4 Wissensbasierte Systeme

Wir haben Wissensrepräsentationsformalismen mit ihren Operatoren kennengelernt. Jetzt soll es darum gehen, wie solche Formalismen in komplexen Systemen eingesetzt werden. Anstatt uns auf der Ebene der Interpreter mit einzelnen Formeln auseinanderzusetzen, wollen wir jetzt das Verhalten von ganzen Regelmengen untersuchen: wie sind Regelmengen für verschiedene Problemklassen am besten zu organisieren? Welche Problemklassen gibt es? Wie können wir sie beschreiben? Nach einem kurzen historischen Abriß des Weges von klassischen Expertensystemen hin zu modernen wissensbasierten Systemen stelle ich zwei Methoden der Organisation von Problemlösungen detailliert vor: die heuristische Klassifikation und die überdeckende Diagnose. Beide betreffen die Problemklasse der Diagnose oder Klassifikation.[23] Ich gehe dann auf den Wissenserwerb für wissensbasierte Systeme ein, indem ich zwei Paradigmen vorstelle: den modellbasierten Ansatz und das "schlampige" Modellieren.

4.1 Expertensysteme - ein historischer Abriß

Ein Expertensystem besteht nur in seinem Kern aus einer Problemlösungskomponente. Außer der Problemlösungskomponente muß ein Expertensystem zusätzlich noch über einige Komponenten verfügen, die für den Endbenutzer wichtig sind. So muß es eine **Dialogkomponente** geben, die mit dem Endbenutzer interagiert. Einerseits muß das System Daten erfragen oder über ein Formular annehmen können. Andererseits muß das System dem Benutzer "erklären" können, wie es zu einer Problemlösung kam. Für denjenigen, der das Expertensystem aufbaut bzw. wartet, ist eine **Regelerwerbskomponente** wichtig. Eigentlich ist auch eine **Wartungskomponente** wichtig, die die Konsistenz und Redundanzfreiheit von Regelmengen prüft. Solche Werkzeuge gehören aber noch nicht zum Standard.

MYCIN, ein System zur medizinischen Diagnose, wurde an der Universität Stanford im Heuristic Programming Project entwickelt (Shortliffe 1976). MYCIN diagnostiziert Krankheiten anhand von Bakterienbefunden. Es hat ein Produktionensystem als Repräsentationsformalismus. Es ist Bezugspunkt für die Diskussion um klassische Expertensysteme. Von MYCIN ausgehend, können wir die Entwicklung hin zu neueren Expertensystemen verfolgen (siehe Clancey 1983). Diese Entwicklung wurde durch drei Aspekte vorangetrieben:

- Der Aufbau eines Expertensystems für eine Anwendung soll leichter und schneller vonstatten gehen.
- Die Erklärungen an den Benutzer sollen übersichtlicher und verständlicher sein.
- Ein Expertensystem soll auch als Lehrsystem benutzbar sein, dessen Wissensbasis von Studierenden für selbständiges Lernen genutzt werden kann.

[23] Ich verwende die beiden Begriffe Diagnose und Klassifikation synonym, wie es z.B. auch Frank Puppe tut. Einige Wissenschaftler machen allerdings Unterschiede, so daß man bei jedem Artikel genau gucken muß, wie der Autor seine Begriffe definiert!

Der erste Schritt der Entwicklung führte von MYCIN zu EMYCIN, der **Expertensystemhülle**. Statt für jede Anwendung erneut den Interpreter, die Dialogkomponente und die Erklärungskomponente zu schreiben, wird einmalig eine Expertensystemhülle entwickelt. Eine Expertensystem-Hülle ist eine Entwicklungsumgebung zur Erstellung eines bestimmten Expertensystems für eine Anwendung. Die Hülle enthält typischerweise einen Interpreter für die Problemlösung, eine Dialogkomponente und Erwerbs-Werkzeuge. Die Regeln und Fragen an den Endbenutzer müssen dann anwendungsspezifisch eingegeben werden. Damit ist der Aufbau eines Expertensystems schon etwas leichter geworden, als wenn alle Teile des Systems jedes Mal neu programmiert werden müßten.

Der nächste Schritt war die Trennung von Wissensarten. Ein Endbenutzer muß das Systemverhalten nachvollziehen können. Wenn das System eine Frage stellt oder wenn das System eine Problemlösung anbietet, muß es ausgeben können, wie es dazu kam. Im einfachsten Falle gibt die sogenannte **Erklärungskomponente** die Regeln und ihre Verkettung an, die zur Lösung oder zur Frage an den Benutzer geführt haben. Wie verständlich diese Ausgabe ist, hängt dann von den Regeln ab. Enthalten die Regeln durcheinander gemischt verschiedene Typen von Bedingungen, so ergibt die Rückverfolgung der Regelverkettung auch ein Durcheinander. Berühmtes Beispiel solcher Vermischung ist die folgende Regel des Expertensystems MYCIN (Shortliffe 1976), die Clancey analysiert hat (Clancey 1983:236):

IF Typ der Infektion ist Meningitis &
keine Labordaten sind vorhanden &
der Typ der Meningitis ist bakteroid &
der Patient ist älter als 17 Jahre &
der Patient ist Alkoholiker
THEN Evidenz für E.Coli (0.2) und Diplococcus (0.3)

Hier sind drei Bedingungstypen vermischt:

- Anwendbarkeitsbedingungen für die Regel (Meningitis, Verfügbarkeit der Labordaten und Meningitistyp)
- sachbasierte Bedingungen (Alkoholismus begünstigt E.Coli und Diplococcus)
- dialogbasierte Bedingungen (frage keine Kinder, ob sie Alkoholiker sind)

Werden diese verschiedenen Bedingungstypen unterschieden, so kann man zu sinnvollen Systemausgaben bei verschiedenen Fragen des Benutzers kommen:

- Warum wird jetzt diese Regel angewandt? -

  Weil die Infektion Meningitis und vom Typ bakteroid ist und es keine Labordaten gibt.

- Warum soll festgestellt werden, ob der Patient Alkoholiker ist? -

  Weil das für E.Coli und Diplococcus spricht.

- Warum wird jetzt nach dem Alter gefragt? -

  Weil festgestellt werden soll, ob der Patient Alkoholiker ist und Kinder keine Alkoholiker sind.

Die sachbasierten Bedingungen sind oft - wie in diesem Fall auch - nicht ausreichend explizit. Es ist in der Regel nicht angegeben, daß Alkoholismus Infektionen mit eigenen Darmbakterien (eben E.Coli) begünstigt, weil Alkohol die Immunabwehr schwächt.

Die Fragen beziehen sich auf ein Sachwissen, das sich wie jedes Wissen ändern kann. Falls Kinder häufig zu Alkoholikern würden, wäre die Frage nach dem Alter nicht mehr angemessen. Auch dieses Wissen sollte sich so explizit in einer eigenen Regel finden lassen, daß es überprüfbar wird.

Der Anwender, der die Regelmenge wartet, stellt dieselben Anforderungen wie der Endbenutzer. Auch für die Wartung müssen Ergebnisse nachvollziehbar sein. Die Trennung der verschiedenen Bedingungsarten ist für die Änderbarkeit einer Regelmenge entscheidend. Wenn sich Wissen über ein Gebiet ändert, müssen die Änderungen entsprechend auch im System vorgenommen werden. Wenn man aber die explizite Angabe nicht im System hat, in welche Regeln gerade dieses Wissen eingeflossen ist, kann das System nur sehr schwer gewartet werden. Wenn z.B. der Alkoholismus bei Kindern zunimmt, so sollte es nur nötig sein, eine Regel zu ändern, die ausdrücklich vom Alkoholismus bei Kindern handelt. Das wäre eine Regel, die den Verdacht auf Alkoholismus erhöht oder erniedrigt und dafür bestimmte Fakten wie Alter heranzieht. Stattdessen mußte man bei MYCIN alle E.Coli-Regeln durchgehen. Ebenso erlaubt die explizite Darstellung des Zusammenhangs zwischen Alkohol, Immunabwehr, und Bakterien erst die leichte Einbeziehung von neuen medizinischen Erkenntnissen. Stattdessen mußte man bei MYCIN alle Regeln durchsehen, ob sie indirekt auf so einem Zusammenhang beruhen. Die Trennung von Wissensarten dient also sowohl der Erklärung wie auch der Wartung von Expertensystemen.

Clancey (1986) weist daraufhin, daß sich MYCIN so verhält, als verfolge es die folgende Regel:

```
IF  der Genus bekannt ist & nicht die Spezies bekannt ist
THEN nimm die für diesen Genus wahrscheinlichste Spezies an
```

Tatsächlich gibt es aber keine derartige Heuristik irgendwo im System. Das Zusammenwirken verschiedener Regeln, die sich auf Genus und Spezies eines Mikro-Organismus beziehen, ergibt ein solches Verhalten. Wenn das Verhalten des Systems geändert werden soll, kann dies nicht mit lokalen, einsichtigen Änderungen geschehen. Der nächste Schritt in der Expertensystementwicklung war die Einführung von expliziten Problemlösungsmethoden, auf die wir im nächsten Abschnitt eingehen werden.

Die klassischen Expertensystemhüllen, die auf EMYCIN aufbauen, sind heute als Produkte auf dem Markt und werden eingesetzt. In Deutschland wurden die Expertensystem-Hüllen TWAICE (Nixdorf) und BABYLON (GMD, VW-Gedas) entwickelt. Gegenüber einem in einer üblichen Programmiersprache gefertigten System haben auch sie schon den Vorteil leichterer Änderbarkeit und besserer Inspizierbarkeit. Für einen Nicht-Informatiker sind die Regeln leichter zu überprüfen als ein Pascal-Programm. Dennoch ist der Detailliertheitsgrad immer noch zu fein, die Vermischung verschiedener Informationsarten immer noch gegeben. Die Expertensystemhülle D3 von der Gruppe um Frank Puppe (Puppe et al. 1996) ist eine moderne Entwicklungsumgebung, die strikt zwischen dem eigentlichen Sachbereichswissen und sogenanntem Basiswissen unterscheidet.

Expertensysteme haben sich in der Praxis längst durchgesetzt -- wenn auch unter einer Fülle unterschiedlicher Namen.

- Viele wissensbasierte Systeme, die als Produkte für eine bestimmte Anwendung auf dem Markt sind, tragen nicht mehr das Etikett "Expertensystem". So ist z.B. das Tippfehlerkorrektursystem GRAMMAR das meistverkaufte Expertensystem, ohne daß es als solches vermarktet wird.

- Viele Anwendungen behandeln Konfigurationsprobleme: Rechen- oder Telekommunikationsanlagen werden individuell für einen Kunden konfiguriert, wobei die Wissensbasis den aktuellen Stand all der vielen Komponenten und ihrer technischen Daten umfaßt. Ausgenutzt wird dabei, daß portable Rechner zum Kunden mitgenommen werden können.

- Wartung technischer Systeme ist ein breites Anwendungsfeld. Zum einen kann ein Berater, der am Telefon die Probleme von Kunden entgegennimmt und zu lösen versucht, von einem Expertensystem unterstützt werden. Zum anderen nehmen Wartungstechniker ein Expertensystem mit zum Kunden, wenn sie dort einen Fehler beheben sollen.

- Im Versicherungs- und Bankenwesen werden Expertensysteme eingesetzt, um Rentabilität und Kreditwürdigkeit zu prüfen. Die meisten Kreditkarten werden mithilfe von Expertensystemen auf Mißbrauch hin geprüft: in jeder Nacht werden alle Buchungen auf bestimmte Muster hin überprüft. Wird ein typisches Mißbrauchsmuster erkannt, wird der betreffende Karteninhaber angerufen, ob die Buchung wirklich ausgeführt werden soll.

- Einige Firmen speichern betriebliche Abläufe und organisatorisches Firmen-Know-How in Expertensystemen ab. Auf diese Weise werden Reorganisationen unterstützt.

- Qualitätssicherung ist ein wichtiges Anwendungsfeld für Expertensysteme. Die Prüfstände zur Endkontrolle von Autos sind oft mit einem Expertensystem ausgestattet.

- Im Krankenhaus werden Expertensysteme eingesetzt -- z.B. um die vielen erhobenen Parameter zu einem Patienten zusammenzufassen.

- Als Nachschlagewerk sind Expertensysteme z.B. im juristischen Bereich im Einsatz. Die sich gegenwärtig ständig ändernden Gesetze eines Bereiches können so nicht nur nachgeschlagen, sondern auch angewandt werden. Beispielsweise berechnet ein System die Unterhaltszahlungen, zu denen ein Elternteil verpflichtet ist, anhand der Angaben über Einkommen der Eltern und dergleichen. Das System gibt die entsprechenden Gesetzestexte als Kommentar dazu aus, so daß die Berechnung nachvollziehbar ist..

Weltweit sind gegenwärtig (März 1997) 4929 Expertensysteme erfaßt. Darin sind also nicht diejenigen Systeme enthalten, die eigentlich die Technik wissensbasierter Systeme verwenden, aber nicht das Etikett. Dafür ist auch nicht geprüft, welche der erfaßten Systeme wieder aus dem Verkehr gezogen bzw. durch das nächste Expertensystem abgelöst wurden. Die meisten Systeme wurden bisher in der Industrie (3741), die wenigsten im Handel (68) eingesetzt.

4.2 Heuristische Klassifikation

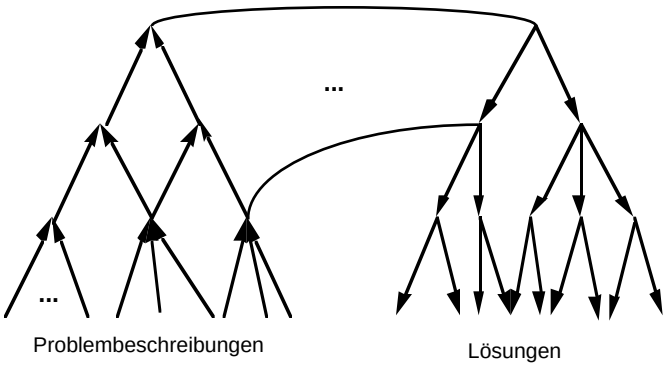
Die Analysen der ersten Expertensysteme ermöglichten ihre Beschreibung auf einer abstrakteren Ebene. Statt sich auf die Details einzelner Regeln einzulassen, kamen abstraktere Begriffe zur Beschreibung des Systemverhaltens auf. Diese abstraktere Ebene wurde insbesondere von Ron Brachman, Bill Swartout, Bill Clancey und Allen Newell formuliert. Sie wird entweder Wissensebene (knowledge-level) oder **epistemische** Ebene genannt. Damit ist wohl auch schon deutlich, daß wir uns in diesem Abschnitt mit dem Beschreibungsansatz der KI befassen, indem wir ihn auf wissensasierte Systeme anwenden.

Als man die Produktionensysteme für Anwendungen in der KI entdeckte, war man glücklich darüber, das Fachvokabular eines Experten auch vom Rechner verwenden lassen zu können. Und zwar nicht als eingefrorener Text - wie das in wohl allen Programmen der Fall ist - sondern mit Regeln, die die Bedeutung der Vokabeln ebenfalls explizieren, so daß der Interpreter wirklich genau diese Bedeutung benutzt statt einer opaken Folge von ASCII-Zeichen. Es stellte sich dann heraus, daß die Regeln immer noch zu detailliert waren, verschiedene Informationsarten vermengten und in einer Weise zusammenwirkten, die nicht operational beschrieben werden konnte. Also wurde nach Strukturierungen von Regeln und Regelmengen sowie nach einem Vokabular gesucht, mit dem man die generelle Methode eines Systems verständlich und operational beschreiben kann. Diese Suche traf mit einer anderen Entwicklungslinie der KI zusammen, die ebenfalls zu höheren Beschreibungsformen führte (Brachman 1977, 1979). Die Quintessenz der Wissensebene wurde von Newell (1982, eingeführt in Kapitel 1) aufgeschrieben. Die Strukturierung und das Vokabular für die Beschreibung von Systemverhalten wurde wesentlich von Clancey (1983, 1985, 1986) erarbeitet.

Die Problemlösungsmethode, die in MYCIN mithilfe der Regeln realisiert werden sollte, bestand aus drei Schritten:

- **establish_hypotheses**: Beschwerden stoßen initiale Hypothesen an;
- **group_and_differentiate**: Übergeordnete Diagnosen und Alternativen zur initialen Diagnose werden gefunden
- **explore_and_refine**: Verfeinerung der Diagnose.

Diese Methode ließ sich bei zahlreichen Expertensystemen feststellen. Verallgemeinert man es noch weiter, so kann man es als Klassifikation beschreiben: Die Systeme abstrahieren von bestimmten Ausgangsdaten, bilden ab auf eine Hierarchie von vorbereiteten Lösungen und verfeinern eine Lösung dann. Ein solches Vorgehen wird allgemein Klassifikation genannt. Der Unterschied zur klassischen Klassifikation besteht darin, daß verschiedene Klassifikationshierarchien durch unsicheres Schließen verbunden werden.



Diese Verbindung ist in der Graphik durch Linien ohne Pfeil angegeben. Sie wird realisiert durch bestimmte Regeln, die eine Problemklassifikation als Bedingung nehmen und eine Lösungsklassifikation als Handlung haben. Die Abstraktion von bestimmten Problembeschreibungen (Eingabedaten, Phänomenen) zu Klassen von Problemen ist links mit aufwärts gerichteten Pfeilen angegeben. So wird beispielsweise in einem medizinischen Diagnosesystem von bestimmten Labordaten auf das Vorhandensein eines Symptoms und von bestimmten Symptomen auf eine Patientencharakterisierung geschlossen. Der heuristische Schritt ist dann der unsichere Schluß von dieser Patientencharakterisierung auf eine Krankheitsklasse. Die genaue Ausprägung der Krankheit bei diesem Patienten wird dann durch eine Verfeinerung in der Hierarchie der Krankheiten festgestellt. Abkürzungsregeln können schon bei der teilweisen Klassifikation eines Problems eine Lösungsklasse angeben.

An einem Beispiel wollen wir diese abstrakte Sicht mit konkreten Regeln in Beziehung zu setzen. Sinn des Beispiels ist es, die verschiedenen Ebenen, auf denen man über Expertensysteme spricht, deutlich zu machen. Dabei vereinfachen wir jede Ebene stark. Wir beschreiben die Problemlösungsmethode der heuristischen Klassifikation durch das Prologprogramm `make_diagnosis`, das `establish_hypotheses`, `group_and_differentiate` und `explore_and_refine` aufruft. Die Ebene der Problemlösungsmethode ist mit der Inferenzstrategieebene verbunden: die Problemlösungsmethode ruft Vorwärts- und Rückwärtsinferenz auf, wie sie oben bei Produktionensystemen beschrieben sind. Die Inferenzmethoden sind nur dadurch erweitert, daß sie einen Regeltyp berücksichtigen. Die Regeln sind einfache Produktionen, bei denen ein Regeltyp vorangestellt ist. Dadurch müssen nicht bei jedem Schritt alle Regeln auf ihre Anwendbarkeit hin geprüft werden, sondern lediglich diejenigen eines bestimmten Typs. Auch für die Erstellung der Wissensbasis ist es leichter, sich lediglich mit den Regeln eines Typs zur Zeit zu beschäftigen. Eine Fragekomponente ist durch eine Klausel und einige Fakten angedeutet. Wichtig ist hier nur ihre Trennung vom Sachbereichswissen. Als Sachbereich nehmen wir naives Wissen über Windpocken, Masern, Scharlach und Gelbsucht an.

```
:- ensure_loaded(library(basics)).
:- load_files(´inference.pl´). % forward und backward-inference laden!
:- dynamic(context/1).
:- dynamic(already_explored/1).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                      %%%
%%%       Problemlösungsmethode heuristische Klassifikation              %%%
%%%                                                                      %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% make_diagnosis(+Complaints, -Diagnoses)
% Das System analysiert (und ggf. erfragt) die Beschwerden und gibt
% Diagnosen aus.

make_diagnosis(Complaints, Diagnoses):-
    init(Complaints),                  % Beschwerden eintragen
    establish_hypotheses(Diagnoses).

% establish_hypotheses(-Diagnoses)

% Solange Vorwärtsinferenz aus den Fakten Abstraktionen und Hypothesen
% herleiten kann, wird dies getan. Dann werden die Diagnosen durch
% Rückwärtsinferenz geprüft. Wenn die Prüfung nicht erfolgreich ist, so
% schlägt auch die erste Klausel fehl und die nächste wird probiert:
% Fragen an den Benutzer. Danach gelingt mit dem zusätlichen Wissen
% vielleicht die erste Klausel und reicht ihre geprüften Diagnosen hoch.
% Ansonsten beendet die dritte Klausel erfolgreich (und ergebnislos) das
% Diagnostizieren.

establish_hypotheses(Diagnoses):-
    group_and_differentiate,           % Eingabedaten abstrahieren
    explore_and_refine(Diagnosis),     % Diagnosen erstellen und prüfen
    establish_hypotheses(Diagnoses).

establish_hypotheses(Diagnoses) :-     % Wenn keine Diagnose bewiesen
                                       % werden kann, kommt diese Klausel
                                       % zum Zuge:
    ask_general_questions,             % Fragen an den Benutzer
    establish_hypotheses(Diagnoses).   % Vielleicht gelingt 1. Klausel
                                       % jetzt?

establish_hypotheses(_).               % Ende


% group_and_differentiate wird hier durch Vorwärtsinferenz mit Regeln des
% Typs ‘data_abstraction’ realisiert.

group_and_differentiate :-
    closure(data_abstraction).         % Berechnung der inferentiellen
                                       % Hülle mit Vorwärtsinferenz

% explore_and_refine(-Diagnosis)

explore_and_refine(Diagnosis) :-
    focus_hypothesis(Hypothesis),            % Vorwärtsinferenz
    test_hypothesis(Hypothesis),             % Rückwärtsinferenz
    refine_hypothesis(Hypothesis,Diagnosis). % Zurückliefern der Diagnose


% focus_hypothesis(-Hypothesis)

% Eine noch nicht untersuchte Hypothese wird zurückgeliefert und als
% untersucht markiert.

focus_hypothesis(Hypothesis) :-
    forward_inference(trigger,Hypothesis),
    \+ already_explored(Hypothesis),
    assert(already_explored(Hypothesis),focus).
```

```
% test_hypothesis(+Hypothesis)

% Eine Hypothese wird durch Rückwärtsinferenz untersucht. Dazu werden nur
% Regeln des Typs ‘heuristic’ verwendet.

test_hypothesis(Hypothesis) :-
    backward_inference(heuristic,Hypothesis).


% refine_hypothesis(+Hypothesis,-Diagnosis)

% Durch Regeln des Typs ‘refinement’ wird aus der Hypothese eine Diagnose
% erstellt.

refine_hypothesis(Hypothesis,Diagnosis) :-
    rule(refinement(Hypothesis),_,then:[Diagnosis]),
    backward_inference(refinement(Hypothesis),Diagnosis).

% ask_general_questions stellt dem Benutzer Fragen.

ask_general_questions :-
    (
     ask(alter,_);
     ask(temperatur,_);
     ask(hautausschlag,_);
     ask(geschwollene_mandeln,_);
     ask(geroeteter_rachen,_);
     ask(schluckbeschwerden,_);
     ask(augen_gelb,_);
     ask(haut_gelb,_)
    ).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                          Hilfsklauseln                                %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

init([]) :-
    init_inference,
    retractall(already_explored(_)).

init([Complaint|Complaints]) :-
    init(Complaints),
    assert(context(Complaint)).

%%% Fragewissen: Attribute und ihre sprachliche Formulierung %%%

can_be_asked(alter,'Wie alt ist der Patient? ','(in Jahren)').

can_be_asked(temperatur,'Temperatur? ','(in Grad Celsius)').

can_be_asked(hautausschlag,'Liegt Hautausschlag vor? ',[ja,nein]).

can_be_asked(hautausschlag_knoten,'Besteht der Hautausschlag aus
               Knoten? ', [ja,nein]).

can_be_asked(hautausschlag_blasen,'Besteht der Hautausschlag aus
               Blasen? ',[ja,nein]).

can_be_asked(hautausschlag_flecken,'Besteht der Hautausschlag aus
               Flecken? ',[ja,nein]).

can_be_asked(hautausschlag_punkte,'Besteht der Hautausschlag aus
               Punkten? ',[ja,nein]).

can_be_asked(geschwollene_mandeln,'Sind die Mandeln geschwollen? ',
               [ja,nein]).

can_be_asked(geroeteter_rachen,'Ist der Rachen geroetet? ',[ja,nein]).
```

```
can_be_asked(schluckbeschwerden,'Liegen Schluckbeschwerden vor? ' ,
                [ja,nein]).

can_be_asked(augen_gelb,'Sind die Augen gelb verfaerbt? ',[ja,nein]).

can_be_asked(haut_gelb,'Ist die Haut gelb verfaerbt? ',[ja,nein]).

can_be_asked(labortest_masern,'Labortest auf Masern: Ergebnis? ' ,
                [positiv,negativ]).

can_be_asked(labortest_windpocken,'Labortest auf Windpocken:
                Ergebnis? ', [positiv,negativ]).

can_be_asked(labortest_streptokokken,'Labortest auf Streptokokken:
                Ergebnis? ',[positiv,negativ]).

can_be_asked(labortest_hepatitis,'Labortest auf Hepatitis: Ergebnis? ',
                [positiv,negativ]).

can_be_asked(windpocken_bereits_gehabt,'Hat der Patient die Windpocken
                bereits gehabt? ',[ja,nein]).

can_be_asked(masern_bereits_gehabt,'Hat der Patient die Masern bereits
                gehabt? ',[ja,nein]).

can_be_asked(leberfunktion,'Test der Leber: Ergebnis? ',
                [gestoert,normal]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                          %%%
%%%                        Wissensbasis: Regeln                              %%%
%%%                                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Qualitativ

rule(data_abstraction,if:[l(temperatur,37)],then:[fieber=nein]).

rule(data_abstraction,if:[ge(temperatur,37),l(temperatur,38)],
                    then:[leichtes_fieber=ja]).

rule(data_abstraction,if:[ge(temperatur,38),l(temperatur,39)],
                    then:[mittleres_fieber=ja]).

rule(data_abstraction,if:[ge(temperatur,39)],then:[hohes_fieber=ja]).


% Generalisierung

rule(data_abstraction,if:[leichtes_fieber=ja],then:[fieber=ja]).

rule(data_abstraction,if:[mittleres_fieber=ja],then:[fieber=ja]).

rule(data_abstraction,if:[hohes_fieber=ja],then:[fieber=ja]).


% Definition

rule(data_abstraction,if:[augen_gelb=ja],then:[gelbfaerbung=ja]).

rule(data_abstraction,if:[haut_gelb=ja],then:[gelbfaerbung=ja]).


% Definition

rule(data_abstraction,if:[fieber=ja,geschwollene_mandeln=ja],
                    then:[angina=ja]).

rule(data_abstraction,if:[fieber=ja,geroeteter_rachen=ja],
                    then:[angina=ja]).
```

```
rule(data_abstraction,if:[fieber=ja,schluckbeschwerden=ja],
                      then:[angina=ja]).


% Trigger

rule(trigger,if:[fieber=ja,hautausschlag=ja],then:[windpocken=ja]).

rule(trigger,if:[fieber=ja,hautausschlag=ja],then:[masern=ja]).

rule(trigger,if:[angina=ja,hautausschlag=ja],then:[scharlach=ja]).

rule(trigger,if:[gelbfaerbung=ja],then:[gelbsucht=ja]).


% Heuristische Regeln

rule(heuristic,if:[fieber=ja,hautausschlag_knoten=ja,windpocken_bereits_g
ehabt=nein],then:[windpocken=ja]).

rule(heuristic,if:[fieber=ja,hautausschlag_blasen=ja,windpocken_bereits_g
ehabt=nein],then:[windpocken=ja]).

rule(heuristic,if:[fieber=ja,hautausschlag_punkte=ja,masern_bereits_gehab
t=nein],then:[masern=ja]).

rule(heuristic,if:[hohes_fieber=ja,hautausschlag_flecken=ja,angina=ja],
then:[scharlach=ja]).

rule(heuristic,if:[gelbfaerbung=ja],then:[gelbsucht=ja]).


% Verfeinerungsregeln

rule(refinement(windpocken=ja),if:[labortest_windpocken=positiv],
                           then:[windpocken=ja]).

rule(refinement(masern=ja),if:[labortest_masern=positiv],
                           then:[masern=ja]).

rule(refinement(scharlach=ja),if:[labortest_streptokokken=positiv],
                           then:[scharlach=ja]).

rule(refinement(gelbsucht=ja),if:[gelbfaerbung=ja,alter=0],
                           then:[gelbsucht_von_neugeborenem=ja]).

rule(refinement(gelbsucht=ja),if:[labortest_hepatitis=positiv],
                           then:[gelbsucht_a=ja]).

rule(refinement(gelbsucht=ja),if:[gelbfaerbung=ja,leberfunktion=gestoert]
,                          then:[gelbsucht_b=ja]).
```

4.3 Überdeckende Diagnose

Wie die heuristische Klassifikation als Abstraktion aus dem System MYCIN hervorgegangen ist, so ist die überdeckende Diagnose aus dem System MOLE hervorgegangen (Eshelman et al. 1987). MOLE wurde zur Fehlerdiagnose bei technischen Geräten eingesetzt. Wir illustrieren hier die überdeckende Diagnose anhand desselben Sachbereichs, damit die Unterschiede und Gemeinsamkeiten deutlich werden. Während die heuristische Klassifikation von den Symptomen auf ihre Ursache zu schließen versucht, betrachtet die überdeckende Diagnose mögliche Ursachen und überprüft, wie gut diese die Symptome erklären. Sind bei der heuristischen Klassifikation die Regeln von der Art

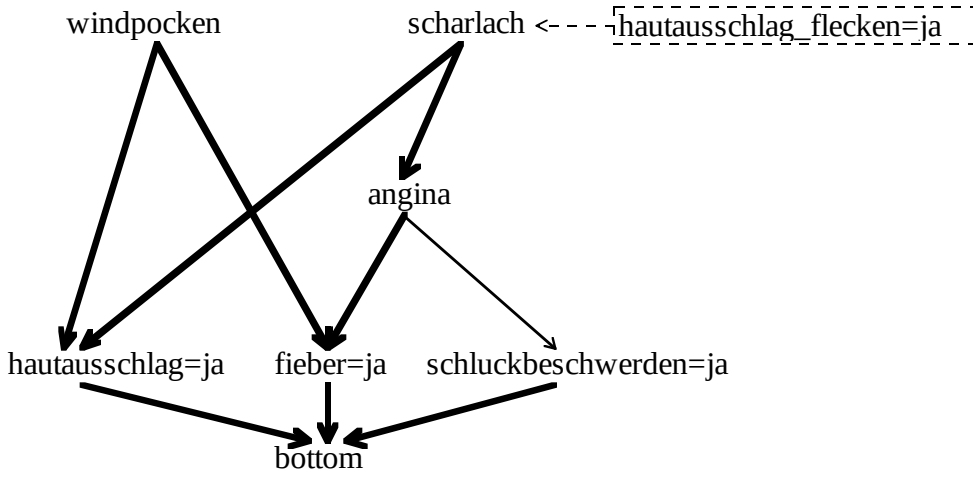
Symptom --> Diagnose

so sind sie bei der überdeckenden Diagnose gerade umgekehrt:

Diagnose --> Symptom.

Die heuristische Klassifikation modelliert die Hinweiskraft verschiedener Symptome für Diagnosen. Die überdeckende Diagnose modelliert Kausalbeziehungen zwischen Ursachen und Wirkungen. Die heuristische Klassifikation verwendet Generalisierungsbeziehungen zwischen Symptomen (Datenabstraktion) und möglichen Diagnosen. Die überdeckende Diagnose geht von einer direkten Verbindung zwischen Ursache und Wirkung aus.

Die Vorstellung bei dieser Problemlösungsmethode ist, daß es Zustände gibt, die erklärt werden müssen und Zustände, die Erklärungen darstellen.



Zwischen diesen verschiedenen Arten von Zuständen gibt es Relationen. Der Pfeil zwischen Angina und Hautausschlag steht für die Relation *will_cause*. Der Pfeil zwischen Angina und schluckbeschwerden=ja repräsentiert die Relation *may_cause*. Weiterhin gibt es positive Evidenzen für Zustände; z. B. wird ein flekkenförmiger Hautausschlag hier als Hinweis auf Scharlach benutzt. Das Ziel ist es einen Zustand (oder eine möglichst kleine Menge von Zuständen) zu finden, der alle Symptome erklärt.

Die Methode besteht aus zwei Schritten:

cover stellt mögliche Ursachen zusammen für diejenigen Zustände, die erklärt werden müssen.

differentiate sucht nach möglichst wenigen Ursachen (*exclusivity*), die möglichst viele Zustände erklären (*exhaustivity*).

Die überdeckende Diagnose verwendet ausschließlich die Rückwärtsinferenz. Allerdings werden für die Auswahl zwischen verschiedenen Diagnosen auch Regeln verwendet, die von Symptomen auf Diagnosen schließen. Dies ließe sich womöglich auch durch Vorwärtsinferenz über kausalen Regeln erreichen. Das folgende Programm ist an die Untersuchung von Guus Schreiber angelehnt (Schreiber 1992).

```
:- ensure_loaded(library(basics)).
:- ensure_loaded(library(sets)).
:- ensure_loaded(library(not)).
```

```
:- load_files('inference.pl').

:- dynamic(context/1).

:- dynamic(considered_explanation/2).

:- dynamic(accepted_explanation/2).

:- dynamic(rejected_explanation/2).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                          %%%
%%%       Problemlösungsmethode überdeckende Diagnose                        %%%
%%%                                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% make_diagnosis(+Complaints,-Cause)

% Das System analysiert Beschwerden, findet mögliche Ursachen, prüft sie
% und trägt sie in die Faktenbasis ein. Alle akzeptierten Erklärungen für
% die Beschwerden werden als Diagnose Cause zurückgeliefert.

make_diagnosis(Complaints,Cause) :-
    init(Complaints),            % Beschwerden eintragen. Aus technischen
                                 % Gründen werden Beschwerden zu
                                 % accepted_explanation eines initialen
                                 % Zustands.
    cover_and_differentiate,     % Erklärungen für Beschwerden suchen.
    findall(accepted_explanation(X,Y),accepted_explanation(X,Y),Cause).
                                 % Ergebnis in Liste aufsammeln.


% cover_and_differentiate

% Zunächst werden die möglichen Ursachen für die Beschwerden in der
% Faktenbasis gesucht (cover). Dann wird anhand der Regeln exhaustivity,
% exclusivity, anticipate1, anticipate2, prefer1, prefer2 und rule_out
% die „plausibelste“ Menge von Ursachen für die Beschwerden gesucht.

cover_and_differentiate :-
    (
     cover;         % cover
     exhaustivity;  % differentiate
     exclusivity;
     anticipate1;
     anticipate2;
     prefer1;
     prefer2;
     rule_out
    ),
    cover_and_differentiate.

cover_and_differentiate.


% cover

% Es werden alle potentiellen Erklärungen für einen Zustand S1 gesucht,
% der bereits einen anderen Zustand _S2 potentiell erklärt. Die
% Erklärungen werden als considered_explanation eingetragen, damit sie
% von den anderen Regeln überprüft werden können. Nur die Veränderung der
% Faktenbasis zählt!
```

```
cover :-
    accepted_explanation(S1,_S2),      % Finde ein S1, das noch erklärt
    \+ considered_explanation(_S3,S1)  % werden muß.
    \+ rejected_explanation(_S5,S1),   % Erklärung bereits gescheitert?
    findall(S4,(cover_relation(S4,S1), % alle potentiellen Ursachen für
                \+ rejected_explanation(S4,S1),    % S1 suchen, die noch
                                                   % nicht abgelehnt sind
                 assertt(considered_explanation(S4,S1),cover)), % und ein-
            [_|_]).                                % tragen.


% exhaustivity

% S1 ist mögliche Ursache für S2. Es gibt keine andere mögliche Ursache
% für S2. Deshalb wird S1 als Erklärung für S2 akzeptiert.

exhaustivity :-
    focus_state_pos(S2),               % Fokussieren auf S2,
    considered_explanation(S1,S2),     % eine Wirkung von S1.
    \+ (considered_explanation(S3,S2), % Es gibt keine andere mögliche
    \+ S1=S3),                         % Ursache für S2.
    assertt(accepted_explanation(S1,S2),exhaustivity).

% exclusivity

% S1 ist mögliche Ursache für S3 und S4. Auch S2 ist mögliche Ursache für
% S3, erklärt aber nichts anderes. Deshalb wird S1 als Erklärung für S3
% und S4 akzeptiert.

exclusivity :-
    focus_state_pos(S3),               % Fokussieren auf S3,
    considered_explanation(S1,S3),     % S3 ist eine Wirkung von S1.
    considered_explanation(S1,S4),     % S1 kann auch S4 erklären.
    \+ S3=S4,
    \+ (considered_explanation(S2,S3), % S2 ist eine alternative
                                       % Erklärung für S3, die aber
        considered_explanation(S2,S5), % nichts anderes erklärt.
        \+ S3=S5),
    assertt(accepted_explanation(S1,S3),exclusivity).


% anticipate2

% S1 ist mögliche Ursache für S2. S1 wird ausgeschlossen, wenn eine
% notwendige Wirkung von S1 nicht beobachtet wird.

anticipate2 :-
    focus_state_neg(S2),               % Fokussieren auf S2,
    considered_explanation(S1,S2),     % eine Wirkung von S1.
    anticipate_relation(S1,S3),        % Finde S3, eine andere Wirkung
    \+ S2=S3,                          % von S1.
    not_finding_state(S3),             % Wenn S3 nicht beobachtet wird,
    assertt(rejected_explanation(S1,S2),anticipate2), % ist S1 auch keine
                                                      % Erklärung für S2
    assertt(rejected_explanation(S1,S3),anticipate2), % und S3.
    retractt(considered_explanation(S1,S2),anticipate2),% S1 wird nicht
    retractt(considered_explanation(S1,S3),anticipate2),% mehr betrachtet
    retractt(accepted_explanation(S1,S2),anticipate2),  % und schon gar
    retractt(accepted_explanation(S1,S3),anticipate1). %nicht akzeptiert.
```

```
% anticipate1

% S1 ist mögliche Ursache für S2. S1 wird vorerst als Erklärung
% akzeptiert, weil eine notwendige Wirkung von S1 beobachtet wurde.

anticipate1 :-
    focus_state_pos(S2),
    considered_explanation(S1,S2),
    anticipate_relation(S1,S3),
    \+ S2=S3,
    finding_state(S3),
    assertt(accepted_explanation(S1,S2),anticipate1),
    assertt(accepted_explanation(S1,S3),anticipate1).

% prefer1

% S1 wird als Diagnose für S2 akzeptiert, weil per Rückwärtsinferenz
% positive Evidenz für S1 gefunden wird. Von prefer_connection werden
% Regeln der Form ‘if: Symptom then: Diagnose’ verwendet.

prefer1 :-
    focus_state_pos(S2),
    considered_explanation(S1,S2),
    prefer_connection(S1,S2),
    assertt(accepted_explanation(S1,S2),prefer1).

% prefer2

% S1 wird als Diagnose für S2 akzeptiert, weil S2 eine sichere Begründung
% für S1 ist.prefer_state verwendet Regeln der Form ‘if: Symptom then:
% Diagnose’ per Rückwärtsinferenz.

prefer2 :-
    focus_state_pos(S2),
    considered_explanation(S1,S2),
    prefer_state(S1),
    assertt(accepted_explanation(S1,S2),prefer2).

% rule_out

% Wenn es einen Befund gibt, der bei der Diagnose S1 nicht vorkommen
% kann, wird S1 nicht weiter betrachtet. rule_out_state verwendet
% Rückwärtsinferenz, um den mit S1 inkompatiblen Befund zu finden.

rule_out :-
    focus_state_neg(S2),
    considered_explanation(S1,S2),
    rule_out_state(S1),
    assertt(rejected_explanation(S1,S2),rule_out),
    retract(considered_explanation(S1,S2),rule_out),
    retract(accepted_explanation(S1,S2),rule_out).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% focus_state_pos(-Focus)

% Fokussieren auf einen Zustand, der als Ursache bereits akzeptiert
% wurde, nun aber selbst Wirkung einer tieferliegenden Ursache sein kann.
% Diese tieferliegende Ursache ist noch nicht als Erklärung akzeptiert.

focus_state_pos(Focus) :-
    accepted_explanation(S1,_S2),
    considered_explanation(_S3,S1),
    \+ accepted_explanation(_S4,S1),
    Focus=S1.

% focus_state_neg(-Focus)
```

```
% Fokussieren auf einen Zustand, der als Ursache und Wirkung bereits
% akzeptiert wurde.

focus_state_neg(Focus) :-        % Fokussieren auf einen Zustand mit
                                 % akzeptierter Erklärung
    accepted_explanation(S1,_S2),
    accepted_explanation(_S3,S1),
    Focus=S1.


% assertt/2 und retractt/2 verhalten sich wie assert/1 und retract/2. Es
% wird lediglich die zweite Argumentstelle zu Kontrolle auf den
% Bildschirm ausgegeben und die Prädikate schlagen nie fehl.


finding_state(State) :-
    accepted_explanation(State,_).

finding_state(State) :-
    get_attribute(State,_Attribute),
    istrue(State).

not_finding_state(State) :-
    rejected_explanation(State,_).

not_finding_state(State) :-
    get_attribute(State,_Attribute),
    not(istrue(State)).


cover_relation(S1,S2) :-
    rule(will_cause,if:[S1],then:[S2]).

cover_relation(S1,S2) :-
    rule(may_cause,if:[S1],then:[S2]).

anticipate_relation(S1,S2) :-
    rule(will_cause,if:[S1],then:[S2]).

prefer_state(S1) :-
    backward_inference(positive_evidence_state,S1).

prefer_connection(S1,S2) :-
    backward_inference(positive_evidence_connection,[S1,S2]).

rule_out_state(S1) :-
    backward_inference(negative_evidence_state,not(S1)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                          Hilfsklauseln                                  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


init([]) :-
    retractall(context(_)),
    retractall(considered_explanation(_,_)),
    retractall(accepted_explanation(_,_)),
    retractall(rejected_explanation(_,_)).

init([Complaint|Complaints]) :-
    init(Complaints),
    add_complaint(Complaint).
```

```
add_complaint(Complaint) :-
    assertt(context(Complaint),add_complaint),
    assertt(considered_explanation(Complaint,complaint),add_complaint),
    assertt(accepted_explanation(Complaint,complaint),add_complaint).


% Fragewissen: Attribute und ihre Formulierung

can_be_asked(alter,'Wie alt ist der Patient? ','(in Jahren)').

can_be_asked(fieber,'Hat der Patient Fieber? ',[ja,nein]).

can_be_asked(hautausschlag,'Liegt Hautausschlag vor? ',[ja,nein]).

can_be_asked(hautausschlag_knoten,'Besteht der Hautausschlag aus
               Knoten? ', [ja,nein]).

can_be_asked(hautausschlag_blasen,'Besteht der Hautausschlag aus
               Blasen? ', [ja,nein]).

can_be_asked(hautausschlag_flecken,'Besteht der Hautausschlag aus
               Flecken? ', [ja,nein]).

can_be_asked(hautausschlag_punkte,'Besteht der Hautausschlag aus
               Punkten? ', [ja,nein]).

can_be_asked(geschwollene_mandeln,'Sind die Mandeln geschwollen? ',
               [ja,nein]).

can_be_asked(geroeteter_rachen,'Ist der Rachen geroetet? ',[ja,nein]).

can_be_asked(schluckbeschwerden,'Liegen Schluckbeschwerden vor? ',
               [ja,nein]).

can_be_asked(augen_gelb,'Sind die Augen gelb verfaerbt? ',[ja,nein]).

can_be_asked(haut_gelb,'Ist die Haut gelb verfaerbt? ',[ja,nein]).

can_be_asked(alkoholiker,'Ist der Patient Alkoholiker? ',[ja,nein]).

can_be_asked(windpocken_bereits_gehabt,'Hat der Patient die Windpocken
               bereits gehabt? ',[ja,nein]).

can_be_asked(masern_bereits_gehabt,'Hat der Patient die Masern bereits
               gehabt? ',[ja,nein]).

can_be_asked(leberfunktion,'Test der Leber: Ergebnis? ',
               [gestoert,normal]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                          %%%
%%%                        Wissensbasis: Regeln                              %%%
%%%                                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%if: Ursache, then: Wirkung

rule(will_cause,if:[angina],then:[fieber=ja]).

rule(may_cause,if:[angina],then:[geschwollene_mandeln=ja]).

rule(may_cause,if:[angina],then:[geroeteter_rachen=ja]).

rule(may_cause,if:[angina],then:[schluckbeschwerden=ja]).


rule(will_cause,if:[windpocken],then:[fieber=ja]).

rule(will_cause,if:[windpocken],then:[hautausschlag=ja]).

rule(will_cause,if:[masern],then:[fieber=ja]).
```

```
rule(will_cause,if:[masern],then:[hautausschlag=ja]).
rule(will_cause,if:[scharlach],then:[angina]).
rule(will_cause,if:[scharlach],then:[hautausschlag=ja]).


rule(may_cause,if:[gelbsucht],then:[augen_gelb=ja]).
rule(may_cause,if:[gelbsucht],then:[haut_gelb=ja]).
rule(will_cause,if:[gelbsucht_von_neugeborenem],then:[gelbsucht]).
rule(will_cause,if:[gelbsucht_a],then:[gelbsucht]).
rule(will_cause,if:[gelbsucht_b],then:[gelbsucht]).


% if: Symptom, then: Diagnose


rule(positive_evidence_state,if:[hautausschlag_knoten=ja],
                             then:[windpocken]).
rule(positive_evidence_state,if:[hautausschlag_blasen=ja],
                             then:[windpocken]).
rule(negative_evidence_state,if:[hautausschlag_knoten=nein,
                             hautausschlag_blasen=nein],
                             then:[not(windpocken)]).
rule(positive_evidence_state,if:[hautausschlag_punkte=ja],then:[masern]).
rule(negative_evidence_state,if:[hautausschlag_punkte=nein],
                             then:[not(masern)]).
rule(positive_evidence_state,if:[hautausschlag_flecken=ja],
                             then:[scharlach]).
rule(negative_evidence_state,if:[hautausschlag_flecken=nein],
                             then:[not(scharlach)]).
rule(positive_evidence_state,if:[leberfunktion=gestoert],
                             then:[gelbsucht_b]).
rule(positive_evidence_connection,if:[ge(alter,18),alkoholiker=ja],
                             then:[[gelbsucht_b,gelbsucht]]).
rule(positive_evidence_connection,if:[alter=0],
                        then:[[gelbsucht_von_neugeborenem,gelbsucht]]).


rule(negative_evidence_state,if:[windpocken_bereits_gehabt=ja],
                             then:[not(windpocken)]).
rule(negative_evidence_state,if:[masern_bereits_gehabt=ja],
                             then:[not(masern)]).
```

ACHTUNG! Die Problemlösungsmethode ist hier zur Verdeutlichung in Prolog geschrieben. Es ist aber nur fast der vollständige Prolog-Code angegeben, der notwendig ist, damit das Programm tatsächlich eine Diagnose durchführt. Es fehlen aber lediglich Kleinigkeiten, die noch einmal Platz beansprucht und die Lesbarkeit beeinträchtigt hätten. Was deutlich werden sollte, ist der Zusammenhang zwischen den sehr abstrakten Begriffen einerseits, in denen über wissensbasierte Systeme gesprochen wird, und den Inferenzen und konkreten Regeln andererseits, die wir im Kapitel 3 gesehen haben. Auch zeigen die beiden Problemlö-

sungsmethoden, welche Gedanken zur Strukturierung eines Wissensbasis man sich machen muß, bevor man ein Expertensystem erstellt.

4.4 Wissenserwerb für Expertensysteme

Wir haben im vorigen Abschnitt gesehen, daß die Bemühungen um eine Verkürzung der Zeit, die Systementwickler für den Aufbau eines konkreten Expertensystems brauchen, zur Trennung von Wissensarten und expliziter Repräsentation der Problemlösung geführt haben. Um den Wissenserwerb für Expertensysteme zu verbessern, mußte man also zunächst die Expertensystem-Hüllen verbessern. Das Problem des Wissenserwerbs ist damit aber noch nicht gelöst! Es wird eine Entwicklungsumgebung gebraucht, die das Eintragen und Testen von Wissen unterstützt. Eine solche Entwicklungsumgebung muß die folgenden Tätigkeiten unterstützen:

- Inspektion vorhandener Daten und Regeln
- Erweitern der vorhandenen Daten und Regeln
- Prüfen und Verbessern vorhandener Daten und Regeln.

4.4.1 KADS

Da viele der neueren Arbeiten zum Wissenserwerb für Expertensysteme auf das Paradigma von KADS (knowledge acquisition and documentation system) aufbauen, soll es hier kurz vorgestellt werden. Die Grundidee von KADS ist die Strukturierung des Wissenserwerbs in zwei Phasen und vier Ebenen (Wielinga et al. 1992). Die zwei Phasen sind

- Entwicklung eines *conceptual model*, das das zu entwickelnde Expertensystem deklarativ beschreibt;
- Umsetzung des *conceptual model* in ein *design model*, das die Implementierung des Expertensystems beschreibt.

In der ersten Phase wird Wissen über die Anwendung erhoben. Es wird eine dem Sachbereich und der Aufgabenstellung nahe Terminologie verwendet und noch nicht an die Operationalisierung als System gedacht. Die Wissenserhebung muß nicht von InformatikerInnen vorgenommen werden, sondern kann auch von PsychologInnen oder Sachbereichsexperten durchgeführt werden. In der zweiten Phase geht es um die Umsetzung des Wissens in ein operationales System. Das *design model* verwendet Informatikterminologie. Es stellt die Dokumentation des Expertensystems dar, während das *conceptual model* die Spezifikation darstellt.

Das *conceptual model* ist die Zusammenfassung von detaillierteren Modellen. Es stellt die oberste von vier Ebenen dar. Diese Ebenen sind:

- **Organisationsmodell** (organizational model) und **Anwendungsmodell** (application model): hier wird die Einbettung eines Prozesses (z.B. Diagnose) in eine Organisation (z.B. Krankenhaus) und die Funktionalität des zu entwickelnden Systems erfaßt.

- **Aufgabenmodell** (task model): hier wird die Funktionalität des Gesamtsystems in einzelne Aufgaben unterteilt, die wiederum aus Teilaufgaben bestehen können.

- **Kooperationsmodell** (model of cooperation) und **Sachbereichsmodell** (model of expertise): Im Kooperationsmodell werden den Aufgaben bestimmte Akteure zugeordnet, die sie ausführen sollen - Akteure können Programme oder Menschen sein. Im Sachbereichsmodell wird spezifiziert, wie die Aufgaben zu lösen sind.

- Das ***conceptual model*** faßt Kooperationsmodell und Sachbereichsmodell zusammen.

Der KADS-Ansatz zum Wissenserwerb wird **modellbasiert** genannt. In den neueren Arbeiten werden insbesondere formale Repräsentationen für die Spezifikation (also das *conceptual model*) diskutiert, wobei entweder die Trennung vom *design model* aufgehoben wird, indem diese Repräsentationen operational sind, oder formale Beweisverfahren untersucht werden, mit denen die formale Spezifikation zur Prüfung der Implementation eingesetzt werden kann. Wir können unsere Darstellung der Problemlösungsmethoden heuristische Klassifikation und überdeckende Diagnose als ein *conceptual model* interpretieren. Da wir es in Prolog geschrieben haben, ist die Trennung zum *design model* aufgehoben. Unsere Programme sind direkte Umsetzungen der Spezifikation dieser Problemlösungsmethoden, die Guus Schreiber in der KADS Spezifikationssprache ML2 vorgenommen hat (Schreiber 1992).

4.4.2 Sloppy Modeling

Frühe Arbeiten betrachteten den Wissenserwerbsprozeß als Transfer von Wissen eines Experten in ein System. Diese Sicht setzt voraus, daß Experten über explizites und erklärbares Wissen verfügen, das sie zur Lösung ihrer Aufgaben heranziehen. Dies ist aber meist nicht der Fall. Gerade die Anfänger verwenden explizites Wissen zur Problemlösung - Experten haben Fertigkeiten entwickelt, die ihnen selbst unbewußt sind. Der Unterschied zwischen Fertigkeiten und Wissen bewirkt, daß das, was Experten als Erklärung für ihr Verhalten angeben können, nicht unbedingt die Grundlage ihrer Fähigkeit, Probleme zu lösen, wiedergibt. In der Psychologie wurde festgestellt, daß die Erklärung eigenen Verhaltens mit denselben Prozessen vorgenomen wird, wie die Erklärung des Verhaltens anderer Leute. Es gibt folglich keinen direkten Zugang zu den eigenen kognitiven Prozessen. Es ist daher nicht angemessen, Wissen über einen Sachbereich und Problemlösungsverhalten einfach dadurch zu erheben, daß man Experten befragt und ihre Antworten operational repräsentiert. Die Fragen können in dem Experten eventuell die Konstruktion von Erklärungen auslösen. Experten geben dann ad hoc Antworten, die einer naiven Theorie entsprechen. Naive Theorien sind Erklärungen, die noch nicht kritisch überprüft wurden. Natürlich ist es nicht angemessen, Expertensysteme zu bauen, deren Wissensbasis eine naive Theorie wiedergibt!

Wissenserwerb besteht also nicht darin, vorhandenes Wissen einfach zu formalisieren und zu operationalisieren. Vielmehr besteht die Aufgabe des Wissenserwerbs darin, einen Sachbereich zu modellieren. Ein Expertensystem zu bauen, heißt also, eine operationale Beschreibung für die Problemlösung in einem Sachbereich zu konstruieren. Die Aufgabe eines Wissenserwerbssystem besteht darin, diesen Prozeß zu unterstützen. Als Herausforderung für Wissenserwerbssysteme habe ich die "schlampige Modellierung" (sloppy modeling) eingeführt (Morik 1989):

- Man kann niemals erwarten, ein vollständiges, korrektes und angemessenes Modell für einen Sachbereich, eine Aufgabenstellung erreichen zu können. Vielmehr wird sich jedes auch noch so gründlich erarbeitete Modell nach einiger Zeit als unvollständig, inkorrekt oder unangemessen herausstellen, entweder weil neue Erkenntnisse erzielt wurden oder weil sich der Sachbereich geändert hat.

- Ein System, das den Wissenserwerbsprozeß unterstützt, sollte seinen Benutzern so viel Freiheit wie möglich gewähren. Es sollte die Modellierungstätigkeit unterstützen, die ein kreativer Prozeß ist. Das System sollte nicht auf Festlegungen bestehen, die ein Benutzer (noch) nicht treffen kann.

Der zweite Punkt grenzt *sloppy modeling* deutlich von dem Verfahren der schrittweisen Verfeinerung ab. Bei der schrittweisen Verfeinerung wird dem Benutzer ein bestimmtes Vorgehen aufgezwungen: jede Begriffsbestimmung oder Regel schränkt die Möglichkeiten weiterer Definitionen oder Regeln ein. Mit vorhergehenden Bestimmungen widersprüchliche Definitionen werden von Systemen, die schrittweise Verfeinerung anwenden, verboten.[24] Nun ist es sicherlich eine Hilfe, wenn ein System Widersprüche feststellen kann. Es sollte jedoch dem Benutzer überlassen, wann und in welcher Weise er diese Widersprüche auflösen möchte. Der zweite Punkt grenzt *sloppy modeling* ebenfalls von dem isolierten maschinellen Lernen ab. Dort werden zunächst nur Beispiele erstellt, aus denen dann Regeln gelernt werden. Auch dies zwingt dem Benutzer ein bestimmtes Vorgehen auf. So kann der Benutzer nicht diejenigen Regeln, die er bereits entwickelt hat, in das System eingeben, sondern muß sich auf Beispiele beschränken. Ein Wissenserwerbssystem im Paradigma des *sloppy modeling* ermöglicht es seinen Benutzern, sowohl vom Allgemeinen zum Speziellen (schrittweise Verfeinerung) wie auch vom Speziellen zum Allgemeinen (Lernen aus Beispielen) vorzugehen.

Zur Modellierung gehören die folgenden Schritte, die von einem Wissenserwerbssystem unterstützt werden müssen:

- Der Rahmen des Modells wird erarbeitet, die Signatur festgelegt, die grobe Struktur des Modells umrissen.

- Der Rahmen wird ausgearbeitet, Begriffe definiert, Regeln aufgestellt, Fakten erhoben.

- Das Modell wird überprüft, fehlende Bestimmungen und Widersprüche entdeckt und es wird ausprobiert, ob sich damit Probleme der Anwendung auch lösen lassen.

Wichtig beim *sloppy modeling* ist es nun, daß alle diese Schritte vom System unterstützt werden und zwar so, daß alle Entscheidungen revidierbar sind. Es wird nicht davon ausgegangen, daß ein Schritt vollständig bearbeitet wird, bevor der nächste angegangen wird! Benutzer können frei zwischen diesen Schritten hin- und herspringen. Alle Festlegungen, die einmal getroffen wurden, können vom System unterstützt wieder rückgängig gemacht werden. So kann z.B. anhand des Versuchs, mit dem bisher operationalisierten Modell ein Problem zu lösen, festge-

[24] Termsubsumtionssysteme werden z.B. meist so aufgebaut, daß zunächst die Oberbegriffe und dann die Unterbegriffe in die Tbox eingetragen werden.

stellt werden, daß die Signatur unangemessen ist. In dem Falle muß das Wissenserwerbssystem die Veränderung der Signatur unterstützen. Ein Beispiel mag dies verdeutlichen. Nehmen wir an, wir hätten vor, die verschiedenen Diagnosen bei der Gelbfärbung der Haut zu modellieren. Vielleicht hätten wir mit dem Fall des Patienten Tim und der einfachen Regel angefangen

```
color (yellow) --> disturbed (liver)
```

Diese Repräsentation wird sich schon bald als ungeschickt herausstellen. Zum ersten kann diese Regel nicht verallgemeinert werden, da die Regel

```
color(X) --> disturbed (Z)
```

alle möglichen Organe bei allen möglichen Farben gestört sein läßt. Die Variable Z ist nicht durch Bedingungen eingeschränkt. Wir könnten nun ein Prädikat relation einführen, das eine Beziehung zwischen einer Farbe und einem Organ herstellt. Ein Faktum wäre dann

```
relation(yellow, liver)
```

und die Regel wäre

```
relation (X, Y) & color (X) --> disturbed (Y).
```

Zum zweiten macht die Regel nicht deutlich, wessen Hautfarbe gelb ist - es gilt für alle Patienten. Also werden wir die Signatur ändern wollen und statt des einstelligen Prädikats color das zweistellige color2 einführen. Ein System, das die Repräsentationsänderung unterstützt, kann nun per Vorwärtsinferenz mit der Transformationsregel

```
color(X) & patient (Y) --> color2 (X,Y)
```

für alle bereits eingetragenen Fakten für color und das Faktum

```
patient (tim)
```

die entsprechenden color2 -Fakten erzeugen. Entsprechend muß aber auch die Regel geändert werden:

```
color2 (X,Y) & relation (X, Z) --> disturbed (Z,Y)
```

Nun ist allerdings Tim, wenn denn einmal eine Gelbfärbung bei ihm aufgetreten ist, für alle Zeiten gelb. Vielleicht wollen wir also lieber das vierstellige Prädikat color4 einführen, das Person, Farbe und Anfangs- und Endzeitpunkt der Gelbfärbung angibt. Entsprechend wird dann auch das Prädikat disturbed vierstellig. Es zeigt sich nun aber, daß außer blau und gelb andere Farben keine medizinische Bedeutung haben. Insofern wollen wir vielleicht statt der allgemeinen Prädikate colour4 und relation das dreistellige Prädikat yellow einführen. Die entsprechenden Transformationsregeln wären dann:

```
color4(yellow, X, T1, T2) --> yellow(X, T1, T2) und

disturbed (liver, X, T1, T2) --> liver_disturbed(X, T1, T2)
```

Dies überführt alle bereits gemachten Eintragungen z.B. über Tim in die neue Repräsentation. Wir müssen aber auch noch die Regel ersetzen durch

```
yellow(X, T1, T2) --> liver_disturbed(X, T1, T2).
```

Um solche Repräsentationsänderungen zu unterstützen, muß ein System mindestens alle betroffenen Fakten und Regeln heraussuchen und präsentieren können und Regeln für die Vorwärtsinferenz zur Verfügung stellen. Besser als die normale Vorwärtsinferenz ist eine spezielle Transformationsregel, deren Konklusion auch dann noch gilt, wenn alle Prämissen gelöscht sind.

Die Erkenntnis, daß Wissenserwerb ein Prozeß des Modellierens ist und daß Modellierung ein zyklischer, infiniter Prozeß ist, hat sich inzwischen allgemein durchgesetzt. Eine Realisierung eines Wissenserwerbssystems ist das an der GMD entwickelte MOBAL (Morik et al. 1993). Es beinhaltet zudem maschinelle Lernverfahren, auf die wir in Kapitel 5 zu sprechen kommen werden.

4.5 Literatur

Brachman, R.J. (1977): What's in a Concept: Structural Foundations for Semantic Networks, in: International Journal of Man-Machine Studies, 9, 127-152

Brachman, R.J. (1979): On the Epistomological Status of Semantic Networks, in: Findler, N.V. (ed.): Associative Networks: Representation and Use of Knowledge by Computers, New York: Academic Press

Clancey, W.J. (1983): The Epistomology of a Rule-Based Expert System - a Framework for Explanation, in: Artificial Intelligence, 20, 215-251

Clancey, W.J. (1985): Heuristic Classification, in: Artificial Intelligence, 27, 289-350

*Clancey, W.J. (1986): From Guidon to Neomycin and Heracles in Twenty Short Lessons, in: AI Magazine, Vol. 7, No.3, 40-60

Eshelman, L., Ehret, D., McDermott, J., Tan, M. (1987): MOLE: A Tenacious Knowledge Acquisition Tool, in: *Int. Journal of Man-Machine Studies*, Vol. 26, No. 1, 41-54

Morik, K. (1989): Sloppy Modeling, in: Morik (ed): *Knowledge Representation and Organization in Machine Learning*, Springer, 107 - 134

Morik, K. (1991): Underlying Assumptions of Knowledge Acquisition and Machine Learning, in: *Knowledge Acquisition and Machine Learning*, Vol. 3, 137-156.

*Morik, K., Wrobel, S., Kietz, J-U., Emde, W. (1993): *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications*, Academic Press

Newell, Allen (1982): The Knowledge Level, in: *Artificial Intelligence*, Vol. 18, 87-127

*Puppe, F., Gappa, U., Poeck, K., Bamberger, S. (1996): Wissensbasierte Diagnose- und Informationssysteme, Berlin: Springer

Shortliffe, E. (1976): Computer-Based Medical Consultations: MYCIN, American Elsevier

Schreiber, G. (1992): Pragmatics of the Knowledge Level, Doktorarbeit an der Univ. Amsterdam

Wielinga,B.J., Schreiber, A.T., Breuker, J.A. (1992): KADS - A Modeling Approach to Knowledge Engineering, in: Knowledge Acquisition Journal, Vol. 4, No. 1, special issue on KADS

5 Maschinelles Lernen

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 Turing (1950) war die Lernfähigkeit eines Rechners die wichtigste Intelligenzleistung. Er empfahl, einen Rechner "zu erziehen", so daß er seine Leistungen verbessert, da man unmöglich alles einprogrammieren könne. Insofern war maschinelles Lernen und Programmsynthese damals noch nicht unterschieden.

5.1 Was ist Lernen?

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:

> *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 mithilfe 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 (1986) gibt auch ein drastisches Beispiel dafür an, daß Leistungssenkung ein Lernergebnis sein kann. Wenn Zwangsarbeiter eines Konzentrationslagers 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 (1983) argumentiert gegen die Leistungsmessung bei der Definition vom Lernen. Er führt als Beispiel einen Spaziergänger in einer ihm noch unbekannten Stadt an, der an der öffent-

lichen Bücherhalle 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 Bücherhalle gibt. Erst wenn ein Passant ihn nach dem Weg zur Bücherhalle 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:

> *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 (1986):

> *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.

5.2 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 mithilfe 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.

5.2.1 Menschliches und maschinelles Lernen

Die kognitive Orientierung verwendet psychologische Arbeiten zum Begriffserwerb. 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.

Der Begriffserwerb kann in zwei Phänomenbereiche unterteilt werden: die Kategorisierung oder Aggregation und die Charakterisierung oder Definition. Die **Kategorisierung** gruppiert Objekte, Ereignisse und Sachverhalte der Welt in Klassen - eben: Kategorien. Damit ist die Extension eines Begriffs gegeben. 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 eigentlich drei Phänomenbereiche:

Kategorisierung ---> Charakterisierung ---> Klassifikation (Erkennung)

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

5.2.1.1 Gründe der Kategorienbildung

Zunächst könnte man als einen guten Grund dafür, Objekte der Welt zu einer Kategorie zusammenzufassen, angeben, daß sie ein Merkmal gemeinsam haben. 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,[25] ansonsten wird die Kategorie konventionalisiert. Quine (1977) sah in der Notwendigkeit, etwas vorhersagen zu können, das Motiv für individuelle und gesellschaftliche Kategorienbildung. 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 Handlungs-zusammenhang sinnvoll werden kann, führen Murphy und Medin (1985, S. 294) gestreifte Objekte mit mehr als einem Bein an, die zwischen 11 und 240 kg wiegen. Im Kontext eines Spielfilms, in dem diese Objekte Außerirdische sind, die die Menschheit bedrohen, wird die Kategorie sinnvoll. Es ist dann wichtig zu erkennen, wer dieser Kategorie angehört, wie er

[25] Zu ad hoc Kategorien s. Barsalou (1983)

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, Medin 1985, S.298). 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 Carey (1985) gibt obendrein Hinweise auf die Aggregation von Objekten. 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 sowie 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.

5.2.1.2 Probleme ähnlichkeitsbasierter Charakterisierung

Der ähnlichkeitsbasierte 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 ähnlichkeitsbasierten 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 (1967) wies die Abhängigkeit der Farbwahrnehmung von der durch

Wörter einer natürlichen Sprache gegebenen Einteilung des Farbspektrums nach. 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 ähnlichkeitsbasierten 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 (1985) beobachtete außerdem, daß Kinder, wenn sie einmal bestimmte Merkmale dafür benutzten, einen Begriff zu charakterisieren, gegensätzliche Begriffe mit anderen Werten derselben Merkmale definierten. 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.

Carey (1985) betont die Abhängigkeit der Begriffsstruktur von dem Wissen eines Menschen. Die Definition des Belebten hängt ab von dem Wissensstand über Biologie. Auch Keil und Kelly (1987) zeigen, daß Versuchspersonen mit wenig Wissen über einen Sachbereich eher beschreibende Merkmale auswählen, während zur Verwendung definitorischer Merkmale mehr Wissen nötig ist. Die Verschiebung von Beschreibungen zu Definitionen ist damit nur indirekt einer Altersstufe 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 Beispiele dafür 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 ähnlichkeitsbasierten 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, 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 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.

5.2.1.3 Beiträge aus dem maschinellen Lernen

Das maschinelle Lernen ist zunächst dem ähnlichkeitsbasierten 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.

Die Notwendigkeit für komplexere Repräsentationen und das Einbeziehen von Hintergrundwissen wurde von Michalski schon 1983 und von Kodratoff und Ganascia 1986 dargestellt. Reichere Formalismen sind etwa Termsubsumtionssysteme (Lernen einer Tbox aus einer Abox -- Kietz, Morik 1994, Lernbarkeit von Begriffsdefinitionen in CLASSIC -- Pitt 1996) oder Hornklauseln. Die induktive lo-

gische Programmierung behandelt das Lernen von eingeschränkten prädikatenlogischen Formeln unter Verwendung von Hintergrundwissen.

Der Bedarf für einen Begriff wird von Michalski und Stepp (1986) durch einen Zielgraphen expliziert, der ihr *conceptual clustering* Verfahren CLUSTER/S steuert. Wrobel (1994) entwickelt den Ansatz von Emde, Habel und Rollinger (1983) weiter. Durch Ausnahmen einer ansonsten erfolgreichen Regel ist ein Bedarf für einen neuen Begriff gegeben, wenn sich anders die Ausnahmen nicht von den erfogreichen Regelanwendungen unterscheiden lassen. Die Ausnahmen sollen durch einen neuen Begriff zusammengefaßt werden, auf den eine zusätzliche Prämisse der Regel verweist. Das System KLUSTER (Kietz, Morik 1994) beschreibt den Bedarf für einen neuen Begriff aufgrund der Unfähigkeit, mit den vorhandenen Begriffen eine Kategorie zu charakterisieren.

Aktuelle Arbeiten versuchen, die Verankerung des Begriffserwerbs in der Welt zu modellieren. So schlägt Wrobel (1991) einen kognitiv motivierten Forschungsrahmen vor, in dem strikt inkrementell gelernt wird. 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. Saitta und Giordana(1990) schlagen eine Begriffsstruktur vor, in der Merkmale Handlungen zur Verifizierung des Merkmals darstellen. Die Verankerung von Bewegungsbegriffen 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 (Klingspor, Morik, Rieger 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, Morik, wird erscheinen). Es gibt bereits ein Buch über LHM (Spada, Reimann 1996), eine Serie von weiteren Büchern wird 1998 erscheinen.

5.2.1.4 Literaturhinweise zur Psychologie

Barsalou, L.W. (1983): Ad hoc Categories, in: *Memory and Cognition*, 11, 1983

Carey, S. (1985): *Conceptual Change in Childhood*, MIT Press

Keil, F.C., Kelly, M.H. (1987): Developmental Changes in Category Structure, in: Harnad (ed): *Categorical Perception - The Groundwork of Cognition*, Cambridge University Press

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

Lenneberg, E.H. (1967): *Biological Foundations of Language*, New York

Kietz, J.-U., Morik, K. (1991): Constructive Induction : Learning Concepts for Learning, Arbeitspapiere der GMD, Nr. 543, Gesellschaft für Mathematik und Datenverarbeitung, Schloß Birlinghoven, 1991

Mühlenbrock, M., Morik, K. (wird erscheinen): Rekonstruktion des Erwerbs einer Begriffsstruktur zur Erklärung des Tag/Nacht-Zyklus'

Murphy, G. L,. Medin, D.L.(1985): The Role of Theories in Conceptual Coherence, in: *Psychological Review*, Vol 92, Nr.3

Piaget, J. (1977): *The Development of Thought*, New York: Viking Pinguin

Rosch, E. (1978): Principles of Categorization, in: Rosch, Lloyd (eds): *Cognition and Categorzation*, Hillsdale, N.J.: Erlbaum

Scholnick, E.K. (ed)(1983): *New Trends in Conceptual Representation - Challenges to Piaget's Theory?*, Hillsdale, N.J.: Erlbaum

Spada, H., Reimann, P. (Hg.) (1996): *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*, Oxford: Elsevier

5.2.2 Induktion und Abduktion

Der deduktive Schluß ist am längsten und gründlichsten untersucht worden. Im maschinellen Lernen steht der induktive Schluß im Vordergrund. Neuerdings wird auch der abduktive Schluß einbezogen. Wir können die drei Schlüsse folgendermaßen darstellen:

$(A \rightarrow B), A \models B$ ist ein **deduktiver Schluß**.

Beispiel: $\forall x \mid mensch(x) \rightarrow sterblich(x), mensch(uta) \models sterblich(uta)$

Die deduktiven Inferenzregeln sind:

$\frac{A \rightarrow B \quad A}{B}$ (Modus Ponens) und $\frac{\forall X \mid a(X) \rightarrow b(X)}{a(c) \rightarrow b(c)}$ (Instantiierung)

$a(c_1), b(c_1), ..., a(c_n), b(c_n) \mid\!< \forall x \mid a(x) \rightarrow b(x)$ ist ein **induktiver Schluß**.

Beispiel:

$$mensch(uta), sterblich(uta), mensch(udo), sterblich(udo), mensch(uwe), sterblich(uwe) \mid\!< \forall X \mid mensch(X) \rightarrow sterblich(X)$$

Wir nennen die Grundbeispiele Daten, D, und die allquantifizierte Aussage Hypothese, H. Wenn wir noch eine Theorie, T, als Hintergrundwissen hinzunehmen, so ist der induktive Schluß:

$T, D \mid\!< H$, wobei $T, H \models D$, $T \not\models D$ und $T, D \not\models \neg H$

Das heißt, die Beispiele folgen erst aus der Theorie, wenn der allgemeine Satz H (die Hypothese) hinzugenommen wird, vorher nicht. Außerdem ist die Hypothe-

se konsistent mit der Theorie und den Beispielen, d.h. aus Theorie und Beispielen folgt nicht die Negation der Hypothese.

Beispiel: T: $\forall X \mid$ mensch(X) --> säugetier(X) ,
mensch(uta), mensch(udo), mensch(uwe),
D: sterblich(uta), sterblich(udo), sterblich(uwe) |<
H: $\forall X \mid$ säugetier(X) --> sterblich(X)

Daraus, daß Menschen Säugetiere sind, folgt, daß auch Uta, Udo und Uwe Säugetiere sind. Nimmt man hinzu, daß alle Säugetiere sterblich sind, so kann man folgern, daß auch Uta, Udo und Uwe sterblich sind. Ohne eine induzierte Hypothese folgt es nicht. Daß alle Säugetiere nicht sterblich sind, läßt sich aus der Theorie und den Beispielen nicht folgern.

Der **abduktive Schluß** schließlich wird sehr unterschiedlich aufgefaßt. Im einfachsten Falle dreht er den Modus Ponens um.

(A -> B), B |> A

Beispiel: $\forall X \mid$ mensch(X) -> sterblich(X), sterblich (uta) |> mensch(uta)

Nehmen wir eine Theorie als Hintergrundwissen hinzu, so ist der abduktive Schluß:

T, (A -> B), B |> A, wobei T, (A -> B), B $|\neq \neg$A und A $\neq$ B

Die folgenden Abschnitte handeln überwiegend vom induktiven Schluß. Nur das erklärungsbasierte Lernen, das auch dargestellt wird, verwendet ihn nicht zum Lernen.

5.2.3 Anwendungen maschinellen Lernens

Maschinelles Lernen wird überwiegend eingesetzt, um eine Menge von Regeln oder einen Entscheidungsbaum aus Daten zu gewinnen oder eine gegebene Regelmenge zu verbessern. Die Regeln werden dann entweder direkt von Menschen verwendet oder einem Expertensystem und damit dessen Benutzern zur Verfügung gestellt. Schon der Einsatz einfacher Lernverfahren führt zu einer erheblichen Verkürzung der Entwicklungszeit einer Wissensbasis. Meist wird anhand einer ausgewählten Teilmenge von klassifizierten Daten (**Lernset**) eine Menge von Regeln oder ein Entscheidungsbaum induziert. Das Lernergebnis wird dann anhand einer anderen Teilmenge der klassifizierten Daten (**Testset**) geprüft. Dabei wird der Testset ohne die vorgegebene Klassifikation mit dem Lernergebnis klassifiziert. Wenn die Klassifikation durch das Lernergebnis mit der benutzergegebenen Klassifikation übereinstimmt, ist es **korrekt**. Wenn nicht, wird noch einmal mit einem anderen Lernset gelernt oder per Hand das Lernergebnis verbessert.

Donald Michie (1989) berichtet über erfolgreiche Anwendungen von Lernverfahren, die Entscheidungsbäume induzieren. 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 vorliegenden 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. Mithilfe 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.[26] Dies ist deshalb so wertvoll, weil die Firmen nur selten über ihre Anwendungen maschinellen Lernens berichten, so daß öffentlich zugängliches Material fehlt.

Oft ist eine Induktionskomponente in eine Wissenserwerbsumgebung für eine Expertensystem-Hülle integriert wie zum Beispiel beim System IKEE für die Expertensystem-Hülle TWAICE.[27] Exemplarische Anwendungen verschiedener Lernverfahren wurden in dem ESPRIT-Projekt "Machine Learning Toolbox" (P2154) in Zusammenarbeit von Industrieunternehmen, Universitäten und Forschungsinstitutionen untersucht. So wurde zum Beispiel das System MOBAL[28] für unterschiedliche Anwendungen erprobt. Ein medizinischer Sachbereich wurde einerseits mithilfe von benutzergegebenen Regeln dargestellt. Andererseits lernte das System typische Therapieabläufe aus im Krankenhaus gesammelten und von einem Arzt klassifizierten (und bereinigten) Daten. Mithilfe der Konsistenzprüfung von MOBAL (Wrobel 1994) wurden Abweichungen festgestellt und dann analysiert (Morik et al. 1994). Eine andere Anwendung des Systems MOBAL entspricht genauer dem klassischen Anwendungsbereich maschinellen Lernens: eine Wissensbasis zur Zugangsberechtigung von Benutzern zu bestimmten Rechnerleistungen soll mithilfe des Systems erstellt werden. Dabei unterstützt das System verschiedene Aufgaben der Modellierung. Lernverfahren können Regeln aus Daten gewinnen und anhand von Ausnahmen verfeinern. Der Benutzer kann ebenfalls Regeln eingeben. Diese Regeln werden beim Regellernen und Regelverfeinern berücksichtigt (Sommer et al. 1994). Neben den Lernverfahren verfügt MOBAL aber noch über andere Komponenten, die den Benutzer bei der Modellierung unterstützen. In der Anwendung ist es meist mit einem isolierten Lernverfahren nicht getan!

---

[26] Dies entspricht einer Umsatzsteigerung von 10.000 US$ im Jahr.

[27] TWAICE und IKEE sind Entwicklungen der Nixdorf Computer AG, die nunmehr Teil der SNI ist.

[28] MOBAL wurde an der Gesellschaft für Mathematik und Datenverarbeitung in Birlinghoven entwickelt (Morik et al. 1993).

Wenn in dem Aufbau und der Verfeinerung von Wissensbasen mithilfe 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, 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 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 Bewgriffen der Planungsebene und den Sensor- und Handlungsdaten zu verbessern (Klingspor, Morik, Rieger 1996).

Wissensentdeckung in Datenbanken (data mining): Es 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 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 1996). Ein prädikatenlogisches Verfahren zum Regellernen mit direktem Datenbankzugriff ist RDT/DB (Brockhausen, Morik 1997).

5.3 Lernaufgaben

Hier werden drei Lernaufgaben vorgestellt: das Lernen von Funktionsapproximationen, von Begriffsdefinitionen und von allen gültigen Regeln. Dabei wird unterschieden zwischen überwachtem Lernen (Lernen aus Beispielen) und unüberwachtem Lernen (Lernen aus Beobachtungen). **Beispiele** sind einer Kategorie zugeordnete (klassifizierte) Aussagen. Im Gegensatz dazu sind **Beobachtungen** nicht klassifiziert. Beim Lernen aus Beispielen wurde also die Kategorisierung bereits vom Benutzer oder einem anderen System vorgenommen. Beim Lernen aus Beobachtungen gehört die Aggregation zur Lernaufgabe.

Ein weiterer Unterschied ist, ob die Beispiele oder Beobachtungen auf einmal oder nach und nach dem Verfahren gegeben werden. **Inkrementell** ist ein Verfahren, das nicht alle Eingabedaten auf einmal bekommt und dann lernt, sondern jeweils ein zusätzliches Beispiel oder eine neue Beobachtung einliest, daraus lernt, dann das nächste einliest, und so weiter. So ist der Versionenraum (siehe 5.4) in-

krementell, ID3 hingegen nicht inkrementell (siehe 5.7.1). Die Schwierigkeit inkrementellen Lernens besteht darin, daß Entscheidungen bereits getroffen werden, bevor der Rest der Beispiele oder Beobachtungen zur Verfügung steht.

Entscheidend für die Schwierigkeit einer Lernaufgabe sind nicht nur die Bedingungen an die Lösung, sondern auch der Repräsentationsformalismus, in dem die Hypothesen (mögliche Lernergebnisse) ausgedrückt werden.

5.3.1 Begriffslernen

Das Begriffslernen ist die klassische Aufgabe des maschinellen Lernens. Begriffe wie z.B. "kreditwürdige Personen", "Situation für den Landeanflug mit Autopiloten", "Streptokokken-Infekt" können unmittelbar in Planungssystemen, Entscheidungsunterstützungssystemen oder Diagnosesystemen zur Klassifikation angewandt werden. Die Beschreibungen der Beobachtungen enthalten Merkmale (und Relationen) von Objekten. Die Beschreibungen der Beispiele enthalten zusätzlich die Angabe, ob das Beispiel eine Instanz des zu lernenden Begriffs ist (positives Beispiel) oder nicht (negatives Beispiel). Als Bedingung an die Lösung der Lernaufgabe werden logische Verhältnisse zwischen den Instanzen und der Hypothese formuliert. Da unterschiedliche Hypothesen für eine gegebene Menge von Beobachtungen/Beispielen die Bedingungen Konsistenz und Vorhersage oder Vollständigkeit und Korrektheit erfüllen können, wird manchmal ein zusätzliches Präferenzkriterium angegeben. Dies kann z.B. sein, daß immer die allgemeinste Begriffsdefinition gewählt werden soll oder gerade die speziellste. Einige Verfahren sind in der Lage, Hintergrundwissen T zu berücksichtigen. Wir haben dann die Repräsentationssprachen: eine Beschreibungssprache L_E für Beispiele/Beobachtungen, eine Hypothesensprache L_H und eine Sprache für das Hintergrundwissen L_T. Natürlich hängen diese drei zusammen.

Beispielsweise kann
L_E variablenfreie Hornklauseln,
L_T Grundfakten und
L_H Hornklauseln sein,
wobei die Signaturen sich überschneiden, d.h. die Menge der Prädikate in L_E, L_T und L_H sind nicht disjunkt.

Begriffslernen aus Beispielen:	**Begriffslernen aus Beobachtungen:**
Gegeben:	Gegeben:
Hypothesensprache L_H für den Begriff,	Hypothesensprache L_H für den Begriff,
Hintergrundwissen T in einer Sprache L_T (ggf. leer)	Hintergrundwissen T in einer Sprache L_T (ggf. leer)
Menge P positiver Beispiele in einer Beschreibungssprache L_E	Menge E von Beobachtungen in einer Sprache L_E

[29] In der Lernliteratur wird entweder nicht (wie bei Termsubsumtions-Formalismen) zwischen Klassifikation von Begriffen und Realisierung von Begriffen unterschieden oder es wird unter Klassifikation gerade die Realisierung in Termsubsumptionssystemen verstanden.

Menge N negativer Beispiele in L_E

Relation, die Beispiele e anhand von Hypothese c klassifiziert (z.B. das Abgleichsprädikat `covers(c,e)`[29] oder die deduktive Ableitung T,c|-- e)

ggf. Präferenzkriterium, das Hypothesen (partiell) ordnet

$\exists$ p $\in$ P:T,P-{p}|-/- p(Notwendigkeit)

Ziel:

c $\in$ L_H mit

T, c |-/- $\perp$ (Konsistenz)

$\forall$ p $\in$ P: `covers` (c,p) ist wahr bzw. T, c |-- P (Vollständigkeit)

$\forall$ n $\in$ N: covers (c,n) ist falsch bzw. $\forall$ n $\in$ N: T, c|-/- n (Korrektheit)

(c erfüllt das Präferenzkriterium)

Relation, die Begriffsdefinitionen c und Beobachtungen e verbindet (z.B. das Abgleichsprädikat `covers(c,e)` oder die deduktive Ableitung T,c|--e)

ggf. Präferenzkriterium, das Hypothesen (partiell ordnet)

$\exists$ e $\in$ E: T,E-{e}|-/- e (Notwendigkeit)

Ziel:

c $\in$ L_H mit

T, c |-/- $\perp$ (Konsistenz)

$\exists$ e $\in$ L_E, e $\notin$ E und T, c |-- e (Vorhersage)

(c erfüllt das Präferenzkriterium)

5.3.2 Regellernen

Das Lernen aller gültigen und redundanzfreien Regeln (kurz: Regellernen) ist die schwierigste Lernaufgabe. Sie findet in einer Menge von Daten (Beobachtungen und Hintergrundwissen) alle Regeln, die bei diesen Daten gelten. Es geht nicht darum, eine Vorhersage für die Realität, der die Daten entstammen, anzunähern, sondern verständliche Aussagen über die Daten zu machen - wie auch immer sich Daten und Regeln auf die Realität beziehen. Insofern ist der Grundgedanke bei der Funktionsapproximation und dem Regellernen völlig unterschiedlich: Während beim ersten das Lernergebnis eine gegebene Wahrheit annähern soll, beläßt das Regellernen die Suche nach Wahrheit beim Benutzer des Systems. Ein Regellernverfahren faßt die Daten zusammen und überläßt es dem Benutzer festzustellen, wie der Bezug zur Realität einzuschätzen ist. Daher ist eine häufige Anwendung des Regellernens die Datenkorrektur. Wenn Regeln, die im minimalen logischen Modell der Daten gültig, notwendig und vollständig sind, vom Benutzer für unwahr befunden werden, so kann dies ein Hinweis auf fehlerhafte oder unvollständige Daten sein. Beispielsweise haben wir bei Mercedes-Daten über Fahrzeuge gelernt, daß ein Fahrzeug 0 bis 8 Achsen hat. Da wir wissen, daß dies nicht stimmt, konnten wir gezielt die falschen Einträge in der Datenbank bereinigen, die für ein Fahrzeug 0 Achsen angaben.

Gegeben:

Sprache für Regeln L_H

Beobachtungen E in einer Sprache L_E

ggf. Theorie in einer Sprache L_T

T, E |-/- ⊥ (Konsistenz)

Ziel:

C ∈ L_H so daß M (C) ⊇ M+(T, E) (Gültigkeit)

∀ c ∈ C, ∃ e ∈ E: T, E-{e} |-/- e und T, E-{e}, c |-- e (Notwendigkeit)

Wenn c gültig und notwendig ist, dann C |-- c (Vollständigkeit)

Es gibt keine echte Teilmenge G von C, die gültig und vollständig ist (Minimalität)

Bei dieser Lernaufgabe werden die Begriffe Modell M und minimales Modell M+ verwendet. Gegeben eine Interpretation I für eine Menge von Formeln F. I ist ein **Modell** von F, geschrieben M(F), wenn alle Formeln von F in I wahr sind. Wenn I ein Modell von F ist und es keine Interpretation I' gibt, so daß I ⊇ I' und I' ist ein Modell von F, dann heißt I **minimales Modell** von F, geschrieben M+(F).

Normalerweise gibt es viele verschiedene und beliebig große Modelle für prädikatenlogische Formeln. Einige Beschränkungen der Prädikatenlogik führen aber dazu, daß es nur ein minimales Modell für eine Menge von Formeln gibt. Zum Beispiel haben definite Hornklauseln immer genau ein minimales Modell, wenn es eines gibt. Eine **Hornklausel** heißt **definit**, wenn sie entweder aus einem positiven und beliebig vielen negativen Literalen besteht oder nur aus einem positiven Literal. In Prolog sind also Fakten und Regeln definite Hornklauseln, nicht jedoch Anfragen. Wegen der harten Anforderung der Gültigkeit des Regellernens werden solche Beschränkungen bevorzugt.

5.4 Lernen als Suche

Mitchell (1982) hat Lernen aus Beispielen als Suche beschrieben. Der Suchraum für Begriffe ist die Menge aller mithilfe von L_H bildbaren Ausdrücke. Das sind alle möglichen Charakterisierungen, für die dann festgestellt werden muß, ob sie alle positiven Beispiele abdecken und kein negatives. Der einfachste Lernalgorithmus ist demnach der Aufzählungsalgorithmus: er zählt alle in L_H 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 Beispiele und Nicht-Beispiele gleichermaßen abdeckende Aussage. 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 Raum der Zielbegriff liegen könnte.

Mitchell schlug für Definitionssprachen eine Halbordnung (quasi-ordering) aufgrund der spezieller-als- bzw. genereller-als-Relation vor. Eine Halbordnung ist eine reflexive, transitive aber nicht antisymmetrische Relation. Wenn Begriffe mithilfe von Attributwerten charakterisiert werden, die sich in einer Hierarchie entlang dieser spezieller-als- bzw. genereller-als-Relation partiell anordnen lassen, läßt sich der Suchraum als Kreuzprodukt der geordneten Attributwerte immer (halb-)ordnen.

spezieller-als:

c1 ist spezieller als c2 genau dann, wenn

$\forall$ e $\in$ L_E gilt: covers (c1, e) --> covers (c2, e), d.h.:

{ e $\in$ L_E | covers (c2, e) } $\supseteq$ { e $\in$ L_E | covers (c1, e) }

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.

Schrittweises Generalisieren:

g, g' $\in$ L_H, e $\in$ L_E, g ist minimal genereller als c bezügl. e genau dann, wenn

g ist genereller als c und

covers (g, e) und

es gibt kein g' $\in$ L_H, so daß g genereller als g' ist und

covers (g', e) gilt.

Es wird also eine speziellste Generalisierung g 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 formalisiert Mitchell auch die Spezialisierung.

Schrittweises Spezialisieren:

s, s' $\in$ L_H, e $\in$ L_E, s ist minimal spezieller als c bezügl. e genau dann, wenn

s ist spezieller als c und

$\neg$ covers (s, e) und

es gibt kein s' $\in$ L_H, so daß s spezieller ist als s' ist und

$\neg$ covers (s', e) gilt.

Es wird also eine generellste Spezialisierung s 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.

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.

Mitchell (1982) führte zusätzlich zu diesen beiden Algorithmen die bidirektionale Suche im **Versionenraum** (versions space) ein, die Spezialisierung und Generalisierung kombiniert. 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. Sobald sich die beiden Mengen überschneiden, ist die Lösung gefunden: es ist die Hypothese aus der Schnittmenge.

Versionen-Raum Lernverfahren:
Initialisiere die Menge G mit den generellsten Begriffen
und die Menge S mit den speziellsten Begriffen.
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 alle 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 alle g $\in$ G, die e nicht abdecken,
        generalisiere S bis e abgedeckt wird.
        entferne alle s $\in$ S, die echt genereller sind als ein anderes s' $\in$ S
    entferne alle g $\in$ G, für die es kein s $\in$ S gibt, das spezieller ist
    entferne alle s $\in$ S, für die es kein g $\in$ G gibt, das genereller ist
Sobald G und S gleich sind und nur noch eine Hypothese enthalten, dann gib
    diese aus und halte an!

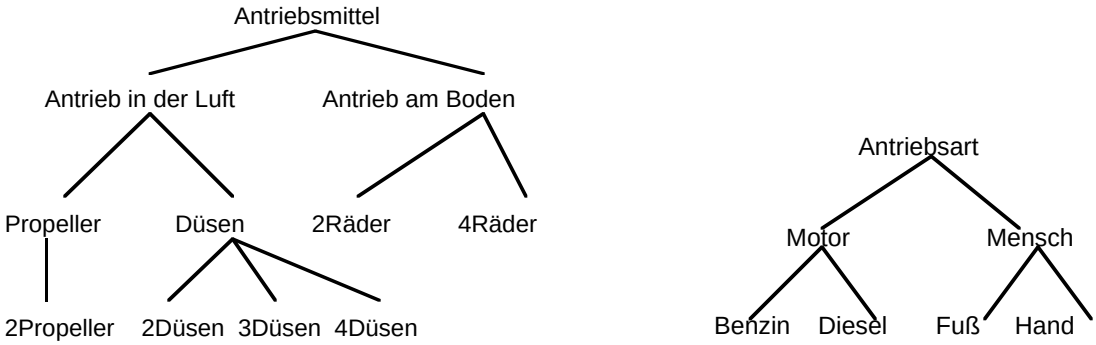
Die Mengen G und S sind also folgendermaßen definiert:

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

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

Dabei sind p_i und n_i die bisher dem System gezeigten positiven und negativen Beispiele.

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



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

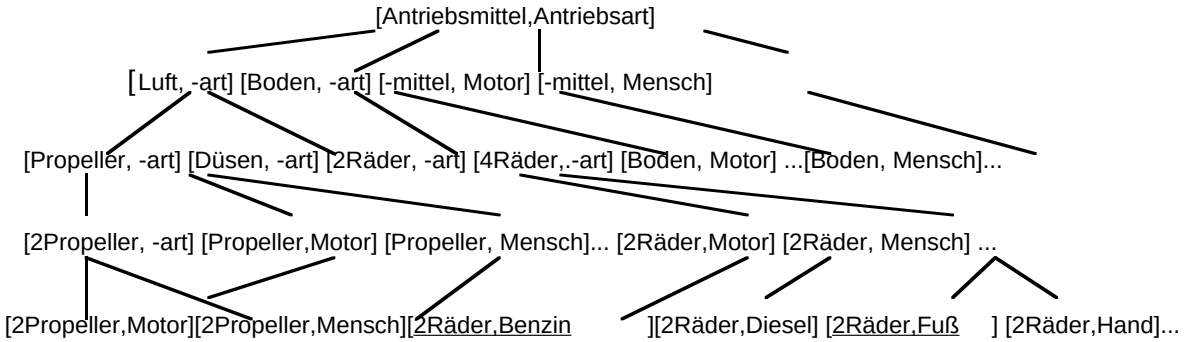
P: { [2Räder, Benzin], [4Räder, Diesel], [2Räder, Fuß]}

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 cover für die zwei Merkmale ist folgendermaßen:

covers([a,b],[c,d]) gdw. covers(a,c) & 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:



In diesem Beispiel kann man das Abgleichsprädikat durch die Vorgängerrelation zwischen Knoten im Merkmalsbaum definieren. Der strukturierte Suchraum entsteht dann beim Abgleichen. Obendrein muß die schrittweise Generalisierung und Spezialisierung formuliert werden sowie der globale Ablauf. In Prolog läßt sich das leicht machen. Ein Versionenraum-Programm verhält sich etwa so:

?- POSITIVES BEISPIEL?

[2Räder, Benzin]

G: {[-mittel, -art]} S: {[2Räder, Benzin]}

BEISPIEL?

[3Düsen, Hand] n

G: { [Boden, -art], [-mittel, Motor]} S: { [2Räder, Benzin]}

BEISPIEL?

[4Räder, Diesel] p

G: { [Boden, -art], [-mittel, Motor]} S: { [Boden, Motor]}

BEISPIEL?

[2Räder, Fuß] p

G: {[Boden, -art]} S: {[Boden, -art]}

LÖSUNG: [BODEN, -ART]

5.5 Lernen als Funktionsapproximation

Viele Lernprobleme lassen sich als Funktionsapproximation auffassen (z.B. das Begriffslernen, s. 5.3.1). Das Ziel ist es, eine Hypothese zu finden, die zur Vorhersage von zukünftigen Ereignissen benutzt werden kann. Gegeben sind Trainingsbeispiele 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 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. Trainingsbeispiele sind die Erfahrungen mit anderen Kunden. 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:

Es existieren:

- 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 einen Wert $y_i=t(x)$ zuweist. Im allgemeinen ist die Funktion t(x) nicht deterministisch, sondern liefert einen Wert y_i nur mit einer Wahrscheinlichkeit $P(y_i|x)$.
- Eine Hypothesensprache L_H.

Das Ziel ist:

- Die Hypothese h aus H, welche den folgenden Ausdruck (den sog. "zu erwartenden Fehler" oder auch "zu erwartende Risiko") minimiert

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

$P(x_i)$ 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. 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 vohergesagten Wertes h(x) der Hypothese vom Sollwert t(x).

$$Q(x_k,h)=\left[t(x_k)-h(x_k)\right]^2$$

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 Trainingsbeispiele gegeben, die vom Generator anhand von $P(x_i)$ gezogen wurden und für die wir $t(x_i)$ kennen. Diese Trainingsbeispiele werden dazu benutzt, den zu erwartenden Fehler R(h) mit dem "beobachteten Fehler" $R_{emp}(h)$ zu approximieren[30]. Der beobachtete Fehler für eine Menge von Trainingsbeispielen $[x_1,t(x_1)]$, ..., $[x_n,t(x_n)]$ 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 "Empirical Risk Minimization" (ERM) genannt.

---

[30] Diese Art der Approximation ist allerdings nur für große Anzahlen von Trainingsbeispielen verläßlich. Eine Verbesserung wird durch die gleichzeitige Betrachtung der Komplexität des Hypothesenraumes und der Anzahl der Trainingsbeispiele erreicht (s. Vapnik 1995). Man kann dann berechnen, wie sehr R(h) und $R_{emp}(h)$ voneinander abweichen.

Gegeben:

- Eine Menge von Trainingsbeispielen $[x_1,t(x_1)]$, ..., $[x_n,t(x_n)]$.
- Hypothesensprache L_H.

Gesucht:

- Die Hypothese h aus H, für die der beobachtete Fehler $R_{emp}(h)$ minimal ist.

Der in 5.5 vorgestellte Backprop Algorithmus für neuronale Netze folgt diesem Prinzip sehr direkt. Die Hypothesensprache des Netzes sind alle möglichen Kombinationen von Gewichten in den Neuronen des Netzes.

5.6 Neuronale Netze: Backprop

Von den hier vorgestellten Verfahren realisiert das Backprop Verfahren für neuronale Netze das ERM Prinzip am direktesten. Die Lernaufgabe ist:

Gegeben:

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

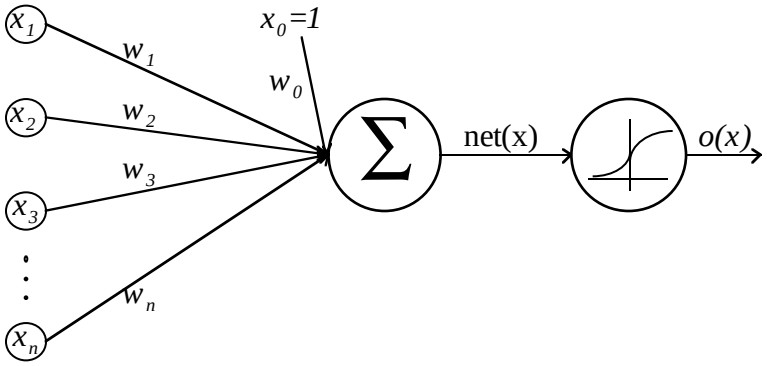
Ziel:

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

Der im weiteren vorgestellte **Backpropagation** Algorithmus ist in der Lage, sowohl Klassifikations- als auch Regressionsprobleme zu bearbeiten. Der Backpropagation-Algorithmus stellt nur eine Methode aus dem Bereich der neuronalen Netze dar und wird hier exemplarisch behandelt. Eine Übersicht über weitere Verfahren findet sich in (Rieger (1993).

Bei der Betrachtung von neuronalen Netzen muß zwischen einer biologisch und einer durch das maschinelle Lernen motivierten Herangehensweise unterschieden werden. 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 Backpropagation-Algorithmus vergleichbar sind.

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 Backpropagation-Algorithmus benutzt Neuronen des folgenden Typs (genannt „sigmoides Neuron“).



Ein Neuron besitzt einen Vektor von aktuellen Eingabewerten $\vec{x} = (x_1,...,x_n)$ 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

$$net(\vec{x}) = \sum_{i=1}^{n} w_i \cdot x_i$$

Um die Ausgabe $o(\vec{x})$ des Neurons für den Eingabevektor $\vec{x}$ zu berechnen, wird die Summe $net(\vec{x})$ durch eine sigmoide Funktion $\sigma(x)$ auf den Wertebereich $[0..1]$ normiert.

$$o(\vec{x}) = \sigma(net(\vec{x})) = \frac{1}{1+e^{-net(\vec{x})}}$$

Diese nichtlineare Transformation des Ausgabewertes wird benutzt, da durch sie der unten beschriebene Trainingsalgorithmus anwendbar wird und so effizientes Lernen ermöglicht. Wird $\sigma(x)$ weggelassen und $net(\vec{x})$ direkt als Ausgabe verwandt, spricht man von linearen Neuronen. Allerdings ist die Menge der Funktionen, die Netze mit linearen Neuronen darstellen können, auf lineare Funktionen reduziert. Erst die nichtlineare Ausgabefunktion $\sigma(x)$ ermöglicht dem neuronalen Netz das effiziente Lernen von komplizierten nichtlinearen Funktionen.

Sehr häufig werden Netze der folgenden *feedforward* **Struktur** 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-Schicht 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. Nehmen wir an wir hätten

die Attribute (vgl. das Beispiel aus Abschnitt 5.7.1) `Größe`, `Filmart` und `Farbigkeit`. Um die Werte der Attribute in das Netz eingeben zu können, müssen sie auf Zahlen abgebildet werden. Für das Attribut `Größe` mit den Werten `groß` und `klein` wird z. B. `groß` auf 1 und `klein` auf 0 abgebildet. Die anderen Attribute werden auf ähnliche Weise kodiert.

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 Ausgabe-Schicht ist die Klassifikation des Beispiels. Vielfach besteht die Ausgabe-Schicht auch aus mehreren Neuronen und gibt somit einen Vektor aus.

Man unterscheidet zwei Durchlaufrichtungen durch das Netz: Vorwärtgerichtet und rückwärtsgerichtet. Beim **Vorwärtsdurchlauf** werden die Ausgaben der Ausgabe-Schicht ausgehend von den Eingaben der Eingabe-Schicht berechnet. Hierzu werden zuerst die Neuronen der Mittel-Schicht berechnet und danach die der Ausgabe-Schicht. Die Ausgabe ist eine Zahl im Bereich $[0..1]$ und kann durch einen Schwellwert (z. B.: wenn größer 0.5, dann positiv, sonst negativ) 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 Trainingsbeispielen E werden die Gewichte der Neuronen mit einem stochastisches Gradienten-Abstiegsverfahren so eingestellt, daß die folgende Fehlerfunktion F minimiert wird.

$$F = \sum_{e \in E} [t(e) - o(e)]^2$$

$t(e)$ ist die Klassifikation des Trainingsbeispiels e und $o(e)$ ist die Ausgabe des Netzes. Der folgende Algorithmus beschreibt die Lernphase des Netze und die damit verbundene Veränderung der Gewichte.

Bis der Fehler ‘klein genug’ ist wiederhole für jedes Trainingsbeispiel $e \in E$:

Vorwärtsdurchlauf:

1. Berechne $o(e)$ mit einem Vorwärtsdurchlauf.

Rückwärtsdurchlauf:

2. Für alle Neuronen $k \in \text{Output - Schicht}$:
$$\delta \leftarrow o_k(e) \cdot (1 - o_k(e)) \cdot (t_k(e) - o_k(e))$$

3. Für Neuronen $k \in \text{Hidden - Schicht}$: $\delta := o_k(e) \cdot (1 - o_k(e)) \cdot \sum_{k \in \text{Output-Schicht}} w_{kh} \cdot \delta_k$

4. Verändere alle Gewichte anhand der Regel:
$$w_{ji} \leftarrow w_{ji} + \eta \cdot \Delta w_{ji} \text{ und } \Delta w_{ji} \leftarrow -\delta_j \cdot x_{ji}$$

x_{ji} ist der i-te Eingabewert von Neuron j. w_{ji} ist das entsprechende Gewicht. Die Idee des Algorithmus ist es, bei jedem Beispiel die Gewichte in kleinen Schritten so zu verändern, daß der Fehler geringer wird. Δw_{ji} beschreibt die Richtung der Veränderung und die Lernrate η gibt die Schrittgröße an. Die Lernrate sollte

„angemessen“ gewählt werden. Ein großes η verringert die Trainingszeit. Wird es jedoch zu groß gewählt, kann es sein, daß die Gewichte nicht konvergieren. $\Delta w_{ji} = \frac{\delta E_e}{\delta w_{ji}}$ ist der Gradient (die Ableitung) der Fehlerfunktion $F_e = [t(e) - o(e)]^2$ für Trainingsbeispiel e mit Bezug auf das Gewicht w_{ji}. Eine ausführliche Herleitung dieses Algorithmus findet sich in Mitchell (1997). Da es sich bei dem Algorithmus um ein Bergsteigeverfahren handelt, ist es nicht garantiert, daß die optimale Einstellung der Gewichte gefunden wird. Obwohl der Algorithmus in lokalen Minima steckenbleiben kann, liefert er in der Praxis oft akzeptable Lösungen.

Nachdem die Gewichte anhand der Trainingsdaten eingestellt worden sind, können durch Vorwärtsdurchläufe auch neue Beispiele klassifiziert werden.

5.7 Begriffslernen in Aussagenlogik

In diesem Abschnitt werden zwei klassische induktive Lernverfahren beschrieben. Das erste ist ein top-down Lernen aus Beispielen mit statistischer Merkmalsselektion. Es stellt die Erkennungsfunktion für einen Begriff als Entscheidungsbaum dar. Ein Entscheidungsbaum hat Kanten, an denen Attributwerte stehen, Knoten sind Verzweigungspunkte und Blätter stellen Begriffsnamen dar. Um zu entscheiden, ob ein neues Beispiel zu einem Begriff gehört, wird den Kanten gefolgt, deren Beschriftung einem Attributwert des Beispiels entspricht, bis ein Blatt erreicht ist. Das Beispiel wird dem Begriff zugeordnet, der an dem Blatt angegeben ist. Diese Verfahren heißen "top-down induction of decision trees". Die bekannteste Realisierung ist ID3 (Quinlan 1986) bzw. C4.5 (Quinlan 1993).

Das zweite Verfahren, *conceptual clustering*, lernt top-down aus Beobachtungen, d.h. der Benutzer muß keine Begriffszugehörigkeit angeben. Auch dieses Verfahren enthält eine statistische Bewertungsfunktion. Dabei entsteht eine Hierarchie von Begriffen unter einem Oberbegriff. Es gibt verschiedene conceptual clustering Verfahren. Hier wird die Star-Methode vorgestellt. Die Methode wurde von Michalski entwickelt, eine Realisierung ist CLUSTER (Michalski, Stepp 1983).

5.7.1 ID3

Die Lernaufgabe für top-down Induktion von Entscheidungsbäumen ist:

Gegeben:

eine Menge von Beispielen in einer Attribut-Werte-Repräsentation

Ziel:

ein Entscheidungsbaum, der ein neues Objekt klassifiziert

Diese Lernaufgabe ist eine Spezialisierung der oben genannten Beschreibung von Lernen als Suche. Die Beispielbeschreibungssprache ist durch eine Liste von Attributen mit ihren möglichen Werten gegeben. Die Hypothesensprache verwendet dieselben Attribute in einem mächtigeren Formalismus, dem Entscheidungsbaum.

Der Kern des Verfahrens ist die Bewertung des Informationsgewinns eines Attributes für die Klassifikation eines Objektes. Wie gut kann ich ein Objekt klassi-

fizieren, ohne ein Attribut zu kennen? Wie gut kann ich klassifizieren, wenn ich den Wert eines bestimmten Attributes kenne? Welches Atribut bringt den größten Informationsgewinn? Aus der Liste aller Attribute wird zunächst das mit dem größten Informationsgewinn gewählt, angewandt und dann aus der Liste entfernt. Für die verbleibenden Attribute wird dann wieder genauso verfahren, bis schließlich kein weiteres Attribut mehr Informaionen liefert - oder die Liste leer ist. Der Informationsgehalt eines Attributs wird durch die Entropie angegeben:

$$-\sum_{m=1}^{n} p_m \log_2 p_m$$

bei n verschiedenen Attributwerten und der Wahrscheinlichkeit p_m für den m-ten Wertes. Man vergleicht dann den Informationsgehalt der verschiedenen Attribute mit dem Informationsgehalt der Beispielmenge selbst (also ohne Attribute) und wählt das informativste Attribut oder gar keines. Natürlich kann die Bewertung auch anders gewählt werden, z.B. nach Bayes oder im Sinne der Textkompression (wieviele Zeichen brauche ich bei minimaler Codierung, um etwas auszudrücken - das ist der Informationsgehalt). Dieses Forschungsthema soll hier aber nicht behandelt werden.

Der Algorithmus von ID3:

ID3(K,C,A)

Knoten K, Menge von Beispielen C und Liste von Attributen A

1. Wenn
   - Attributliste leer ist und nicht alle Beispiele in C gehören zum gleichen Begriff, dann Fehlermeldung.
   - alle Beispiele in C zum gleichen Begriff gehören, dann ist K ein Blatt und wird mit dem Begriff beschriftet.
   - C leer ist, dann Fehlermeldung.
2. Wähle das Attribut a mit maximalem Informationsgewinn.
3. Reduziere die Attributliste A um a. Wir erhalten A'.
4. Für alle Attributwerte w_i von a:
   - Hänge Knoten K_i and K an und beschrifte ihn mit w_i.
   - Ermittle Menge der Beispiele C_i aus C, deren Attribut a den Wert w_i hat.
   - Rufe ID3(K_i,C_i,A') rekursiv auf.

Ein Beispiel soll den Algorithmus verdeutlichen. Als Attributliste für die Beschreibung eines Sachbereichs von fotographischen Aufnahmen für bestimmte Zwecke ist die folgende Attributliste mit den zugehörigen Attributwerten gegeben:

Attributliste:

(Größe {groß, klein},

Filmart {Foto, Dia},

Farbigkeit {s_w, bunt})

Die Beispiele sind:

{ (groß, Dia, s_w, -), (klein, Dia, s_w, -), (groß, Dia, bunt,-),

(groß, Foto, s_w, +), (groß, Foto, bunt,-), (klein, Foto, bunt,-) }

Ohne ein Attribut zu kennen, ist die Menge der Beispiele bereits ziemlich geordnet, weil 5 von 6 Beispielen negativ sind und nur eines positiv ist. Die Entropie der Beispielmenge ist 0,649:

$\frac{5}{6}\log_2\frac{5}{6} = -0{,}219$ für negative Beispiele

$\frac{1}{6}\log_2\frac{1}{6} = -0{,}430$ für das positive Beispiel

Summe mit umgekehrtem Vorzeichen: 0,649

Der Informationsgewinn, wenn wir den Wert des Attributs Größe kennen, ergibt sich aus der Entropie ohne Attribut minus der Entropie für Größe:

groß

(groß, Dia, s_w, -), (groß, Dia, bunt,-),(groß, Foto, s_w, +), (groß, Foto, bunt,-)

$\frac{3}{4}\log_2\frac{3}{4} = -0{,}311$ für negative Beispiele, die als groß beschrieben sind

$\frac{1}{4}\log_2\frac{1}{4} = -0{,}5$ für das positive Beispiel

Die Summe mit umgekehrtem Vorzeichen ist 0,811.

klein

(klein, Dia, s_w, -), (klein, Foto, bunt,-)

$\frac{2}{2}\log_2\frac{2}{2} = 0$ für die beiden negative Beispiele, die als klein beschrieben sind.

Für das Attribut Größe ergibt sich also:

$\left(\frac{4}{6}0{,}811\right)+\left(\frac{2}{6}0\right) = 0{,}541$ als Entropie.

Der Informationsgewinn durch dieses Attribut ergibt sich aus dem Vergleich mit der Entropie ohne ein Attribut:

0,649 - 0,541 = 0,108

Dieselbe Berechnung muß für alle anderen Attribute durchgeführt werden, damit dann das Attribut mit dem größten Informationsgewinn ausgewählt werden kann. Für Filmart ergibt sich:

Foto:

(groß, Foto, s_w, +), (groß, Foto, bunt,-), (klein, Foto, bunt,-)

$\frac{2}{3}\log_2\frac{2}{3} = -0{,}399$ für die zwei negativen Beispiele

$\frac{1}{3}\log_2\frac{1}{3} = -0{,}528$ für das positive Beispiel

Die Summe mit umgekehrtem Vorzeichen ist: 0,927.

Dia:

(groß, Dia, s_w, -), (klein, Dia, s_w, -), (groß, Dia, bunt,-)

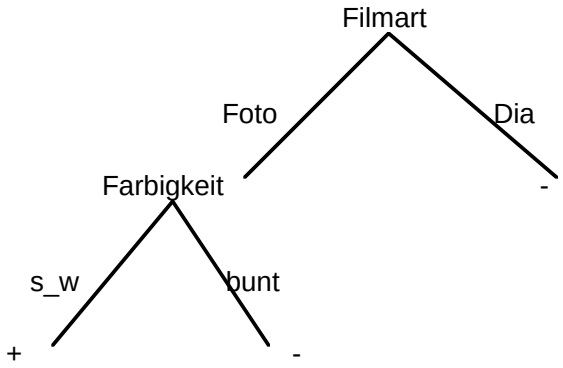
$\frac{3}{3}\log_2\frac{3}{3} = 0$ für die drei negativen Beispiele.

Es ergibt sich also als Entropie für Filmart:

$\left(\frac{3}{6}0{,}927\right)+\left(\frac{3}{6}0\right) = 0{,}463$

Der Informationsgewinn ist: 0,649 - 0,463 = 0,185. Damit ist Filmart besser geeignet, die Daten zu ordnen als Größe.

Wenn Filmart ausgewählt wurde, werden zwei Kanten angelegt und mit Foto bzw. Dia beschriftet. Der unter Dia gebildete Knoten enthält nur negativ klassifizierte Beispiele und wird damit zum Blatt. Der unter Foto gebildete Knoten muß genauso wie der oberste behandelt werden. Am Ende ergibt sich der folgende Entscheidungsbaum:



Ein neues Objekt, zum Beispiel (klein, Foto, s_w), wird nach diesem Entscheidungsbaum nun klassifiziert (hier: positiv).

Man kann einen Entscheidungsbaum in eine Menge von Regeln übersetzen. Diese Regelmenge kann in einem Nachbearbeitungsschritt optimiert werden.

5.7.2 Conceptual Clustering

Conceptual clustering ist aus dem statistischen Verfahren der *cluster* Analyse hervorgegangen. Im Gegensatz zur Statistik, die lediglich numerische Angaben zurück liefert, wird aber beim maschinellen Lernen die Auswertung der Evaluation in Form verständlicher Begriffscharakterisierungen ausgegeben. Zum Beispiel werden die Begriffe in einer Hierarchie angeordnet. Die Lernaufgabe unterscheidet sich von der top-down Induktion von Entscheidungsbäumen dadurch, daß der Aggregationsschritt dazugehört. Es wird also nicht aus Beispielen, sondern aus Beobachtungen gelernt.

Ich beschreibe hier lediglich das star-Verfahren (Michalski, Stepp 1986). Bekannte Verfahren sind aber auch COBWEB (Fisher 1987) und UNIMEM (Lebowitz 1987). Die Beschreibungssprache für Beobachtungen ist eine offene Prädikatenlogik (also: ohne Quantoren bzw. nur mit All-Quantoren), wobei Sorten vorgegeben werden. Zum Beispiel gibt es nominale, lineare (geordnete) und hierarchische Wertebereiche für Variablen.

Ein **cluster** ist eine intensional definierte Menge von Beobachtungen, also eine Begriffscharakterisierung. Ein **star** ist eine abgrenzende Beschreibung, also ein elementares cluster.

Die Grundidee des Verfahrens ist die Abgrenzung von Mengen von Beobachtungen. Dazu werden zunächst k beliebige Beobachtungen gewählt. Oft ist k=2, so daß ein binärer Baum von Begriffen gebildet wird. Die ausgewählten Beobachtungen werden dann gegeneinander abgegrenzt, d.h. es werden Charakterisierungen gefunden, die die beiden Beobachtungen unterscheiden. Dieser Schritt ist die star-Bildung. Aus solchen Charakterisierungen wird eine überschneidungsfreie Abdeckung aller Beobachtungen durch k Begriffe konstruiert. Wählt man recht ähnliche Beobachtungen, so erhält man eher typische Charakterisierungen, deckt aber vielleicht nicht gut genug alles ab. Wählt man sehr unterschiedliche Beobachtungen, deckt man vermutlich viele Beobachtungen gut ab, verpaßt aber vielleicht abgrenzende Merkmale. Ein Bewertungskriterium entscheidet, ob die Menge der Beobachtungen hinlänglich strukturiert ist, oder nicht. Wenn nicht, werden noch einmal andere Ausgangsbeobachtungen gewählt, mit denen die Schritte noch einmal durchlaufen werden. Die gefundenen Begriffe werden weiter verfeinert, indem das Verfahren auf alle von dem jeweiligen Begriff abgedeckten Beobachtungen angewandt wird. Das Verfahren hält an, sobald das Bewertungskriterium erfüllt ist und so viele Ebenen von Begriffen gebildet wurden wie vom Benutzer gefordert. Das Verfahren für eine Ebene von Begriffen noch einmal im Überblick:

Star-Methode:

1. Wahl von k Ausgangsbeobachtungen
2. Bestimmung des star für jede Ausgangsbeobachtung gegen die andere(n)
3. Konstruktion einer disjunkten Abdeckung
4. Evaluierung:

   wenn das Bewertungskriterium erfüllt ist, alle Charakterisierungen ausgeben und zur nächsten Ebene übergehen.

wenn das Bewertungskriterium nicht erfüllt ist, für neue Ausgangsbeobachtungen wieder die Schritte 1.-4. ausführen.

Anhand eines einfachen Beispiels soll das Verfahren genauer vorgestellt werden. Dabei nehmen wir Beobachtungen an, die alle durch eine Relation r(X,Y) beschrieben werden, wobei die Wertebereiche für X und Y beide vom Typ "nominal" sind, d.h. die möglichen konstanten Terme werden aufgezählt.

X: { Video, 16mm, Super8}

Y: { Spiel, Trick, Dokumentar}

Es gibt also potentiell 9 Beobachtungen, die mit der Relation ausgedrückt werden können. Nehmen wir an, die folgenden vier Beobachtungen wären gegeben:

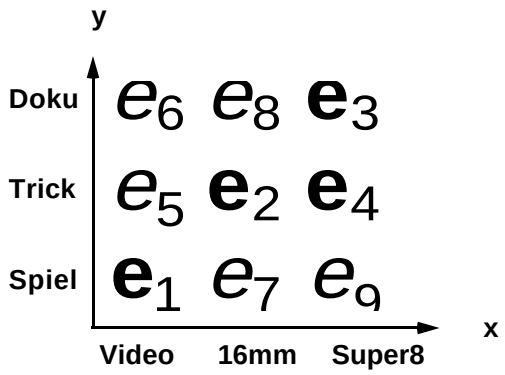
e1: r(Video, Spiel)

e2: r(16mm, Trick)

e3: r(Super8, Dokumentar)

e4: r(Super8, Trick)

Wir können die möglichen Beobachtungen in einer zweidimensionalen Graphik darstellen, wobei die tatsächlichen Beobachtungen fett gedruckt sind.



Wie kann man diesen Bereich nun strukturieren?

Wählen wir als Ausgangsbeobachtungen e1 und e4. Der erste Schritt ist jetzt die *star*-Bildung. Sie soll die Unterschiede zwischen Beobachtungen deutlich machen. Zunächst wird maximal generalisiert, danach soweit als nötig spezialisiert. Ein *star* wird notiert als G(b1|b2), wobei b1 gegen b2 abgegrenzt wird, d.h. es wird alles notiert, was b2 nicht hat, aber b1. Ein *star* besteht aus einer Disjunktion von Merkmalen. Diese Disjunktion ist die maximale Generalisierung, die gerade b2 noch ausschließt. Es werden bei zwei Ausgangsbeobachtungen zwei *stars* gebildet.

Im Beispiel:

G(e1|e4): $r(\neg\ \text{Super8}, Y) \vee r(X, \neg\text{Trick})$

$= r(\text{Video} \vee \text{16mm}, Y) \vee r(X, \text{Spiel} \vee \text{Dokumentar})$

Damit sind e1, e2 und e3 abgedeckt und nur e4 ist ausgeschlossen. Von den möglichen Beobachtungen sind e5, e6, e7, e8 und e9 abgedeckt.

G(e4|e1): r(¬Video, Y) ∨ (X, ¬ Spiel)

= r(16mm ∨ Super8, Y) ∨ (X, Trick ∨ Dokumentar)

Damit sind e2, e3 und e4 abgedeckt, nur e1 ist ausgeschlossen.

Nun werden diese beiden *stars* G spezialisiert zu RG. Bisher war die Relation immer nur an einer Argumentstelle eingeschränkt worden. Jetzt wird zu jeder Einschränkung einer Argumentstelle eine passende Einschränkung der anderen gesucht. Dazu werden die Y-Werte zu den im *star* angegebenen X-Werten aufgesammelt. Zu Video oder 16mm gibt es nur Spiel oder Trick. Entsprechend werden zu den X-Werten die vorkommenden Y-Werte aufgesammelt. Also ergibt sich

RG(e1|e4): r(Video ∨ 16mm, Spiel ∨ Trick) ∨

r(Video∨ Super8, Spiel ∨ Dokumentar)

Damit wird e1, e2 und e3 immer noch abgedeckt. Die Spezialisierung ist nur an den möglichen Beobchtungen zu erkennen. Es ist jetzt e8 ausgeschlossen.

RG(e4|e1): r(16mm ∨ Super8, Trick ∨ Dokumentar)

Damit sind e2, e3, e4 abgedeckt. Von den vorher auch abgedeckten möglichen Beobachtungen sind jetzt e5, e6, e7 und e9 nicht mehr abgedeckt.

Bei G(e4|e1) ergibt sich kein Unterschied für RG(e4|e1), ob nun von gegebenen X-Werten aus nach Y-Werten oder von gegebenen Y-Werten nach X-Werten gesucht wird.

Auch die spezialisierten *stars* sind noch nicht überschneidungsfrei für alle (also auch die möglichen) Beobachtungen. Jedes Disjunkt eines *stars* wird mit jedem Disjunkt des anderen *stars* verglichen. So deckt r(Video ∨ 16mm, Spiel ∨ Trick) e1, e2, e5 und e7 ab und r(16mm ∨ Super8, Trick ∨ Dokumentar) deckt e2, e3, e4 und e8 ab. Sie werden im nächsten Schritt durch Einschränkungen eines Terms disjunkt gemacht. Im Beispiel soll e2 von nur einem *star* abgedeckt werden, muß also aus dem anderen ausgeschlossen werden. Dabei gibt es verschiedene Möglichkeiten, dies zu tun. Hier wird das Verfahren exponentiell. Es wird die erste gefundene Einschränkung gewählt und erst die Qualitätsbewertung der Begriffsdefinition entscheidet, ob diese Möglichkeit gut genug war. Im Beispiel kann RG(e4|e1) eingeschränkt werden zu

r(Super8, Trick ∨ Dokumentar)

Aber auch das zweite Disjunkt von RG(e1|e4), r(Video∨ Super8, Spiel ∨ Dokumentar), und RG(e4|e1), r(16mm ∨ Super8, Trick ∨ Dokumentar), überschneiden sich. Um dies überschneidungsfrei zu bekommen, kann RG(e4|e1) eingeschränkt werden auf:

r(16mm ∨ Super8, Trick)

Damit sind nun *cluster* gebildet, und es stehen zwei alternative Begriffsdefinitionen zur Bewertung an:

1) r(Video ∨ 16mm, Spiel ∨ Trick) für *cluster* um e1

deckt e1, e2, e5 und e7 ab

r(Super8, Trick ∨ Dokumentar) für *cluster* um e4

deckt e3 und e4 ab

2) r(Video∨ Super8, Spiel ∨ Dokumentar) für *cluster* um e1

deckt e1, e3, e6 und e9 ab

r(16mm ∨ Super8, Trick) für *cluster* um e4

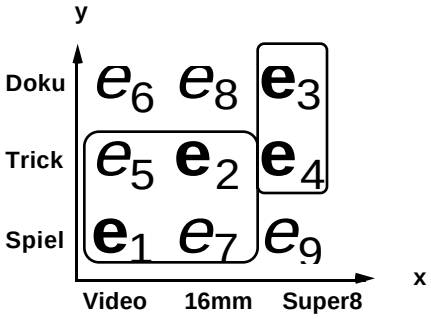
deckt e2 und e4 ab

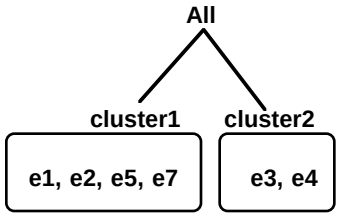
Die Bewertungsfunktion kann vom Benutzer vorgegeben werden. Michalski nennt sie **lexical evaluation function** (LEF). Ein einfaches Maß für ein *cluster* ist:

$$1 - \frac{\text{Anzahl abgedeckter Beobachtungen}}{\text{Anzahl abgedeckter Objekte}}$$

Für die Bewertung der Begriffsqualität werden die Bewertungen zusammengehöriger *cluster* addiert. In unserem Beispiel sind jeweils vom ersten *cluster* um e1 zwei tatsächlich beobachtete Objekte abgedeckt und insgesamt vier. Der *cluster* um e4 deckt im ersten und zweiten Fall zwei Objekte ab, die beide auch beobachtet wurden. Damit ergibt sich für beide Fälle dieselbe Bewertung von (1- 2/4) + (1- 2/2) = 1/2. Diese Bewertung erlaubt hier also keine Auswahl zwischen den Alternativen, und wir können eine der beiden Definitionen beliebig wählen.

Das Ergebnis einer Iteration ist die Aufteilung des gesamten Bereichs in zwei *cluster*: Hier: r(Video ∨ 16mm, Spiel ∨ Trick) und r(Super8, Trick ∨ Dokumentar). Diese Begriffe können nun verfeinert werden, indem innerhalb von ihnen wieder *cluster* gebildet werden. In unserem Beispiel macht es wohl keinen Sinn, da die abgedeckten Bereiche bereits sehr klein sind. In realen Anwendungen mit Hunderten von Beobachtungen wird der Algorithmus auf jedes *cluster* erneut angewandt bis eine Mindestanzahl abgedeckter Beobachtungen unterschritten ist.





In der graphischen Darstellung sieht man deutlich, daß dies Begriffspaar nicht alle möglichen Beobachtungen erfaßt. Es kann also nicht vollständig klassifizieren. Bei einem anderen Bewertungsmaß, das die Anzahl aller Objekte mit einbezieht, könnten beide alternativen Begriffspaare abgelehnt werden, weil sie nicht den gesamten Bereich aller möglicher Beobachtungen abdecken. Damit ist

die **Vorhersagekraft** (predictiveness) eingeschränkt: nicht alle möglichen Beobachtungen können mit den Begriffen klassifiziert werden. Man könnte also noch einmal andere Ausgangsbeobachtungen wählen, mit denen erst allgemeine *stars* gebildet und dann spezialisiert werden. Zum Beispiel kann man gegensätzlichere Ausgangsbeoachtungen wählen, etwa e1 und e3. Wenn auch die neuen Begriffspaare nicht genügend Vorhersagekraft haben, kann man die Bewertungen der beiden Iterationen vergleichen. Hat sich die Qualität immerhin verbessert, werden für den nächsten Versuch zentrale Beobachtungen zum Ausgangspunkt gemacht (z.B. e2 und e4), hat sie sich weiter verschlechtert, gibt es für diese begrenzte Menge von Beobachtungen keine Alternative mehr (andere gegensätzliche Beobachtungspaare, hier: e6 und e9, wurden nicht beobachtet). Man kann dann annehmen, daß entweder die nicht erfaßten möglichen Beobachtungen tatsächlich nicht vorkommen können oder die Begriffsbildung nicht erfolgreich war.

Das Bewertungsmaß entscheidet, ob das *star*-Verfahren vollständige Begriffsdefinitionen lernt. Die Korrektheit ist im Gegensatz zu Verfahren, die aus Beispielen lernen, schwieriger festzustellen: es gibt keine vorgebene Einteilung der Beobachtungen in Begriffe, mit der die gefundene Einteilung verglichen werden könnte.

5.8 Deduktives Lernen

Für **erklärungsbasiertes Lernen** ist die Lernaufgabe nicht, Beispiele oder Beobachtungen zu einer Begriffsdefinition zu verallgemeinern, sondern eine Begriffsdefinition für eine Anwendung zu operationalisieren. Wenn eine Begriffsdefinition in einer Terminologie vorliegt, die erst mühsam aus der von der Anwendung vorgegebenen Terminologie gewonnen werden muß, ist die Neudefinition des Begriffs in Anwendungstermini eine Operationalisierung.

Gegeben:

Zielbegriff mit einer Definition

Übungsbeispiel: positives Beispiel für den Zielbegriff

Sachbereichstheorie

Operationalitätskriterium: ein Prädikat, das entscheidet, welche Termini zur Neudefinition des Zielbegriffs herangezogen werden dürfen.

Ziel:

Eine Definition des Zielbegriffs, die dem Operationalitätskriterium gehorcht.

Die Idee dabei ist, daß eine Lösung (Übungsbeispiel) anhand des Wissens (Sachbereichstheorie) nachvollzogen wird. Dabei wird die Sachbereichstheorie mit den Angaben zum Beispiel in Verbindung gebracht. Aus dieser Verbindung werden dann die operationalen Bestandteile herausgezogen und für zukünftige Beispiele zur Klassifikation genutzt. Es handelt sich also um ein sicheres Lernverfahren: es wird deduziert, daß das Übungsbeispiel von dem Zielbegriff abgedeckt wird. Die Deduktionsschritte werden für zukünftige Beispiele direkt genutzt. Es findet keine Generalisierung statt. Der operationale Begriff ist eine Spezialisierung. Da die Sachbereichstheorie erhalten bleibt, können aber auch alle Beispiele, die vor dem Lernen klassifizierbar waren, weiterhin klassifiziert werden.

Das einfache Verfahren von Mitchell (1985) verwendet einen Theorembeweiser, um das Übungsbeispiel aus der Sachbereichstheorie abzuleiten. In dem Beweispfad werden dann mithilfe der Substitutionen Variablen eingeführt. Die Termini, in denen das Übungsbeispiel beschrieben ist, werden als operational definiert. Die Blätter des Beweisbaumes ergeben dann den operationalen Begriff.

Die drei in der Literatur immer wieder angeführten Beispiele für dieses Verfahren betreffen Mord und Selbstmord, Tassen sowie die Stapelbarkeit von Objekten. Letzteres wird hier vorgeführt:

Zielbegriff:

leichter(X,Y) --> stapelbar(X,Y)

Übungsbeispiel:

auf(obj1, obj2)
isa(obj1, kiste)
isa(obj2, tisch)
farbe(obj1, rot)
farbe(obj2, blau)
volumen(obj1, 1)
dichte(obj1, 0.1)

Sachbereichstheorie:

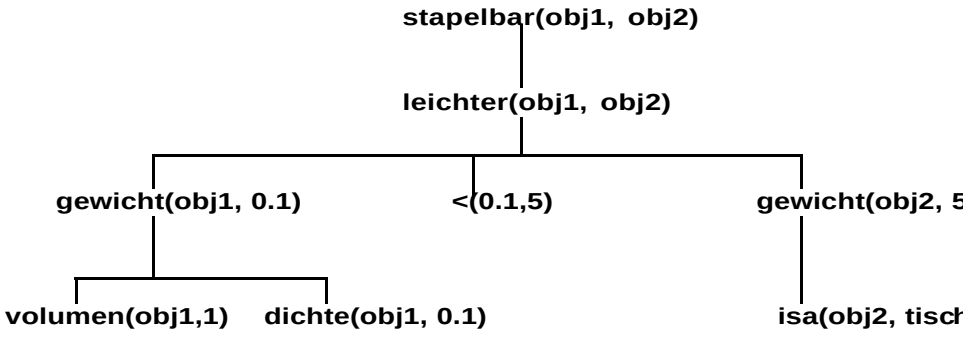
volumen(P,V) & dichte(P,D) --> gewicht(P, V*D)
gewicht(P, W1) & gewicht(Q, W2) & W1 < W2 --> leichter(P,Q)
isa(P,tisch) --> gewicht (P,5)
0.1 < 5

Operationalitätskriterium:

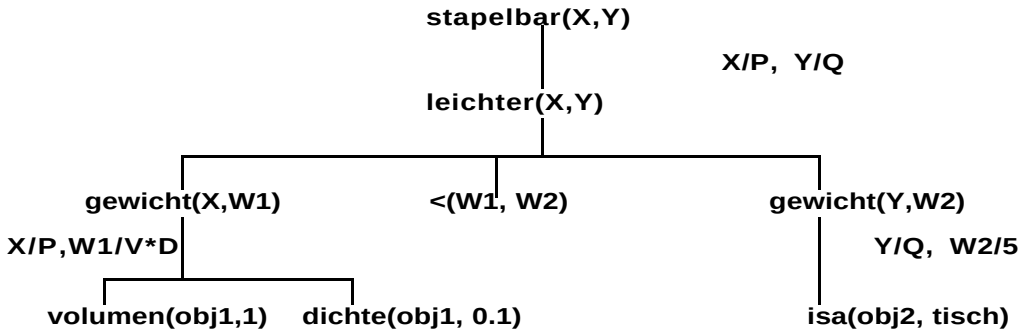
volumen, dichte, < und isa sind operational und sonst keine Prädikate.

Mithilfe der Sachbereichstheorie kann nachgewiesen werden, daß die Kiste auf den Tisch stapelbar ist.

Der Beweisbaum sieht folgendermaßen aus:



Dieser Beweis wird für die künftige Nutzung durch andere Beispiele verbessert, indem Variablen gemäß der Substitutionen, die im Beweis verwendet wurden, anstelle der Konstanten gesetzt werden.



Die Substitutionen sind in dem Beweisbaum angegeben.[31] Als operationale Begriffsdefinition ergibt sich:

volumen(X,V) & dichte(X; D) & V*D < 5) & isa (Y, tisch) --> stapelbar(X,Y)

Dieses Verfahren kann leicht in Prolog programmiert werden, wobei man Prolog als Theorembeweiser benutzt, der gleich die Substitutionen mitliefert (Mitchell, Kedar-Cabelli 1986). Falls die Sachbereichstheorie unübersichtlich ist und man stets Anwendungen einer speziellen Form hat, führt die operationale Definition zu einer Performanzsteigerung des Klassifikationssystems. In anderen Fällen jedoch nicht!

An dem Beispiel sind die Schwächen des Verfahrens gut zu erkennen. Das Operationalitätskriterium ist hier einfach eine Aufzählung von Prädikaten. In echten Anwendungen kann ein reicheres Kriterium nötig sein, das sich dann nicht mehr so einfach abprüfen läßt (DeJong, Mooney 1986).

Eigenarten des Übungsbeispiels, die vielleicht nicht immer in der Anwendung vorkommen, geraten genauso in die neue Begriffsdefinition wie die für die Anwendung wichtigen Eigenschaften. So ist hier der Gewichtsvergleich zwischen Tischen und allen anderen Objekten in der Begriffsdefinition enthalten. Das ist sinnvoll, wenn in der Anwendung grundsätzlich nur auf Tische etwas gestellt werden soll. Wenn aber die Operationalisierung der Definition darin bestehen sollte, daß das Prädikat leichter durch die Prädikate volumen, dichte und < ersetzt wird, so ist der neue Begriff zu speziell geworden. Wir müßten dann ein anderes Übungsbeispiel wählen, in dem auch das Gewicht des zweiten Objekts berechnet wird, so daß ein symmetrischer Beweisbaum entsteht. Die Wahl des Übungsbeispiel ist also entscheidend. Der neue Begriff kann auch deshalb zu speziell definiert sein, weil Prädikate nicht generalisiert werden. (DeJong, Mooney 1986).

Wenn wir ein solches Übungsbeispiel wählen würden, fehlt aber vielleicht in der Sachbereichstheorie der Vergleich der Gewichte (<) für diese Werte. Eine vollständige Sachbereichstheorie ist notwendig bei diesem Verfahren, da der Beweis gelingen muß. Erweiterungen erklärungsbasierten Lernens beschäftigen sich daher mit der Vervollständigung von Beweisen. Aus demselben Grund muß die Sachbereichstheorie natürlich konsistent sein.

---

[31] Für den Schritt der Variabilisierung wird das Mord- (tötet(X,Y)) bzw. Selbstmordbeispiel (tötet (X,X)) oft angeführt.

5.9 Induktives Lernen in Prädikatenlogik

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. Die ersten Ergebnisse von Plotkin (1971) waren wenig ermutigend: im allgemeinen Fall ist in der Prädikatenlogik erster Stufe nicht entscheidbar, ob die speziellste mit Hintergrundwissen und Beispielen konsistente und gemäß einer Interessantheitsordnung minimale Generalisierung gefunden wird! Inzwischen ist dieser Satz reformuliert worden: die Prädikatenlogik muß eingeschränkt werden, damit eine speziellste Generalisierung gefunden werden kann. Die aktuellen Ansätze unterscheiden sich zum einen darin, wie die Generalisierung definiert wird, zum anderen in den konkreten Einschränkungen der Prädikatenlogik.

5.9.1 Generalisierung

Bei den logik-basierten Verfahren geht es darum, genau anzugeben, wann ein Literal oder eine Klausel eine Generalisierung eines anderen Literals bzw. einer anderen Klausel darstellt. Wenn man die Generalisierungsbeziehung formalisieren kann, dann kann man hoffentlich auch ein Verfahren finden, das zu gegebenen Literalen bzw. Klauseln eine Generalisierung konstruiert. Und das wäre dann ein induktiver Schluß.

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

> Eine Klausel C1 ist genereller als eine andere, C2, geschrieben $C1 \geq C2$, wenn C1 --> C2 gilt.

> Eine Klausel C1 ist genereller als eine andere, C2, bezüglich einer Theorie T, wenn T, C1 --> C2 gilt.

Um dann 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.

Deshalb wird die schwächere **Subsumtionsbeziehung** bevorzugt. Die Subsumtion ist eine korrekte, aber unvollständige Ableitungsrelation. Eine Klausel C1 ist genereller als eine andere Klausel C2, $C1 \geq C2$, wenn gilt: C1 subsumiert C2. Bei Literalen ist das einfach.

> Ein Literal L1 subsumiert ein anderes Literal L2, genau dann wenn es eine Substitution θ gibt, so daß $L1\theta = L2$.

Wenn wir Klauseln als Mengen schreiben, so ist die generellere Klausel eine Teilmenge der spezielleren. Natürlich müssen die Terme so substituiert werden, daß es paßt. Dazu suchen wir die geeignete Substitution θ.

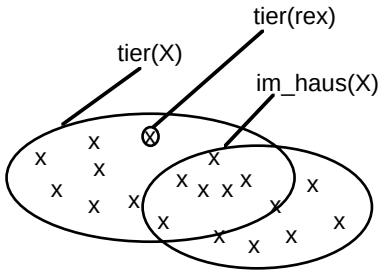
> Eine Klausel C1 subsumiert eine Klausel C2, $C1 \geq C2$, genau dann wenn $C2 \supseteq C1\theta$.

Damit wird Substitution und Teilmengenbeziehung zur Grundlage der Formalisierung von Induktion.

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

tier(X) --> säugetier (X) ≥ tier(rex) --> säugetier (rex) und

tier(X) --> säugetier (X) ≥ tier(X) & im_haus(X)--> säugetier (X).



Die Säugetiere umfassen einmal nur rex (und vielleicht noch andere Objekte), einmal mindestens die Schnittmenge der beiden Objektmengen, schließlich sogar mindestens alle Tiere.

{¬ tier(X), ¬ im_haus(X), säugetier(X)} ⊇ {¬ tier(X), säugetier(X)}.

An dem Beispiel ist auch deutlich zu sehen, daß neben der Teilmengenbeziehung, der Subsumtion, auch die logische Folgerung gilt. Die Faustregel lautet: je mehr Literale eine Klausel hat, desto spezieller ist sie. Das von Plotkin (1971) erkannte Problem läßt sich ebenfalls an dem Beispiel zeigen: Wenn durch eine Theorie gegeben ist, daß einige Literale gleichbedeutend sind mit einem anderen Literal, so hilft das einfache Abzählen nichts. Wenn für alle Tiere bekannt ist, daß sie sterblich sind, so wird eine der oben angeführten Klauseln über Tiere nicht spezieller, wenn sterblich(X) hinzugefügt wird. Es ist einfach redundant. Wir müssen das **Hintergrundwissen** also etwas raffinierter berücksichtigen.

C1 ist genereller als C2 bezüglich einer Theorie T, wenn in jeder Interpretation I, die T wahr macht, für alle Atome A gilt, daß, wann immer C2 auf A zutrifft, dann trifft auch C1 auf A zu.

Wir sehen also in der Interpretation[32] nach, welche Objekte von einer Klausel abgedeckt werden. Wir nehmen das Hintergrundwissen insofern hinzu, als wir nur in Modellen der Theorie nachsehen. Dies ist die Bedeutung der Generalisierung mit Hintergrundwissen. Plotkin (1971) führte die Subsumtion relativ zu einer gegebenen Theorie T folgendermaßen ein:

Eine Klausel C1 ist genereller als eine andere, C2, bezüglich einer Theorie T, genau dann wenn

T, C1 |-- C2, wobei in der Ableitung C1 höchstens einmal vorkommt.

[32] Praktischerweise nimmt man eine Herbrand-Interpretation, die für C1, C2, T konstruiert ist.

Sein Beispiel ist:
C2: flauschig(X) & katze(X) --> kuscheltier(X).
T: katze(X) --> haustier(X).
haustier(X) & flauschig (X) & klein(X) --> kuscheltier(X)

Dann ist eine generellere Klausel zum Beispiel:
C1: katze(X) --> klein(X)

Wir können in C2 katze(X) durch seine Konsequenz, haustier(X), ergänzen. Dann ist C1 die fehlende Klausel, um aus der Theorie C2 abzuleiten. Daß alle Katzen klein sind, wird nur einmal verwendet. Ärgerlicherweise hat die Generalisierung nur indirekt mit dem zu tun, wovon die generalisierte Klausel C2 handelt, nämlich kuscheltier(X). Deshalb hat Wray Buntine (1988) eine andere Subsumtion bezüglich Hintergrundwissen eingeführt, die **generalisierte Subsumtion**.

Eine Klausel C1 ist genereller als eine andere, C2, bezüglich einer Theorie T, genau dann wenn es

eine Substitution θ gibt, die den Klauselkopf von C1 mit dem von C2 unifiziert, geschrieben $\exists\theta$, so daß $C1_{kopf}\ \theta = C2_{kopf}$,

eine Skolemsubstitution σ, die alle Variablen in C2 durch neue Konstanten ersetzt, und

es gibt einen Klauselkörper von C1 mit den Substitutionen θ und σ, der logisch aus dem skolemisierten Klauselkörper von C2 folgt, geschrieben
$T, C2_{körper}\ \sigma \models \exists\ (C1_{körper}\ \theta\sigma)$

Wir müssen also die generellere Klausel durch Substitutionen erst einschränken, damit sie aus der spezielleren folgt. Durch die Unifikation der Klauselköpfe kann bei dem Beispiel der Kuscheltiere jetzt nicht mehr etwas über die Größe von Katzen als Generalisierung gewonnen werden. Wir können hier generalisieren:

C1: flauschig(X) & haustier (X) --> kuscheltier (X)

Dabei ist θ {}, σ {X/a}. Mit der Schnittregel und der Unifikation {X/a} können wir aus der ersten Klausel der Theorie und dem Körper von C2 gerade $C1_{körper}\ \theta\sigma$ folgern.

5.9.2 Generalisierungsverfahren

Wenn wir die Generalisierungsordnung über Klauseln kennen, dann können wir auch ein Verfahren konstruieren, das zu zwei Klauseln (bezüglich Hintergrundwissen T) eine Generaliserung findet. Dabei wollen wir nicht irgendwie, sondern so speziell wie möglich generalisieren.

5.9.2.1 Least General Generalization (LGG)

Plotkin (1970) führt ein zweistufiges Verfahren ein. Zerst werden Literale generalisiert: zwei Literale werden anti-unifiziert. Es wird also die inverse Operation zum allgemeinsten Unifikator ausgeführt. Für zwei Literale, L1 und L2 wird die Generalisierung Lg gebildet, indem

nach einer Substitution θ gesucht wird, so daß

$Lg\theta = L1$ und $Lg\theta = L2$ und

für alle anderen Literale Lg' mit $Lg'\theta = L1$ und $Lg'\theta = L2$ gibt es eine Substitution ρ mit $Lg'\rho = Lg$.

Der Algorithmus dazu nimmt zwei Literale, $p(s_1, ..., s_n)$ und $p(t_1, ..., t_n)$, mit demselben Prädikatsymbol als Eingabe und generalisiert von links nach rechts die beiden Terme an gleicher Argumentposition: s_1, und t_1, ..., s_n und t_n. Wo immer dasselbe Paar von Termen vorkommt, wird es durch eine neue Variable ersetzt. Dies Vorgehen ist wie das der Unifikation. Die Operation über den Termen ist aber gerade entgegengesetzt:

$LGG(s_i, t_i) = X$, falls s_i, t_i konstante Terme oder Variablen ($\neq X$) sind;

$LGG(f(s_1, ..., s_n), f(t_1, ..., t_n)) = f(LGG(s_1, t_1), ..., LGG(s_n, t_n))$

$LGG(f(s_1, ..., s_n), g(t_1, ..., t_m)) = X$

Beispiel:

L1: unterhalt(ulf, maria, alimente(ulf, 1000))

L2: unterhalt(udo, marion, alimente(udo, 500))

LGG(L1, L2): unterhalt(X, Y, alimente(X, V))

Wo immer ulf und udo übereinanderstehen, werden sie durch X ersetzt, wo immer maria und marion übereinanderstehen, werden sie durch Y ersetzt, wo immer 1000 und 500 übereinanderstehen, werden sie durch V ersetzt.

Die Generalisierung von Klauseln betrifft lediglich solche Paare von Literalen, die dasselbe Prädikatsymbol mit gleicher Stelligkeit und dasselbe Vorzeichen haben. Um nun zwei Klauseln zu generalisieren, müssen wir erst einmal alle Literale der beiden Klauseln miteinander kombinieren. Enthält zum Beispiel die Klausel C1 die Literale L11, L12 und L13, die Klausel C2 die Literale L21, L22 und L23, so erhalten wir zwei Listen, bei denen die zu generalisierenden Paare direkt übereinanderstehen.

C1: [L11, L11, L11, L12, L12, L12, L13, L13, L13]

C2: [L21, L22, L23, L21, L22, L23, L21, L22, L23]

Der Algorithmus zur speziellsten Generalisierung von zwei Klauseln, C1 und C2, besteht aus drei Schritten:

Für alle Paare von Literalen $L1_i \in C1$, $L2_j \in C2$, suche diejenigen mit demselben Prädikatsymbol und gleicher Stelligkeit und gleichem Vorzeichen heraus und bilde den $LGG(L1_i, L2_j)$.

Die Generalisierung von C1 und C2 ist die Vereinigung der generalisierten Literale.

Die Generalisierung wird reduziert, d.h. redundante Literale werden entfernt.

Beispiel:

C1: member(2, [2]) -->member(2, [1,2])

C2: member(c, [b, c]), member(c, [c]) --> member(c, [a, b, c])

Aus Platzgründen wird member im folgenden durch m abgekürzt.

C1:[¬ m(2, [2]), ¬ m(2, [2]), ¬ m(2, [2]), m(2,[1,2]), m(2,[1,2]), m(2,[1,2])]

C2:[¬m(c,[b,c], ¬m(c,[c]), m(c,[a, b,c]),¬m(c,[b,c]),¬m(c, [c]), m(c, [a, b,c])]

Das dritte, vierte und fünfte Paar haben nicht dasselbe Vorzeichen. Es werden also lediglich die folgenden Generalisierungen gebildet:

LGG(¬ member(2, [2]), ¬member(c,[b,c])) = ¬member(A, [C|D])

LGG(¬ member(2, [2]), ¬member(c, [c])) = ¬member(A, [A])

LGG(member(2,[1,2]), member(c, [a, b,c])) = member(A, [B, C|D])

LGG(C1, C2): member(A, [C|D]), member(A, [A]) --> member(A, [B, C|D])

Der LGG von Klauseln kann sehr lang werden, da jedes Paar von Literalen zu einem generalisierten Literal der Ergebnisklausel wird. Im schlimmsten Fall ist das Lernergebnis für zwei Klauseln, C1 mit k Literalen und C2 mit n Literalen, k n lang!

Der Reduktionsschritt probiert bei jedem Literal der Ergebnisklausel C aus, ob es weggelassen werden kann, ohne zu einer Generalisierung zu führen, also ob

$C \geq C-\{L\}$ gilt.

Wenn also C genereller als oder gleich generell wie C-{L} ist, obwohl ja nun C mehr Literale enthält, dann ist L redundant und kann gestrichen werden. Leider ist dieser Schritt NP-schwierig.

5.9.2.2 Generalisierte θ-Subsumtion

Seine Definition der generalisierten Subsumtion operationalisiert Buntine (1988), indem er für jede Klausel C_i der generelleren Klauselmenge zeigt, daß sie zurückgeführt werden kann auf eine Klausel der spezielleren Klauselmenge, indem

- Variable 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 die Theorie teilweise ausgewertet wird , d.h. ein Atom aus C_i wird mit einer Klausel der Theorie resolviert.

Dieses Verfahren ist entscheidbar, wenn die Theorie keine Funktionen enthält. Wir wenden es umgekehrt an, um zu gegebenen spezielleren Klauseln eine Generalisierung bezüglich Hintergrundwissen zu finden.

Das System ITOU (Rouveirol 1992) generalisiert eine Klausel C2, indem es

- Literale aus dem Klauselkörper streicht,
- alle Vorkommen eines Terms in C2 durch dieselbe Variable ersetzt, oder
- einige Vorkommen eines Terms in C2 durch verschiedene Variablen ersetzt.

Tatsächlich ist es schwierig zu entscheiden, welches Literal zu streichen ist und wie die Terme in C2 ersetzt werden sollen.

5.9.3 Induktive Logische Programmierung

Es gibt eine Reihe von Systemen, die die Begriffslernaufgabe in einer eingeschränkten Prädikatenlogik lösen. Dabei werden unterschiedliche Generalisierungsoperatoren, unterschiedliche Beschränkungen der Prädikatenlogik für Beispiele, Hintergrundwissen und Lernergebnis sowie unterschiedliche Heuristiken angewandt. Der Gebrauch von Heuristiken widerspricht allerdings der Grundidee induktiver logischer Programmierung. Man will ja eine Generalisierung finden, die genau die Eigenschaften der Daten wiederspiegelt. Nur wenn zugesichert werden kann, daß das Lernergebnis die speziellste Generalisierung oder die generellste Diskriminierung ist, kann der Anwender das Lernergebnis zur Datenkorrektur verwenden. Wenn das Lernergebnis in einem klar definierten Verhältnis zu den Eingabedaten steht, dann liefert es eine Zusammenfassung der Daten und hilft bei ihrer Inspektion. Ein heuristisch gewonnenes Ergebnis kann keinerlei Zusicherungen machen und ist daher weniger überzeugend. Wir unterscheiden also zwischen Systemen,

- die sichere Ergebnisse liefern, indem sie vollständig einen eingeschränkten Hypothesenraum durchsuchen (Beschränkung der Hypothesensprache) -- Beispiele sind GOLEM (Muggleton, Feng 1992), LINUS(Lavrac, Dzeroski 1994), CLINT (de Raedt 1991), Cillg (Kietz 1996)

  und solchen,

- die heuristisch den Hypothesenraum durchsuchen -- z.B. FOIL (Quinlan 1990).

Die Beschränkungen der Repräsentationsformalismen für vollständige Verfahren waren in den letzten Jahren das primäre Forschungsfeld induktiver logischer Programmierung. Die Beweise sind in Kietz (1996) zu finden.

5.9.4 Lernen als nicht-monotoner Schluß -- Regellernen

Die schwierige Aufgabe des Regellernens wurde zuerst von Nicolas Helft (1989) untersucht. Er ordnet den induktiven Schluß in die nicht-monotonen Schlüsse ein. Monoton ist ein Schluß, wenn durch Hinzufügen von neuen Aussagen keine bisherigen Folgerungen ungültig werden. Monotonie ist also:

$T \models X$ also auch $T \cup N \models X$, wobei T ein Theorie, N eine neue Aussage oder eine Menge neuer Aussagen, X eine Aussage oder eine Menge von Aussagen ist.

Nicht-monoton ist ein Schluß, für den diese Eigenschaft nicht garantiert ist. Durch das Hinzufügen von Lernergebnissen wird zwar nichts falsch, was vorher wahr war. Aber es werden mögliche Modelle des gegebenen Wissens ausgeschlossen. Insofern kann man alle Lernverfahren als nicht-monotone Verfahren ansehen. Dieser Sichtweise ging Helft (1989) nach. Seine Lernaufgabe:

Gegeben:

Wissen eines Sachbereichs mit Beobachtungen D

Ziel:

Eine Generalisierung G, die aus D induziert ist

Wie sieht nun die induktive Ableitung aus? Helft verwendet dafür eine zweistufige Bewertung, die Formeln anhand ihrer Folgerbarkeit aus minimalen Modellen von D beurteilt, und dann Bewertungen von Formeln für alle minimalen Modelle erstellt. Die generellste Generalisierung für D sind dann alle Formeln r,

deren Bewertung 1 ist,

die nicht schon (deduktiv) aus D folgen und

für die es keine generellere Generalisierung gibt.

Ein minimales Modell von D enthält genau die Interpretation aller Aussagen aus D und nicht mehr. Helft ergänzt für alle konstanten Terme aus D negierte Aussagen, wenn es über sie keine positiven Aussagen in D gibt. Dies entspricht einer *closed world assumption*, weil keine weiteren Aussagen als nur die durch D gegebenen in dem Modell gültig sind. Würde Helft sich auf Hornformeln beschränken, so gäbe es überhaupt nur ein minimales Modell. Er nimmt aber g-Klauseln (*groundable clauses*). Das sind Klauseln, bei denen keine weitere Einschränkung gemacht wird, als daß zu jeder Variablen aus einem positiven Literal (also der Konklusion) auch dieselbe Variable in einem negativen Literal (also in der Prämisse) vorkommt und keine Funktionen als Terme auftreten. Man kann also eine Disjunktion in der Konklusion haben, und man braucht keine Funktionen zu berücksichtigen. Außerdem muß eine Klausel injektiv über Grundformeln sein. Das heißt, für jedes Paar von Variablen X, Y einer Klausel gibt es eine Substitution, so daß $X\sigma \neq Y\sigma$, und $X\sigma$, $Y\sigma$ sind Grundinstanzen. Damit werden überflüssige Variablen unterdrückt.

Eine Formel r erhält bezüglich eines Modells M

die Bewertung Val(r, M) = 1,

wenn sie aus dem M logisch folgt (d.h. M ist ein Modell für r)

und es für die Prämisse von r Grundinstanzen in M gibt

und die Prämisse injektiv über dem Modell ist.

Sonst erhält sie die Bewertung Val(r, M) = 0.

Im zweiten Schritt erhält eine Formel r bezüglich aller minimalen Modelle

den Wert Val(r, D) = 1, wenn sie für alle minimalen Modelle den Wert 1 hatte,

den Wert Val(r, D) = 0, wenn sie für alle minimalen Modelle den Wert 0 hatte,

den Wert Val(r, D) = 0.5, wenn sie für mindestens ein minimales Modell, aber nicht für alle, den Wert 1 hatte.

Damit läßt sich dann die Generalisierung G für das Sachbereichswissen D so angeben:

G(D) = { r | Val(r,D)=1 & ¬ (D |= r) &

wenn r' ∈ G(D) & r' |= r dann r |= r' }

Alle solche Formeln r sind Generalisierungen von D, die in den Modellen gültig sind, aber nicht schon logisch folgern. Daß sie auch die allgemeinsten Generalisierungen sind, legt er durch die dritte Bedingung fest: jede andere Generalisierung r' kann nur äquivalent mit r sein, wenn es eine Folgerungsbeziehung zwischen r' und r gibt. Helft induziert also generellste Formeln (MGDs) und nicht speziellste. Daß sie dennoch nicht überallgemein sind, erreicht er durch die *closed world assumption*.

Ein Beispiel soll dies verdeutlichen. Nehmen wir als Sachbereichswissen D:

fliegt(tweety),

vogel(tweety),

vogel(polly),

∀ X | vogel(X) --> federn(X)

Dann ist das minimale Modell mit der *closed world assumption*:

fliegt(tweety),

vogel(tweety),

federn(tweety),

vogel(polly),

federn(polly),

¬ fliegt(polly)

Für die folgenden beiden Formeln gelten die für G(D) angegebenen Bedingungen, d.h. sie sind gültig in dem Modell, werden aber nicht schon logisch gefolgert und sind maximal generell.

G(D):

∀ X | fliegt(X) --> vogel(X)

∀ X | fliegt (X) --> federn(X)

Das sind recht genau diejenigen Formeln, die der Intuition entsprechen. Insbesondere wurden Fehlschlüsse, wie etwa "alles, was Federn hat, fliegt", vermie-

den. Dadurch, daß über die Semantik, die den gegebenen Aussagen des Sachbereichs zugrunde liegen (eben das Modell), die Generalisierungen gefunden wurden, erreicht dieses Verfahren meist einleuchtende induzierte Formeln.

Leider wurde dieses Verfahren, obwohl laut Helft von ihm in Prolog implementiert, bisher nicht eingesetzt. Es gibt aber eine Reihe von Regellernverfahren der induktiven logischen Programmierung, deren erstes RDT (Kietz, Wrobel 1992) ist. Dies Verfahren konnte praktisch sowohl in der Robotik (Klingspor, Morik, Rieger 1996) als auch in der Wissensentdeckung in Datenbanken (Brockhausen, Morik 1997) eingesetzt werden. Durch eine vom Benutzer anzugebende syntakische Einschränkung der Hypothesensprache wird eine -- bis auf sicheres *pruning* -- vollständige top-down Suche im Raum aller Regeln der Hypothesensprache ermöglicht.

5.10 Theorie des Lernbaren

Die Theorie des maschinellen Lernens kann in drei Teile eingeteilt werden:

- induktive logische Programmierung
- Lernen im Grenzwert
- wahrscheinlich annähernd korrektes Lernen.

Zum ersten Teil wurde im Abschnitt 5.9.3 etwas gesagt. Für eine Darstellung fehlt hier der Raum. Die beiden anderen Bereiche werden oft zusammengefaßt unter verschiedenen Titeln wie *computational learning theory* oder **algorithmisches Lernen**. Wie implizit schon Plotkin, behandelt man in diesem Bereich des maschinellen Lernens die Frage: was ist überhaupt lernbar und unter welchen Umständen? (Deshalb die Überschrift dieses Abschnittes.) Auch dieser Bereich ist zu umfangreich, um ihn hier darzustellen. Stattdessen werden seine Fragestellungen und schlaglichtartig einige Ergebnisse vorgestellt.

5.10.1 Identifikation im Grenzwert

Die Arbeiten zur **Identifikation im Grenzwert** (identification in the limit) legen folgende Vorstellung zugrunde. Es geht beim Lernen um das Ermitteln einer Theorie oder Funktion oder Sprache anhand einer Folge von Eingaben (wahre und falsche Fakten, Werte aus dem Definitionsbereich und zugehöriger Wert aus dem Wertebereich einer Funktion, Wörter einer Sprache). Nach jeder solchen Eingabe gibt das lernende System ein Lernergebnis aus. Dieser Prozeß geht ewig so weiter. Ein Lernergebnis **erklärt** ein Modell einer Theorie oder eine Funktion oder eine Sprache, wenn das lernende System nach diesem Lernergebnis auf alle folgenden Eingaben nur noch mit syntaktischen Varianten des Lernergebnisses reagiert. In gewisser Weise entspricht dieser Begriff der Erklärung dem der Beschreibungsadäquatheit in der Linguistik. Das Lernergebnis ist sozusagen beschreibungsadäquat, weil es auch neue Eingaben richtig beantwortet. Auch zu der Beobachtungsadäquatheit in der Linguistik gibt es eine Entsprechung in der Lerntheorie: Ein Lernergebnis **beschreibt** ein Modell oder eine Funktion oder eine Sprache, wenn alle folgenden Reaktionen des Systems ebenfalls richtig sind. Hier muß dem Lernergebnis nicht die richtige Theorie, die richtige Funktionsdefinition oder die richtige Grammatik zugrunde liegen, aber es muß zur jeweils richtigen Reaktion auf eine Eingabe führen. Insofern kann das Lernergebnis

dann - in einer Analogie - beobachtungsadäquat im linguistischen Sinne genannt werden.

Der einfachste Lernalgorithmus ist der Aufzählungsalgorithmus, der alle Theorien, Funktionen, Sprachen aufzählt. Er rät einfach ein Ergebnis und, wenn sich dieses Ergebnis bei der nächsten Eingabe als falsch herausstellt, nimmt er das nächste. Nehmen wir diesen einfachsten Algorithmus als Grundlage, wir können uns aber auch jeden anderen denken. Dann **identifiziert** der Algorithmus das richtige Ergebnis (die Theorie, die Funktion, die Sprache) **im Grenzwert**, wenn, nachdem einmal (im Grenzwert) das richtige Ergebnis gefunden wurde, nie wieder ein anderes gewählt wird. Die Bedingung fordert, daß irgendwann das Richtige gefunden wird. Dies ist dann der Grenzwert. Ab diesem Zeitpunkt verändert sich das Lernergebis nicht mehr. Die Bedingung sagt nicht, daß das Lernverfahren oder irgendjemand sonst bemerkt, daß jetzt das Richtige gefunden ist. Der Grenzwert ist also unbekannt. Einen guten Überblick zu diesem Scenario und den darin erforschten Bereichen geben Angluin und Smith in der "Encyclopedia of Artificial Intelligence" oder auch Angluin, Smith (1983).

Ein Beispiel soll deutlich machen, warum es gar nicht möglich ist, zu wissen, wann das richtige Ergebnis erreicht wurde. Man könnte ja meinen, daß, wenn das Ergebnis eines Algorithmus', von dem nachgewiesen wurde, daß er im Grenzwert identifiziert, sich längere Zeit nicht verändert, dieses dann wohl das richtige ist. Das Beispiel zeigt, daß es für jede "längere Zeit" eine noch längere gibt, in der sich das Ergebis als falsch herausstellen kann. Das Beispiel handelt vom Identifizieren einer Funktion. Der Definitions- und der Wertebereich sind die natürlichen Zahlen. Das, was aufgezählt wird, sind Funktionen, hier speziell: alle Polynome mit nur einer Variablen. Das System gibt als Lernergebnis ein Polynom zur Berechnung der Funktion aus. Die Beispiele für die Funktion p werden dem System in Form von Paaren (n, p(n)) in aufsteigender Reihenfolge von n ($n \in |N$) eingegeben.

Beispiel: (0,1) Hypothese des Lernalgorithmus: 1

Beispiel: (1,1) Hypothese des Lernalgorithmus: 1

Beispiel: (2,1) Hypothese des Lernalgorithmus: 1

Beispiel: (3,1) Hypothese des Lernalgorithmus: 1

Beispiel: (4,1) Hypothese des Lernalgorithmus: 1

Nun könnte man allmählich meinen, die Wahrheit identifiziert zu haben: es handelt sich um das konstante Polynom 1! Nähmen wir also an, daß nach 5maliger Wiederholung des Ergebnisses das richtige gefunden ist. Aber dann kommt als nächstes:

Beispiel: (5, 121)

Es könnte sich etwa um die folgende Funktion handeln:

$1 + x\,(x-1)\,(x-2)\,(x-3)\,(x-4)$

Bei dieser Funktion ist für $x \in \{0,1,2,3,4\}$ jeweils ein Faktor gleich 0. Für jede beliebige Schwelle können wir so eine Funktion konstruieren, bei der im Schritt nach der Schwelle das so lange konstante Ergebnis nicht mehr gilt. Deshalb kön-

nen wir nicht fordern, daß aufgrund unvollständiger Information (die Eingaben) bestimmt werden kann, wann das richtige Ergebnis identifiziert wurde. Daher also die schwache Einschränkung darauf, daß jedenfalls ab dem Zeitpunkt, zu dem das Richtige gelernt wurde - wann immer das sei - das Richtige nicht durch etwas Falsches ersetzt wird, sondern höchstens durch etwas genauso Richtiges.

Das Traurige ist nur, daß selbst mit dieser Einschränkung keine induktive Methode gefunden werden konnte, die alle vollständig berechenbaren Funktionen beschreibt oder gar erklärt. Und, ebenso niederschmetternd, es gibt auch keine induktive Methode, die reguläre Sprachen lernt - obwohl doch Kinder sogar die natürlichen Sprache ihrer Umgebung lernen!

Was man tun kann, ist

- die Anforderungen noch weiter abschwächen
- das Scenario dahingehend ändern, daß mehr Informationen in das System eingegeben werden.

Und außerdem gibt es oft Spezialverfahren.

5.10.2 Wahrscheinlich annähernd korrektes Lernen

Wahrscheinlich annähernd korrektes Lernen (probably approximately correct learning - **PAC-learning**) stellt wie das Lernen im Grenzwert ein theoretisches Scenario dar, in dem Eigenschaften von Lernverfahren untersucht werden können. Wie schon im vorigen Abschnitt, so ist auch hier die Motivation, so wenig wie möglich von einem Lernverfahren zu fordern und doch noch etwas darüber aussagen zu können. Die gemeinsame Überlegung hinter diesen beiden Paradigmen ist: es ist völlig aussichtslos, ein korrektes und vollständiges Lernverfahren zu fordern, das nach einer bestimmten Menge von Eingaben sicher und prompt das richtige Ergebnis abliefert und dann anhält. Der Unterschied besteht in den vom jeweiligen Paradigma gewählten Abstrichen. Bei der Identifikation im Grenzwert verzichtet man darauf, daß das Verfahren bei der richtigen Lösung anhält. Beim PAC-learning schwächt man die Anforderung an die Korrektheit des Lernergebnisses ab. Das Lernergebnis ist nur noch mit einer bestimmten Wahrscheinlichkeit von $1-\delta$ 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 polynomial beschränkter Rechenzeit zum Ergebnis kommen und zwar nachdem das Verfahren lediglich Beispiele und davon eine beschränkte Zahl 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 nur das Scenario des PAC-learning vorgestellt. Eine kurze, übersichtliche Einführung bietet Hoffmann (1991), eine ausführliche Behandlung des Bereiches bietet Kearns (1990).

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 Wahrschein-

lichkeitsverteilung der Beispiele. Der Lernalgorithmus erhält außerdem die Parameter δ und ε, $\delta<1$, $\varepsilon < 1$.

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-, 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. Die Begriffsdefinition entstammt der Repräsentationsklasse H.

Eine Repräsentationsklasse ist **lernbar** durch H, wenn es einen Algorithmus $A(\delta,\varepsilon)$ gibt, der bei einer festen aber beliebigen Wahrscheinlichkeitsverteilung und festen, aber beliebigen ε und δ, $\delta<1$, eine Hypothese $h \in H$ ausgibt, die mit einer Wahrscheinlichkeit größer als $1-\delta$ annähernd korrekt ist, und dann anhält.

Eine Repräsentationsklasse ist **polynominal lernbar** aus Beispielen durch H, wenn

- Beispiele aus C und H in polynominaler Zeit klassifiziert werden können und
- der Lernalgorithmus in einer Anzahl von Schritten zum Ergebnis kommt, die sich als Polynom über $1/\varepsilon$, $1/\delta$ und $|c|$ bestimmen läßt.
- Die Repräsentationsgröße $|c|$ ist zum Beispiel die Länge einer Repräsentation.

In diesem Scenario kann man nun für bekannte Sprachklassen bzw. ihre Automaten die prinzipielle Lernbarkeit von Begriffen, die in dieser Sprache ausgedrückt sind, untersuchen. Man kann auch die Anzahl der Beispiele errechnen, die man dem Algorithmus geben muß, damit er lernen kann.

Ein Begriff aus der statistischen Lerntheorie ist innerhalb des PAC-Lernens wieder aufgegriffen worden. Die **Chervonenkis-Dimension** soll die Ausdrucksstärke einer Repräsentationsklasse angeben. 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. Die Vapnik-Chervonenkis-Dimension von H, VCdim(H), ist die Anzahl der Elemente von der größten Menge S, wobei S von H zerschmettert wird. Sie gibt also an, wieviele Unterschiede H machen kann.

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

Wenn es kein Maximum der Kardinalität von S gibt, ist VCdim unendlich. Wenn der Hypothesenraum H endlich ist, so $VCdim(H) \leq \log_2(|H|)$. Um eine Menge der Größe m zu zerschmettern, sind 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 $\log2(|H|)$ (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. $t \leq w_1x_i. + w_2x_i.$

Wenn wir eine 3-elementige Teilmenge S von X haben, wobei für keinen der drei Punkte $x_i = x_j$ oder $y_i = y_j$ gilt, 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 nagativen Beispielen vorstellen. VCdim(H) ist also mindestens schon einmal 3. Aber könnte H nicht auch eine 4-elementige Teilmenge von X zerschmettern? Das Bild A 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.

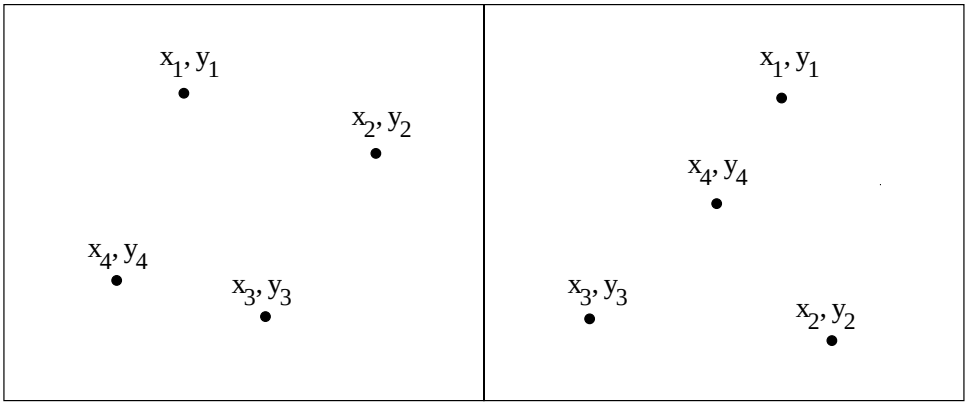


Abbildung A: 4-elementige Teilmengen von X

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

- C ist PAC-lernbar gdw. VCdim(C) endlich ist (Blumer et al., 1990)

- Wenn C endlich ist, so ist VCdim(C) $\leq \log_2(|C|)$ und damit endlich (Blumer et al., 1990).

In den letzten Jahren ist eine Fülle von Beweisen zur Lernbarkeit bestimmter Repräsentationsklassen erarbeitet worden. So sind z.B. Definitonen, die aus gewichteten Attributwerten bestehen, polynomiell lernbar, wenn die Gewichte nicht nur auf 0 oder 1 beschränkt sind, sondern überall zwischen 0 und 1 liegen. Sind die Gewichte auf 0 oder 1 beschränkt, so sind derartige Definitionen nicht mehr polynomiell lernbar. Das ist deshalb interessant, weil neuronale Netze, die sich gut in das PAC-Paradigma einfügen, gerade durch Gewichtsverschiebungen lernen. Sie bearbeiten also ein polynomiell lösbares Problem.

5.11 Literatur

Adriaans, Pieter, Janssen, Sylvia, Nomden, Erik (1993): Effective identification of semantic categories in curriculum texts by means of cluster analysis, in: *ECML-93 European Conference on Machine Learning*, Vienna, Austria 1993

Agrawal, Rakesh, Mannila, Heikki, Srikant, Ramakrishnan, Toivonen, Hannu, Verkamo, A. Inkeri (1996): Fast Discovery of Association Rules, in: Piatetsky-Shapiro, Smyth, Uthurusamy (eds): *Advances in Knowledge Discovery and Data Mining*, Cambridge: The MIT Press 1996

*Angluin, Dana, Smith, Carl (1983): Inductive Inference - Theory and Methods, in: *Computing Surveys*, Vol. 15, No. 3, 1983

Balabanovic, M., 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 Enviroments, AAAI-Press, 1995

Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M. (1990): Learnability and the Vapnik-Chervonenkis dimension, in: *Journal of the ACM*, Vol. 36, No. 4, 929-965, 1990.

Brockhausen, P., Morik, K. (1997): A Multistrategy Approach to Relational Knowledge Discovery in Databases, in: *Machine Learning Journal*, demnächst

Buntine, Wray (1988): Generalized Subsumption and Its Applications to Induction and Redundancy, in: *Artificial Intelligence*, 36, 1988

Buntine, Wray, Muggleton, Stephen (1988): Machine Invention of First-Order Predicates by Inverting the Resolution, in: Procs. of IWML-88 at Ann Arbor, MorganKaufmann, 1988

DeJong, Gerald, Mooney, Raymond (1986): Explanation-Based Learning - An Alternative View, in: *Machine Learning Journal*, Vol. 1, No.2, 1986

de Raedt, Luc (1991): *Interactive Concept-Learning*, Ph.D Thesis, Katholieke Universiteit Leuven, 1991

Dillmann, Rüdiger (1988): *Lernende Roboter - Aspekte maschinellen Lernens*, Berlin: Springer, 1988

Emde, Werner, Habel, Christopher, Rollinger, Claus-Rainer (1983): The Discovery of the Equator or Concept Driven Learning, in: Procs. of IJCAI-83 at Karlsruhe, 1983

Fisher, D. (1987): Knowledge Acquisition Via Incremental Conceptual Clustering, in: *Machine Learning Journal*, Vol. 2, No.2, 1987

Helft, Nicolas (1989): Induction as Nonmonotonic Inference, in: Procs. Knowledge Representation, 1989

Hoffmann, Achim (1991): Die Theorie des Lernbaren - ein Überblick, in: *KI* 1, Themenheft maschinelles Lernen, 1991

Joachims, Thorsten, Freitag, Dayne, Mitchell, Tom (1997): WebWatcher: A Tour Guide for the World Wide Web, to appear in: Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997

Kaelbling, L.P. (1991): Foundations of Learning in autonomous agents, in: *Robotics and Autonomous Systems*, Vol. 8, 131-144

Kearns, Michael J. (90): *The Computational Complexity of Machine Learning*, Cambridge,MA: MIT Press, 1990

Kietz, J.-U., Morik, K. (1994): A Polynomial Approach to the Constructive Induction of Structural Knowledge, *Machine Learning Journal*, Vol. 14, No. 11, 217 - 235

*Kietz, Jörg-Uwe (1996): Induktive Analyse relationaler Daten, Doktorarbeit an der TU Berlin (am LS8 erhältlich)

Kietz, Jörg-Uwe, Wrobel, Stefan (1992): Controlling the Complexity of Learning in Logic through Syntactic and Taskoriented Models, in: Muggleton, Stephen (ed.): *Inductive Logic Programming*, London: Academic Press, 1992

Klingspor, V., Morik, K., Rieger, A. (1996): Learning Concepts from Sensor Data of a Mobile Robot, in: *Machine Learning Journal*, Vol. 23, No.2/3, 305 -- 332

Kodratoff, Y., Ganascia, J.-G. (1986): Improving the Generalization Step in Learning, in: Michalski, R. S., Carbonell, J. G., Mitchell, T. M. (eds.): Machine Learning - An Artificial Intelligence Approach, Palo Alto: Morgan Kaufmann, Vol. 2, 1986.

Lang, K. (1995): NewsWeeder: Learning to Filter Netnews, in: International Conference on Machine Learning (ICML), 1995

Lavrac, Nada, Dzeroski, Saso (1994): Inductive Logic Programming - Techniques and Applications, New York: Ellis Hardwood, 1994

Lebowitz, Michael (1987): Experiments with Incremental Concept Formation - UNIMEM, in: *Machine Learning Journal*, Vol. 2, No.2, 1987.

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

Michalski, Ryszard (1986): Understanding the Nature of Learning - Issues and Research Directions, in: Michalski, Carbonell, Mitchell, T. (eds): *Machine Learning - An Artificial Intelligence Approach*, Vol.II, Morgan Kaufman, 1986

Michalski, Ryszard, Stepp, Robert (1986): CLUSTER/S, in: Michalski, R.S., Carbonell, J.G., Mitchell, T. (eds): *Machine Learning - An Artificial Intelligence Approach*, Vol.II, Morgan Kaufman, 1986

Michalski, Ryszard, Stepp, Robert (1983): Learning from Observation: Conceptual Clustering, in: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.): *Machine Learning*, Palo Alto: Tioga,1983

Michie, Donald (1989): New Commercial Opportunities Using Information Technology, in: Brauer, Freksa (eds): *Wissensbasierte Systeme*, Berlin: Springer, 1989

Mitchell, Tom (1982): Generalization as search, in: *Artificial Intelligence*, Vol. 18, No. 2, 1982

Mitchell, Tom (1985): LEAP: A Learning Apprentice for VLSI Design, in: *Proceedings of the International Joint Conference on Artificial Intelligence 1985*, Los Angeles: Morgan Kaufman, 1985

Mitchell, Tom, Keller, Richard, Kedar-Cabelli, Smadar (1986): Explanation-Based Generalization - A Unifying View, in:*Machine Learning Journal*, Vol. 1, No.1, 1987

**Mitchell, Tom (1997):*Machine Learning*, New York: McGraw Hill, 1997

Morik, Katharina, Wrobel, Stefan, Kietz, Jörg-Uwe, Emde, Werner (1993): Knowledge Acquisition and Machine Learning: Theory, Methods and Applications, San Diego: Academic Press, 1993.

Morik, Katharina (1987): Acquiring Domain Models, in: *Int. Journal of Man-Machine Studies*, 26, 1987

Morik, K., Potamias, G., Moustakis, V., Charisis, G. (1994): Knowledgable Learning Using MOBAL, in: *Applied Artificial Intelligence*, Vol. 8, No. 4, 579 - 592

Muggleton, S., Feng, C. (1992): Efficient Induction of Logic Programs, in: Muggleton, S. (ed.): *Inductive Logic Programming*, Academic Press, 281-298

Pazzani, M., Muramatsu, J., Billsus, D. (1996): Syskill & Webert: Identifying interesting web sites, in: AAAI Conference, Portland, 1996

Pitt, Leonard (1996): CLASSIC Learning, in: Machine Learning Journal, 25, 151, 1996

*Plotkin, Gordon (1970): A Note on Inductive Generalization, in: Meltzer, Michie (eds): *Machine Intelligence 5*, Edinburgh:Edinburgh University Press, 1970

Plotkin, Gordon (1971): A Further Note on Inductive Generalization, in: Meltzer, Michie (eds): *Machine Intelligence 6,* Edinburgh:Edinburgh University Press, 1970

Quine, Willard Van Orman (1977): Natural Kinds, in: Schwartz (ed): *Naming, Necessity, and Natural Kinds,* Cornell University Press, 1977

Quinlan, J.R. (1986): Induction of Decision Trees, in: *Machine Learning Journal*, Vol. 1, No. 1, 81-106

Quinlan, R.J. (1990): Learning Logical Definitions from Relations, in: *Machine Learning Journal*, Vol. 5, No. 3, 239-266

Quinlan, J.R. (1993): C4.5 Programs for Machine Learning, San Mateo: Morgan Kaufman, 1993

Reimer, Ullrich, Pohl, Klaus(1991): Automatische Wissensakquisition aus Texten, in: *KI* 1, Themenheft maschinelles Lernen, 1991

Rieger, Anke (1993): Neuronale Netzwerke, Dortmund: Universität Dortmund, Lehrstuhl Informatik VIII

Rouveirol, Celine (1992): Extensions of Inversion of Resolution Applied to Theory Completion, in: Muggleton, Stephen (ed): *Inductive Logic Programming*, Academic Press, 1992

Saitta, L., Giordana, A. (1990): Abstraction- a general framework for learning, in: Procs. AAAI - Workshop on Automatic Generation of Approximations and Abstractions

Scott, Paul D. (1983): Learning - The Construction of A Posteriori Knowledge Structures, in: Procs. AAAI-83, Washington, 1983

Simon, Herbert (1983): Why should Machines Learn?, in: Michalski, R.S., Carbonell, J.G., Mitchell, T. (eds): *Machine Learning - An Artificial Intelligence Approach*, Vol. I; Tioga Press, 1983, Springer, 1984

Segre, Alberto (1988): *Machine Learning of Robot Assembly Plans,* Boston: Kluwer, 1988

Sommer, E., Morik, K., Andre, J.-M., Uszinsky, M. (1994): What Machine Learning Can Do for Knowledge Acquisition -- A Case Study, in: *Knowledge Acquisition Journal*, Vol. 6, 435 - 460

Turing, Alan (1950): Computing Machinery and Intelligence, in:*Mind* 59, auf deutsch nachgedruckt in:Dotzler, Kittler (eds) (1987):*Turing, A. - Intelligence Service,* Berlin: Brinkmann und Bose Vg, 1987

Vapnik, Vladimir N. (1995): *The Nature of Statistical Learning Theory,* New York: Springer, 1995

Wrobel, Stefan (1994): *Concept Formation and Knowledge Revision,* Dordrecht: Kluwer

Wrobel, Stefan (1991): Towards a Model of Grounded Concept Formation, in: Procs. 12th IJCAI, Los Altos: Morgan Kaufmann, 1991

Zercher, Kai (1991): Wissensintensives Lernen von Regeln zur Fehlerdiagnose von Robotermontage, in: *KI* 1, Themenheft maschinelles Lernen, 1991