ aus der Anwendungsmenge für f.

Dies Verfahren ist einsetzbar für inkrementelles Lernen ebenso wie für die Wartung von Wissensbasen. Die Auswahl aus den maximalen Anwendungsmengen kann nach Gesichtspunkten wie dem Gebrauch der vorkommenden Klauselköpfe in anderen Regeln der Wissensbasis oder auch durch den Benutzer erfolgen. Eine interessante Erweiterung hat Stefan Wrobel dadurch vorgenommen, daß er für große Ausnahmemengen von Instanzen eine Charakterisierung durch "normale" Literale einführt. Die Instanzen, bei denen eine Regel die gewünschte Ableitung liefert, werden von den Instanzen, die in dem Ausnahmeliteral aufgesammelt wurden, abgegrenzt. Daraus ergibt sich eine klassische Lernaufgabe mit positiven und negativen Beispielen, in anderen Worten: ein Bedarf für die Begriffsbildung. Die Revision liefert also einen Grund für die Aggregation von Instanzen. Gibt es noch kein Prädikat in der Signatur, mit dem der Unterschied zwischen erfolgreichen Regelanwendungen und Ausnahmen ausgedrückt werden kann, so fehlt offensichtlich ein Ausdrucksmittel. Die Signatur wird durch ein neues Prädikat erweitert und dieses wird charakterisiert. Damit ist die Wissensrevision oder bedarfsgestützte Spezialisierung eingebettet in ein reichhaltiges Lernszenario, wie es die Kognitionswissenschaft entwirft und wir sind wieder am Anfang des Skriptes angekommen.

6 Schluß

Am Ende einer Vorlesung stellen sich die Fragen:

1. Wofür ist dieser Stoff wichtig? Was kann man damit anfangen?
2. Was von all den vielen Dingen ist zentral und was ist Illustration und technische Einzelheit?[19]
3. Wo ist dieser Stoff im Fachgebiet angesiedelt? Wie repräsentativ ist er für das Fachgebiet?

Ich will auf diese Fragen im Skript kurz, mündlich gern auch ausführlicher eingehen.

Ad 1. Der Stoff ist zum einen interessant, um die Lernfähigkeit auch beim Menschen genauer zu fassen, oder wenigstens zu sehen, wo seine formale Beschreibung schwierig wird und warum. Die Arten, Lernen zu betrachten und zu formalisieren, sind eng verbunden mit

[19] Diese Frage kleiden Studierende gern in die eigentlich untergeordnete: Was muß ich für die Prüfung wissen?

unseren intuitiven Auffassungen von der Tätigkeit (?), die wir mit dem Wort *Lernen* meinen. Insofern geben uns formale Beschreibungen des Lernens Aufschluß über unser eigenes Denken. Dieses Erkenntnisinteresse findet eine Entsprechung in praktischen Anwendungen der Informatik. Lerntechniken sind überall einsetzbar, wo mit expliziten Informationen gearbeitet wird. Jedes Programm kann durch Lernfähigkeit verbessert werden. Dies beginnt zu Beispiel damit, daß die Anti-Unifikation (in Prolog ein built-in Prädikat, aber in jeder Sprache auch rasch selbst implementiert) in das Repertoire der Programmierung aufgenommen wird. Und natürlich kann es als Komponente bei Mensch-Maschine-Schnittstellen (die leider immer noch zu selten über wenigstens adaptive Fähigkeiten verfügen), bei Planungssystemen, bei wissensbasierten Systemen eingesetzt werden. Jedes System wird durch Lernfähigkeit besser! Mein Anliegen ist es, die Basistechniken zu vermitteln, ohne sie an die gerade modischen Fragestellungen zu binden: heute ist die Wissensentdeckung in Datenbanken en vogue, aber morgen wird es etwas sein, was wir noch gar nicht kennen. Anders herum: die Grundlagen, die sich hinter der Wissensentdeckung verbergen, wurden vor Jahren gelegt und unter ganz anderen Stichworten. Folglich müssen heute die Grundlagen für noch unbekannte zukünftige Anwendungen gelegt werden. Es ist meine Überzeugung, daß nur die Kenntnis der Methoden und Tücken Expertise ausmacht – die gerade stereotype Verknüpfung von einer Methode mit einer Anwendungsklasse verstellt eher den Blick auf die Lösung neuer Aufgaben. Mein Argument hier läßt sich zusammenfassen zu den Punkten: Es gibt sehr viele Anwendungsmöglichkeiten in allen Feldern der Informationstechnologie und der menschen-orientierten Formalisierung. Diese Anwendungsmöglichkeiten kann nur erkennen, wer die Grundlagen beherrscht.

Ad 2. In der Vorlesung werden verschiedene Sichtweisen oder Paradigmen auf/für das Lernen vorgestellt: Lernen als Funktionsapproximation, als Datenmodellierung, als Identifikation (der Wahrheit) im Grenzwert. Es wurden außerdem verschiedene Repräsentationsformalismen verwendet: Funktionen, Attribut-Werte-Repräsentation und Prädikatenlogik mit Einschränkungen. Das Lernen selbst wurde in zwei verschiedene Lernaufgaben unterteilt: das Lernen von Begriffen (aus Beispielen oder aus Beobachtungen) und die Wissensentdeckung. Damit sind die drei wichtigsten Dimensionen des maschinellen Lernens herausgearbeitet: Paradigma, Repräsentationsformalismus und Lernaufgabe. Der Index zeigt deutlich an, was ich für wesentliche Inhalte halte: zu technischen Einzelheiten habe ich keine Stichwörter vergeben. Wer zu jedem Stichwort im Index die Formel und den Beweggrund für dies Stichwort weiß, hat die Prüfung sehr gut bestanden. Dabei meine ich mit "Beweggrund" sowohl den Grund, warum eine technische Notation eingeführt wurde, wie auch die Tücken, die damit verbunden sind.

Ad 3. Mit den eben erwähnten drei Dimensionen hoffe ich, ein Raster geschaffen zu haben, das verschiedene Modetrends überdauert. Natürlich gibt es in jedem Feld dieses drei-dimensionalen Raumes viel mehr Methoden, als sie hier behandelt werden konnten. Beispielsweise gehören dem Paradigma der Funktionsapproximation, das in den USA eine dominante Rolle spielt, das zudem gut zur statistischen Denkweise paßt, die in der Datenanalyse dominiert, weit mehr Verfahren an, als nur neuronale Netze! Das *reinforcement* oder *Q-learning* ist gerade ein besonders aktives Feld und wurde hier ganz ausgelassen. Auch die Bayes Netzwerke sind im Sinne diese Paradigmas zu betrachten und zur Zeit außerordentlich beliebt. Ich verweise hier auf das ausgezeichnete Lehrbuch von Tom Mitchell [Mitchell, 1997a]. Daß ich hier dem der europäischen Tradition entstammenden Paradig-

ma der Datenmodellierung breiteren Raum eingeräumt habe, hat als Grund zum einen das Fehlen eines so hervorragend ausgearbeiteten Lehrbuchs. Zum anderen spricht hier meine eigene erkenntnistheoretische Überzeugung. Aber auch im Bereich der induktiven logischen Programmierung habe ich viele Methoden und Diskussionen ausgelassen. Das Paradigma der Identifikation im Grenzwert schließlich ist besonders dünn besetzt, obwohl ich gerade die Arbeiten von Dana Angluin sehr schätze. Mir erscheint es inzwischen als ausgereift genug, um in den Kanon der theoretischen Informatik aufgenommen zu werden. Damit ist die Einordnung innerhalb der Informatik angesprochen. Maschinelles Lernen bewegt sich zwischen theoretischer und angewandter Informatik und hat obendrein noch die Kognitionswissenschaft als Bezugspunkt. Ich habe nur solche theoretischen Bezüge in die Vorlesung aufgenommen, die entweder als Standardwissen von jedem erwartet werden, oder die wahrhaftig praktisch sind, weil man sie als Grundlage für beliebig viele Anwendungen nehmen kann. Theoretische Erkenntnisse um der Theorie willen habe ich allesamt weggelassen.

Literatur

[Adriaans et al., 1993] Adriaans, P., Janssen, S. und Nomden, E. (1993). Effective Identification of Semantic Categories in Curriculum Texts by Means of Cluster Analysis. In *European Conference on Machine Learning (ECML)*.

[Agrawal et al., 1996] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. und Verkamo, A. I. (1996). Fast Discovery of Association Rules. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. und Uthurusamy, R., Herausgeber, *Advances in Knowledge Discovery and Data Mining*, AAAI Press Series in Computer Science, Kapitel: 12, Seiten 277–296. A Bradford Book, The MIT Press, Cambridge Massachusetts, London England.

[Alchourron et al., 1985] Alchourron, Grädenfors und Makinson (1985). On the logic of the theory change - partial meet contraction and revision functions. *Journal o Symbolic Logic*, 50:510–530.

[Anthony und Biggs, 1992] Anthony, M. und Biggs, N. (1992). *Computational Learning Theory.* Univ. Press, Cambridge, Mass. ua.

[Balabanovic und Shoham, 1995] Balabanovic, M. und Shoham, Y. (1995). Learning Information Retrieval Agents: Experiments with Automated Web Browsing. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments.* AAAI-Press.

[Barsalou, 1983] Barsalou, L. W. (1983). Ad hoc categories. *Memory and Cognition*, 11(3):211–227.

[Blumer et al., 1990] Blumer, A., Ehrenfeucht, A., Haussler, D. und Warmuth, M. (1990). Learnability and the Vapnik–Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965.

[Brockhausen und Morik, 1997] Brockhausen, P. und Morik, K. (1997). Wissensentdeckung in relationalen Datenbanken: Eine Herausforderung für das maschinelle Lernen. In Nakhaeizadeh, G., Herausgeber, *Data Mining, theoretische Aspekte und Anwendungen*, Wirtschaftsinformatik. Physica Verlag. erscheint demnächst.

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

[Carey, 1985] Carey, S. (1985). *Conceptual change in childhood.* MIT Press, Boston.

[De Raedt, 1991] De Raedt, L. (1991). *Interactive Concept-Learning.* Ph. D. Katholieke Univ. Leuven.

[Dillmann, 1988] Dillmann, R. (1988). *Lernende Roboter - Aspekte maschinellen Lernens.* Springer.

[Džeroski et al., 1992] Džeroski, S., Muggleton, S. und Russell, S. (1992). PAC-Learnability of Determinate Logic Programs. *Procs. of 5th COLT*, Seiten 128–135.

[Emde et al., 1983] Emde, W., Habel, C. U. und Rollinger, C.-R. (1983). The Discovery of the Equator or Concept Driven Learning. In *IJCAI-83*, Seiten 455–458, Los Altos, CA. Morgan Kaufman.

[Fargier, 1991] Fargier, H. (1991). Using MOBAL for Security Policy Management - Overview and Remarks. Technical Report AAR-40-1, Alcatel Alsthom Recherche, Marcoussis.

[Fisher, 1987] Fisher, D. H. (1987). Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2(Douglas H. Fisher):139–172.

[Gärdenfors, 1988] Gärdenfors, P. (1988). *Knowledge in Flux — Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge, MA.

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

[Giordana und Saitta, 1990] Giordana, A. und Saitta, L. (1990). Abstraction – A General Framework for Learning. In *Procs. AAAI – Workshop on Automatic Generation of Approximations and Abstractions*, Seiten 245–256.

[Haussler, 1988] Haussler, D. (1988). Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. *Artificial Intelligence*, 36:177–221.

[Hoffmann, 1991] Hoffmann, A. (1991). Die Theorie des Lernbaren - ein Überblick. *KI*, (1):7–11.

[Hoffmann, 1997] Hoffmann, A. (1997). *Paradigms of Artificial Intelligence – in Defense of Symbolic AI.* Habilitation an der TU-Berlin.

[Joachims et al., 1997] Joachims, T., Freitag, D. und Mitchell, T. (1997). WebWatcher: A Tour Guide for the World Wide Web. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Seiten 770–775.

[Jung, 1992] Jung, B. (1992). Generalisierungsoperatoren in der Induktiven Logischen Programmierung. Diplomarbeit Nr. 940, Universität Stuttgart, Stuttgart.

[Kaelbling, 1991] Kaelbling, L. P. (1991). Specifying Complex Behavior for Computer Agents. In Steels, L. und Smith, B., Herausgeber, *Proc. of the 8th Conf. on AI and Simulation of Behavior*, Seiten 94–105.

[Kearns, 1990] Kearns, M. (1990). *The Computational Complexity of Machine Learning*. ACM Distinguished Dissertation. The MIT Press.

[Kearns und Vazirani, 1994a] Kearns, M. und Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

[Kearns und Vazirani, 1994b] Kearns, M. und Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

[Keil und Kelly, 1987] Keil, F. und Kelly, M. (1987). Developmental Changes in Category Structure. In Harnad, S., Herausgeber, *Categorical Perception*, Kapitel: 6, Seiten 491–510. Cambridge University Press.

[Kietz, 1993] Kietz, J.-U. (1993). Some Lower Bounds for the Computational Complexity of Inductive Logic Programming. In Brazdil, P., Herausgeber, *Proceedings of the Sixth European Conference on Machine Learning ECML-93*, Lecture Notes in Artificial Intelligence, Seiten 115–123, Berlin, Heidelberg, New York. Springer. Also available as Arbeitspapiere der GMD No. 718, 1992.

[Kietz, 1996] Kietz, J. U. (1996). *Induktive Analyse relationaler Daten*. PhD thesis, Technische Universität Berlin, Berlin.

[Kietz und Morik, 1994] Kietz, J.-U. und Morik, K. (1994). A Polynomial Approach to the Constructive Induction of Structural Knowledge. *Machine Learning*, 14(2):193 – 217.

[Kietz und Wrobel, 1992] Kietz, J.-U. und Wrobel, S. (1992). Controlling the Complexity of Learning in Logic through Syntactic and Task–Oriented Models. In Muggleton, S., Herausgeber, *Inductive Logic Programming*, Kapitel: 16, Seiten 335–360. Academic Press, London.

[Klingspor et al., 1996] Klingspor, V., Morik, K. und Rieger, A. (1996). Learning Concepts from Sensor Data of a Mobile Robot. *Machine Learning*, 23(2/3):305–332.

[Kodratoff und Ganascia, 1986] Kodratoff, Y. und Ganascia, J.-G. (1986). Improving the generalization step in learning. In Michalski, R. S., Carbonell, J. G. und Mitchell, T. M., Herausgeber, *Machine Learning - An Artificial Intelligence Approach*, Kapitel: 9, Seiten 215–244. Morgan Kaufman.

[Land, 1983] Land, E. (1983). Verweis ohne Titel in F.J. Varela 1988, Cogn. Science - A Cartography of Current Ideas. *Proc. Natl. Acad. Sci.*, 80.

[Lang, 1995] Lang, K. (1995). NewsWeeder: Learning to Filter Net News. In *Procs. of the 12th International Conference on Machine Learning*. Morgan Kaufmann.

[Lavrač und Džeroski, 1994] Lavrač, N. und Džeroski, S. (1994). *Inductive Logic Programming — Techniques and Applications*. Nummer 148 in Artificial Intelligence. Ellis Horwood, New York.

[Lebowitz, 1987] Lebowitz, M. (1987). Experiments with Incremental Concept Formation: UNIMEM. *Machine Learning*, 2:103–138.

[Lenneberg, 1967] Lenneberg, E. (1967). *Biological Foundations of Language*. New York.

[Lieberman, 1995] Lieberman, H. (1995). Letizia: An Agent That Assists Web Browsing. In *International Joint Conference on Artificial Intelligence, Montreal*.

[Michalski, 1986] Michalski, R. (1986). Understanding the Nature of Learning. In Michalski, Carbonell und Mitchell, Herausgeber, *Machine Learning - An Artificial Intelligence Approach*. Morgan Kaufmann, Los Altos, California.

[Michalski und Stepp, 1986] Michalski, R. und Stepp, R. (1986). Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects. In Michalski, R., Carbonell, J. und Mitchell, T., Herausgeber, *Machine Learning - An Artificial Intelligence Approach Vol II*, Seiten 471–498. Tioga Publishing Company, Los Altos.

[Michalski, 1983] Michalski, R. S. (1983). A Theory and Methodology of Inductive Learning. In Michalski, R. S., Carbonell, J. G. und Mitchell, T. M., Herausgeber, *Machine Learning — An Artificial Intelligence Approach*, Ausgabe 1, Kapitel: 4, Seiten 83–135. Morgan Kaufmann, Palo Alto, CA.

[Michie, 1989] Michie, D. (1989). New Commercial Oppertunities Using Information Technology. In Brauer, F., Herausgeber, *Wissensbasierte Systeme*, Seiten 64–71, Berlin, Heidelberg, New York, Tokio. Springer.

[Mitchell, 1997a] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

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

[Mitchell, 1997b] Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill.

[Morik, 1987] Morik, K. (1987). Acquiring Domain Models. *Intern. Journal of Man Machine Studies*, 26:93–104. auch erschienen in Knowledge Acquisition Tools for Expert Systems, volume 2, J. Boose, B. Gaines, Herausgeber, Academic Press, 1988.

[Morik und Kietz, 1989] Morik, K. und Kietz, J.-U. (1989). A Bootstrapping Approach to Conceptual Clustering. In *Proc. Sixth Intern. Workshop on Machine Learning*.

[Morik und Lindner, 1995] Morik, K. und Lindner, G. (1995). Coupling a Relational Learning Algorithm with a Database System. In Kodratoff, Y., Nakhaeizadeh, G. und Taylor, C., Herausgeber, *Statistics, Machine Learning and Knowledge Discovery in Databases*, Crete, Greece. ML–Net Familiarization Workshop.

[Morik et al., 1994] Morik, K., Potamias, G., Moustakis, V. S. und Charissis, G. (1994). Knowledgeable Learning Using MOBAL: A Medical Case Study. *Applied Artificial Intelligence*, 8(4):579–592.

[Morik und Rieger, 1993] Morik, K. und Rieger, A. (1993). Learning Action-Oriented Perceptual Features for Robot Navigation. In Giordana, A., Herausgeber, *Learning Robots - Proceedings of ECML Workshop*.

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

[Muggleton, 1991] Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, 8(4):295–318.

[Muggleton und Bain, 1992] Muggleton, S. und Bain, M. (1992). Non-Monotonic Learning. In Muggleton, S., Herausgeber, *Inductive Logic Programming*, Kapitel: 13, Seiten 281–298. Academic Press, London.

[Muggleton und Buntine, 1988] Muggleton, S. und Buntine, W. (1988). Machine Invention of First-order Predicates by Inverting Resolution. In *Proc. Fifth Intern. Conf. on Machine Learning*, Los Altos, CA. Morgan Kaufman.

[Muggleton und Feng, 1992] Muggleton, S. und Feng, C. (1992). Efficient induction of logic programs. In Muggleton, S., Herausgeber, *Inductive Logic Programming*, Kapitel: 13, Seiten 281–298. Academic Press, London.

[Mühlenbrock und Morik, 1998] Mühlenbrock, M. und Morik, K. (1998). Rekonstruktion des Erwerbs einer Begriffsstruktur zur Erlärung des Tag/Nacht-Zyklus. *Kognitionswissenschaft*, 2.

[Murphy und Medin, 1985] Murphy, G. L. und Medin, D. L. (July 1985). The Role of Theories in Conceptual Coherence. *Psychological Review*, 92(3):289–316.

[Natarajan, 1991] Natarajan, B. K. (1991). *Machine Learning. A Theoretical Approach*. Morgan Kaufmann, San Mateo, CA.

[Nebel, 1989] Nebel, B. (1989). KL–One–basierte, hybride Repräsentationssysteme. In Metzing, D., Herausgeber, *13th German Workshop on Artificial Intelligence GWAI–89*, Nummer 216 in Informatik Fachberichte, Seite 484, Berlin u.a. Springer.

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

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

[Pazzani et al., 1996] Pazzani, M., Muramatsu, J. und Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. In *AAAI Conference, Portland*.

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

[Quine, 1977] Quine, W. V. (1977). Natural Kinds. In Schwartz, Herausgeber, *Naming, Necessity, and Natural Kinds*. Cornell Univ. Press.

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

[Reimann und Spada, 1996] Reimann, P. und Spada, H., Herausgeber (1996). *Learning in Humans and Machines - Towards an Interdisciplinary Learning Science*. Elsevier, London.

[Reimer und Pohl, 1991] Reimer, U. und Pohl, K. (1991). Automatische Wissensakquisition aus Texten. *KI*, Seiten 45–51.

[Rouveirol und Puget, 1990] Rouveirol, C. und Puget, J. F. (1990). Beyond Inversion of Resolution. In Porter, B. und Mooney, R., Herausgeber, *Proc. Seventh Intern. Conf. on Machine Learning*, Seiten 122–130, Palo Alto, CA. Morgan Kaufmann.

[Russell und Norvig, 1995] Russell, S. J. und Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, USA.

[Scott, 1983] Scott, P. (1983). Learning: The Construction of a Posteriori Knowledge Structures. In *AAAI-83*, Washington.

[Segre, 1988] Segre, A. (1988). *Machine Learning of Robot Assembly Plans*. Kluwer, Boston.

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

[Shapiro, 1983] Shapiro, E. Y. (1983). *Algorithmic Program Debugging*. ACM Distinguished Doctoral Dissertations. The MIT Press, Cambridge, Mass.

[Simon, 1978] Simon, H. (1978). Acht Vorlesungen über Psychologie. gehalten an der Univ. Hamburg, Fachbereich Psychologie.

[Simon, 1983] Simon, H. (1983). Why Should Machines Learn? In Michalski, R., Carbonell, J. und Mitchell, T., Herausgeber, *Machine Learning: An Artificial Intelligence Approach*, Seiten 25–38. Tioga, Palo Alto, CA.

[Sommer et al., 1994] Sommer, E., Morik, K., Andre, J.-M. und Uszinsky, M. (1994). What Online Machine Learning Can Do For Knowledge Acquisition - A Case Study. *Knowledge Acquisition*, 6(4):435 – 460.

[Turing, 1987] Turing, A. (1987). Computing Machinery and Intelligence, in Mind 59, 1950. In Dotzler und Kittler, Herausgeber, *Alan Turing - Intelligence Service*. Brinkmann und Bose.

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

[van der Laag, 1995] van der Laag, P. (1995). *An Analysis of Refinement Operators in Inductive Logic Programming*. Tinbergen Institute Research Series, Rotterdam.

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

[Wirth, 1989] Wirth, R. (1989). Completing Logic Programs by Inverse Resolution. In Morik, K., Herausgeber, *Proc. Fourth European Working Session on Learning (EWSL)*, Seiten 239–250, London/San Mateo, CA. Pitman/Morgan Kaufmann.

[Wrobel, 1989] Wrobel, S. (1989). Demand-Driven Concept Formation. In Morik, K., Herausgeber, *Knowledge Representation and Organization in Machine Learning*, Seiten 289–319. Springer, Berlin, Tokio, New York.

[Wrobel, 1991] Wrobel, S. (1991). Towards a Model of Grounded Concept Formation. In *Procs. 12th IJCAI*, Seiten 712–719, Los Altos, CA. Morgan Kaufman.

[Wrobel, 1993] Wrobel, S. (1993). On the Proper Definition of Minimality in Specialization and Theory Revision. In Brazdil, P., Herausgeber, *Machine Learning - ECML-93*, Seiten 65–82. Springer, Berlin, Heidelberg, New York.

[Wrobel, 1994] Wrobel, S. (1994). *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, Dordrecht.

[Zercher, 1991] Zercher, K. (1991). Wissensintensives Lernen von Regeln zur Fehlerdiagnose von Robotermontage. *KI*, Seiten 40–44.

Index

This article was processed using the LaTeX macro package with LLNCS style