Einführung in die Künstliche Intelligenz

G. Görz *et al.* (Hrsg.)

21. April 1995

Inhaltsverzeichnis

1 Wissensrepräsentation und Logik **1**

1.1 Wissensrepräsentation und Logik – Eine Einführung 1

1.1.1 Einleitung 1

1.1.2 Modellierung 2

1.1.3 Die Entwicklung eines Modells 11

1.1.4 Repräsentationsformalismen, Inferenzalgorithmen, und Berechenbarkeitseigenschaften 23

1.1.5 Systeme 41

Index 57

Kapitel 1

Wissensrepräsentation und Logik

Kapitelherausgeber: Bernd Owsnicki-Klewe

1.1 Wissensrepräsentation und Logik – Eine Einführung

Bernd Owsnicki-Klewe, Kai v. Luck und Bernhard Nebel

1.1.1 Einleitung

Der Begriff „Wissensrepräsentation“ hat viele schillernde Bedeutungen. So wird er sowohl benutzt, um den Entwurf und die Implementation von Formalismen zu beschreiben, und um die Modellierung – bzw. Rekonstruktion [Schefe 85] – eines Teils der Realität, der Domäne, zu bezeichnen, als auch, um Vorüberlegungen zu einer dieser beiden Teildisziplinen zu umschreiben.[1]

In diesem Beitrag soll eine Einführung in die aktuelle Diskussion über „Wissensrepräsentation“ gegeben werden. Als erstes werden einige Probleme bei der logischen Rekonstruktion von Weltausschnitten dargestellt, die unter dem Blickwinkel der Modellierung von Alltagstheorien analysiert werden.[2] Im folgenden Abschnitt wird exemplifiziert, wie Notationen

[1] Die Programmierung von Anwendungsprogrammen mit Programmiersprachen wie PROLOG, LISP, KEE, ART etc., die häufig auch unter dem Begriff „Wissensrepräsentation“ subsumiert wird, soll hier nicht weiter betrachtet werden.

[2] Die Rekonstruktion von Fachexpertise wird in dem Beitrag über Expertensysteme diskutiert.

semantisch fundiert und die durch die Semantik induzierten Ableitungsbegriffe operationalisiert werden können. Insbesondere stellt sich in diesem Zusammenhang die Frage nach den prinzipiellen Kosten solcher Operationalisierungen, der Entscheidbarkeit der induzierten Ableitungsbegriffe sowie deren Komplexitätsklasse. Eine Betrachtung der Probleme, die bei gegebenen Modellen in gegebener Notation noch bestehen, um eine Wissensbasis als System auf einem Rechner realisieren zu können, beschließt diesen Beitrag.

Dem Beitrag zu Grunde liegt dabei das Bild einer Wissensbasis als

und kann daher nur insgesamt beurteilt werden nach der Güte der Services, die sie leistet. In diesem Bild wird eine Wissensbasis somit nicht als eine passive Sammlung von Ausdrücken einer Wissensrepräsentationssprache gesehen, sondern als ein mehr aktives Objekt, das die durch den Formalismus angegebene Theorie über den Ausdrücken zur Verfügung stellt, d.h. auch die aus den gegebenen Ausdrücken ableitbaren Ausdrücke (s.u.). Der Interpreter operationalisiert dann diesen Ableitungsbegriff.

1.1.2 Modellierung

In diesem Abschnitt betrachten wir das Problem der Darstellung von Wissen ohne Rückgriff auf irgendein spezielles operationales Repräsentationsschema. Die Idee ist dabei natürlich, sich soweit wie möglich von den formalen Repräsentationskonstrukten des Schemas unabhängig zu halten. Genauer gesagt bedeutet das, daß wir auf einer Ebene arbeiten, die wir in Ermangelung eines besseren Begriffs als „konzeptionelle Ebene“ bezeichnen wollen.

Arbeit auf der konzeptionellen Ebene ist also im wesentlichen unabhängig von einem aktuell vorhandenen Formalismus oder gar System; sie konstituiert einen vorgeschalteten intellektuellen Prozeß, der Klarheit über die Gegebenheiten des darzustellenden Weltausschnitts bringen soll und ferner die intendierte Funktionalität in die resultierende Konstruktion einbindet.

Wenn wir uns also in diesem Abschnitt der Rekonstruktion von Weltausschnitten zuwenden, läßt uns der unklare Begriff „Weltausschnitt“ schon wieder stutzen. Häufig wird dieser Begriff benutzt, um sich zu entschuldigen, daß nur ein Teil der Realität betrachtet wird, weil ...

- ... „es den Rahmen dieses Projektes sprengen würde, mehr als nur ...“
- ... „es für die Aufgaben des Systems ausreicht, nur ... zu betrachten“

- ... „weil es sich hier nur um eine Durchführungsstudie handelt“
- ... kurzum – weil wir zu faul sind.

Diese Entschuldigungen gehen davon aus, daß, wenn erst mal – pars pro toto – die relevanten Strukturen als Kategorien festgelegt sind, es sich nur noch um eine Fleißarbeit handeln kann, den Weltausschnitt beliebig zu erweitern. Der unter diesen Prämissen logische Aufruf zu solchen Projekten ist durch Pat Hayes in [Hayes 79] konkretisiert worden und damit schon über zehn Jahre her.[3] Auf der Basis solcher Überlegungen wurde schon etliches Geld in Fleißarbeitsprojekte gesteckt, deren prominentestes Beispiel das Projekt CYC darstellt (s. z.B. [Lenat, Guha 90]).

Die hinter solchen Einschätzungen Stehenden teilen sich des weiteren auf in eine eher reduktionistische Variante und eine eher autistisch orientierte Vorgehensweise. In der reduktionistischen Variante, deren umstrittenster Vertreter R. Schank ist, wird von der Existenz eines – (sehr) endlichen – Satzes von *Primitiven* ausgegangen, die als Bausteine einer mit wenigen Konstrukttypen aufzubauenden semantischen Bedeutung dienen und von daher auch als *semantische Primitive* bezeichnet werden.[4] Die innerhalb der KI von R. Quillian bekanntgemachte autistische Variante geht dagegen eher von einem abgeschlossenen Netz von Beziehungen zwischen Begrifflichkeiten aus, so daß sich die Bedeutung einer Begrifflichkeit (eines Konzeptes, Worts ... je nach Theorieausprägung bei verschiedenen Autoren) *ausschließlich* über seine Beziehungen zu den anderen Begrifflichkeiten ergibt.[5] Ein direktes Aufeinandertreffen von Vertretern beider Richtungen ist z.B. in [Lehnert, Wilks 79] versus [Bobrow, Winograd 79] nachzulesen.

Der Übergang von exemplarischen Modellierungen von extrem begrenzten μWelten zur Modellierung relevanter Ausschnitte der realen Welt scheint, wie schon unter anderem in [Dreyfus 81] argumentiert, nicht ein quantitativer, sondern ein qualitativer Schritt. Dabei ist die Loslösung des kontextuellen Gebrauchs von Bedeutung ein Manko, das z.B. in [Winograd, Flores 86] kritisiert wird. Doch wie bekommt die Semantik wieder einen kontextuellen Sinn? Schon frühzeitig haben Miller et al. in [Miller et al. 60] auf eine *interessenbezogene* Festlegung von Ebenen der Modellierung hingewiesen, die uns etwas aus dem beschriebenen Dilemma herausführt. Diese *interessenbezogene Festlegung von Grundprädikaten*, die *molekulare Schicht* von semantischen Primitiven, ist abhängig von den inferenziellen Aufgaben eines Modells und steht somit im Gegensatz zu den Versuchen, eine aufgaben- bzw. interessenungebundene Modellierung im allgemeinen angeben zu können. Insbesondere aber

[3] Ein erster Aufruf wurde schon 1959 von John McCarthy in [McCarthy 59] gemacht und in [McCarthy 68] erweitert.

[4] Für eine exemplarische Darstellung einer solchen Theorie s. z.B. [Schank 73].

[5] Ein Exemplar dieser Variante ist z.B. in [Quillian 66] beschrieben.

steht dieser Ansatz in seiner letzten, von den Autoren in [Miller et al. 60] nicht gesehenen Konsequenz gegen den rein inkrementellen Ansatz der Weltrekonstruktion.

Dieses scheint aber nur auf den ersten Blick ein trauriges Resultat zu sein. Beim zweiten Hinsehen ist die Festlegung von Interessen bei der Modellierung in den Dimensionen *relevanter Ausschnitt*, *Tiefe* und *Komplexität der inferenziellen Kompetenz* ein Prozeß, der strukturelle Ähnlichkeiten bestimmter Interessen und damit strukturell ähnliche Modelle für bestimmte Klassen von Interessen ermöglichen könnte. Ein sich hieraus ergebendes Problem ist die Untersuchung der Interaktion unterschiedlicher oder sich sogar einander widersprechender Modelle für eine Problemlösung. Spätestens hier ist jedoch die explizite Repräsentation der Aufgaben, für die das jeweilige Modell entworfen wurde, eine notwendige Voraussetzung, um über diese Strukturen inferieren zu können. In diesem Zusammenhang ist nicht zwingend von hybriden Formalismen zu sprechen, wie sie z.B. in [Sloman 85] gefordert werden. Vielmehr ist dieses erstmal eine Frage der Modellbildung und damit eine Frage nach „hybriden Modellen".

1.1.2.1 Naive Theorien

In diesem Abschnitt soll nicht versucht werden, den wirklich großen Bereich der Forschungen über naive Theorien darzustellen, insbesondere weil mit [Davis 90] inzwischen eine gute Einführung existiert. Vielmehr soll auf ein paar Eigenschaften naiver Modelle eingegangen werden, die für diesen Beitrag wichtig erscheinen.

Der Begriff *naive Theorie* wurde von Pat Hayes in [Hayes 79] geprägt und ist Kern eines Unterfangens, das sich mit der Formalisierung des „Alltäglichen" im Gegensatz zu wissenschaftlichen Teiltheorien über die Welt wie die Physik oder Biologie beschäftigt. Die forschungsleitende Frage ist hier die Frage nach den Theorien, die Menschen im Alltag benutzen, um sich in der Welt zurechtzufinden und in dieser zu agieren. Dabei wird davon ausgegangen, daß Menschen sich tatsächlich in der Welt verhalten können und daß sie dafür eine geeignete Theorie bzw. eine Sammlung geeigneter Theorien über die Welt benötigen, aber auch besitzen. Des weiteren wird in diesem Ansatz vorausgesetzt, daß eine Rekonstruktion solcher Theorien mit Hilfe logischer Systeme möglich und daher sinnvoll ist, weil nur solche Systeme eine elaborierte Semantik haben.[6]

Somit ist Gegenstand naiver Theorien z.B. die Formalisierung des Verhalten von Flüssigkeiten im Raum und über die Zeit, wobei man sich nicht in die Details einer wissenschaftlichen Betrachtung z.B. der Physik verliert, sondern gemäß dem Ansatz verfährt

[6] Eine insbesondere diesen Aspekt kritisch beleuchtende Debatte findet sich in [McDermott 87] und den ebenfalls dort erschienenen Repliken.

Ich kann mir ein Glas Wasser einschenken, ohne dafür Physik studiert zu haben.

Durch Hayes Papier [Hayes 79] inspiriert versuchten in der Folge viele Forscher spezielle Klassen von Phänomenen als mehr oder weniger zusammenhängende Bereiche zu formalisieren.[7] Insbesondere wird in der Elaboration naiver Theorien versucht, bestimmte Phänomene auszufaktorisieren und als separaten Bereich (Cluster) zu behandeln.

> Identifying these clusters is both one of the most important and one of the most difficult methodological tasks in developing a naive physics. (...) The symptom of having got it wrong is that it seems hard to say anything very useful about the concepts one has proposed. [Hayes 79, S. 246]

Der bisher einzige – halbwegs – gelungene Versuch der Ausfaktorisierung innerhalb des Unterfangens der Rekonstruktion naiver Theorien scheint hierbei die separate Behandlung von Zeit und Raum zu sein.[8]

Eine Reintegration dieser Teilformalisierungen in die ursprünglichen Kontexte, aus denen sie ausfaktoriert wurden, ist jedoch noch nicht in dem Maße erfolgt, daß von einer erschöpfenden Behandlung dieser Teile gesprochen werden kann. So wurde z.B. von J. Allen et al. (s. z.B. [Allen 83] und [Allen, Hayes 87]) ein Zeitkalkül entworfen, das auf Intervallbasis axiomatisiert eine mögliche Behandlung der Zeitphänomene verspricht. Bei genauerer Analyse der vorgeschlagenen Axiomatisierungen fällt jedoch auf, daß bei der Ausfaktorierung von wesentlichen Aspekten abstrahiert wurde, wie sie z.B. in Ausdrücken der Art

... täglich außer Montags ...

auftreten.[9]

Ein erster Versuch, einen größeren zusammenhängenden Bereich mit Hilfe einer Logik zu rekonstruieren, wurde von Hayes selbst vorgenommen und in [Hayes 85] dokumentiert. In der Zwischenzeit ist eine Menge ganz unterschiedlicher Arbeiten hinzugekommen, die so verschiedene Bereiche behandeln wie die Elaboration von „Teil-Ganzes"-Beziehungen und deren Eigenschaften (insbesondere der begrenzten Transitivität) wie in [Winston et al. 87] oder die Modellierung kinematischer Systeme solider Körper (s. z.B. [Faltings et al. 89]).[10]

Als Vorgehen bei der Entwicklung naiver Theorien wurde von Pat Hayes vorgeschlagen:

[7] Für einen Überblick über solche Arbeiten siehe z.B. auch [Hobbs, Moore 85] oder [Hobbs et al. 85].

[8] Eine vertiefte Darstellung dieser Cluster findet sich in einem anderen Beitrag.

[9] Für eine Diskussion solcher zeitlicher Aspekte s. z.B. [Kortüm 91].

[10] An dieser Stelle sei nochmals auf [Davis 90] verwiesen, das sich hervorragend als Nachschlagewerk für Hinweise auf Ausarbeitungen naiver Theorien eignet.

> I believe this process of formalisation, confrontation against intuition, and correction, can (...) be used to develop naive physics. Here is a domain in which we are all experts, in the required sense. The performance of a formalisation is, here, the pattern of inferences which it supports. Performance is adequate when the „experts" agree that all and only the immediate, plausible consequences follow from the axioms of the formalisation. (In fact, this is a weak notion of adequacy ...) [Hayes 79, S. 267]

Zur Bewertung der Plausibilität der Konsequenzen einer Menge von Axiomen wiederum sollte auch hier beachtet werden:

> We rarely use a representation in an intentional vacuum, but we always have goals ... [Minsky 91, S. 39]

1.1.2.2 Domänen und Aufgaben

Auf der Basis dieser Erläuterungen lassen sich die beiden Eckpunkte bei der Rekonstruktion einen Weltausschnitts wie folgt fassen:

Domänenabhängigkeit: Es ist ein gewisser vorgegebener Weltausschnitt darzustellen. Das ist an sich eine einsichtige Einschränkung der Aufgabe, da kaum gefordert wird, „alles" oder „irgendetwas" darzustellen. Daraus ergeben sich Konsequenzen, die im einzelnen analysiert werden müssen.

Aufgabenabhängigkeit: Es darf erwartet werden, daß die Berücksichtigung der *geforderten Funktionalität* nicht ohne Auswirkungen auf das Resultat der Rekonstruktion bleiben wird.

Die „Domäne" (domain) stellt also den darzustellenden Weltausschnitt dar, die „Aufgabe" (task) beschreibt die geforderte Funktionalität eines Problemlösers, der auf der Domäne arbeitet. Diese beiden Kategorien sind im Ideallfall unabhängig voneinander, obwohl einige Kombinationen u.U. wenig Sinn in der Realität machen.

Unter welchen Bedingungen auch immer wird am Ende dieses Prozesses eine Rekonstruktion des Weltausschnitts stehen, die wir als „Modell" des betreffenden Weltausschnitts bezeichnen.

1.1.2.3 Notation

Der Prozeß der Modellierung hat als Ergebnis also eine Struktur der Domäne, die u.U. auch die Lösung der gestellten Aufgabe beinhaltet. Sie dient dann als Grundlage der „Operatio-

nalisierung", die mit einem gegebenen Repräsentationssystem vorgenommen wird. Diese Struktur muß in einer geeigneten Notation formuliert werden, aus der heraus die Operationalisierung gewonnen werden kann.

Die Verwendung einer solchen Notation konterkariert natürlich die ursprüngliche Idee, unabhängig von einem Repräsentationsschema zu modellieren. Es ist aber wohl müßig, einer solche Forderung in letzter Konsequenz folgen zu wollen. Man wird sich lediglich nach einer Notation umsehen, die generell genug ist, alle Operationalisierungen zuzulassen.

Von einer geeigneten Notation darf man drei Eigenschaften verlangen:

Kommunikativität: Sie muß auf der syntaktischen und semantischen Ebene eindeutig sein. So kann das entstehende Modell zwischen verschiedenen Personen oder Gruppen diskutiert und manipuliert werden.

Ausdrucksfähigkeit: Sie muß in der Lage sein, die beobachteten oder angenommenen Phänomene der Domäne zu rekonstruieren.

Operationalisierbarkeit: Es muß die Möglichkeit bestehen, das entstehende Modell (effizient) in ein lauffähiges System umzuwandeln, das die vorgegebenen Aufgaben löst.

Ausgangspunkt unserer Darstellung wird dabei das System der *Prädikatenlogik erster Stufe* (PL1) sein, da hier die Anforderungen an *syntaktische und semantische Eindeutigkeit* im wesentlichen erfüllt sind.

Die *Ausdrucksfähigkeit* von Logik als Basis der Formalisierung von Alltagstheorien ist Gegenstand vieler Debatten (s. z.B. [McDermott 87]). Es zeigt sich, daß es Wissenskategorien gibt, die sich einer Modellierung mit Prädikatenlogik entziehen oder wenigstens zu vergleichsweise komplizierten Modellen führen. Das führt dazu, daß Erweiterungen – etwa im Sinne von *Zeitlogiken*, *Handlungslogiken* oder *nicht-monotonen Logiken* – verwendet werden, die einerseits solche Phänomene rekonstruieren und andererseits den anderen Anforderungen an eine konzeptionelle Notation gerecht werden.

Die *Operationalisierbarkeit* dieser Notation ist natürlich von erheblichem praktischen Interesse. Ohne tiefere Analysen darf man feststellen, daß die *Semientscheidbarkeit der Prädikatenlogik erster Stufe* ein theoretisches Problem darstellt, dessen Relevanz für die in konkreten Modellen vorkommenden Axiomenmengen genau untersucht werden muß. Diese Untersuchungen erstrecken sich auf die Punkte der *Entscheidbarkeit* und *Komplexität* (s.u.).

1.1.2.4 Aufgaben

Behandelt man ein gegebenes Modellierungsproblem mit einer formalen Notation, wie etwa der Prädikatenlogik, so hat man ein Werkzeug zur Verfügung, das geeignet ist, die Gegeben-

heiten der Domäne durch *Axiome* darzustellen. Die Semantik der Prädikatenlogik ordnet einer Sammlung von Axiomen gewisse algebraische Strukturen als deren *Modelle* zu. Der Modellbegriff der algebraischen Semantik hat hierbei nichts mit dem Modellbegriff zu tun, wie er in diesem Abschnitt bisher verwendet wurde. Deshalb nennen wir in diesem Beitrag, falls eine Verwechslung möglich ist, Modelle, wenn wir den Modellbegriff der logischen Semantik meinen, Tarski-Modelle.

Die Domäne läßt sich dann prinzipiell recht einfach darstellen, es fehlt auf den ersten Blick eine Modellierung der zu lösenden Aufgabe.

Der Ausweg besteht darin, dem formalen Notationsystem entsprechende „deduktive Fähigkeiten" zuzuordnen, die es erlauben, aus der vorgelegten Axiomatik Schlußfolgerungen abzuleiten. Die Prädikatenlogik ist mit einem Begriff der *Beweisbarkeit* ausgestattet, der als Grundlage der Modellierung von Aufgaben dienen wird.

Das allein aber ist nicht ausreichend, um Aufgaben in dem Sinn zu modellieren, wie er bei konkreten Modellierungsprojekten auftritt. Es geht dabei kaum darum, in genereller Weise Formeln aus der Axiomatisierung zu beweisen oder zu widerlegen; die spezifizierte Aufgabe ist selten von dieser Form.

Man wird also das *dynamische Verhalten* des Modells auf einer anderen Ebene spezifizieren müssen, die es erlaubt, die zu Lösung der gegebenen Aufgabe unter Rückgriff auf dieses Verhalten zu formulieren. Doyle und Patil [Doyle, Patil 91a] sprechen in diesem Zusammenhang von „Dienstleistungen", die ein konkretes Repräsentationssystem der Problemlösungskomponente zur Verfügung stellen muß. Damit kommt man dazu, die gestellte Aufgabe dadurch zu modellieren, daß man die erforderlichen Dienstleistungen des resultierenden Systems auf dem konzeptionellen Modell antizipiert.

1.1.2.5 Exkurs Logik

Um das eben genannte etwas zu explizieren, soll in diesem Abschnitt eine kurze Auffrischung der wichtigsten in diesem Beitrag benützten Begriffe der Logik erfolgen. Somit dient dieser Abschnitt nicht als allgemeine Einführung in Logik und kann von Kundigen übersprungen werden. Desweiteren werden wir uns hier nur auf die Prädikatenlogik erster Stufe (PL1) beschränken, da nur diese in diesem Beitrag benötigt wird.[11]

In diesem Beitrag wird PL1 nicht als Konkurrenz zu anderen Formalismen/Notation zur Repräsentation von Wissen aufgefaßt, die eher z.B. unter „semantische Netze" oder „Frames" einzuordnen sind, sondern als *Referenzsprache*, um für die folgenden Modellierungs-

[11] Dieser Exkurs orientiert sich zu großen Teilen an [Eisinger, Ohlbach 87].

beispiele eine Notation zu haben, die im allgemeinen als bekannt vorausgesetzt werden kann, als auch als Mittel zur Erläuterung anderer hier vorzustellender Formalismen.

Die *Syntax* einer Logik legt die Objekte der Logik fest und definiert damit eine Sprache, in der Aussagen formuliert werden können.[12] Folgende einfachen Objekte sind in PL1 verfügbar:

Konstantensymbole: mit indizierten Bezeichnern wie c_1, s_1, $staubi_1$.

Variablensymbole: mit unindizierten Bezeichner wie x, s, $staubi$.

Funktionssymbole: mit kleingeschriebenen Bezeichnern wie $gehalt\_von$, $gehört\_zu$.

Prädikatssymbole: mit großgeschriebenen Bezeichnern wie $HatAlsTeil$, $Motor$.

Aus diesen primitiven Symbolen werden zusammengesetzte Objekte aufgebaut. Dazu ist jedem Funktions- und Prädikatssymbol eine Stelligkeit zugeordnet, die die Anzahl seiner Argumente angibt.[13] Die Menge der nicht-logischen Objekte, der Terme und Atome, werden wie folgt aufgebaut:

- Konstantensymbole sind Terme
- Variablensymbole sind Terme
- Falls f ein Funktionssymbol der Stelligkeit n ist und $t_1, \ldots, t_n$ Terme, dann ist auch $f(t_1, \ldots, t_n)$ ein Term
- Falls P ein Prädikatssymbol der Stelligkeit n ist und $t_1, \ldots, t_n$ Terme, dann ist $P(t_1, \ldots, t_n)$ ein Atom

Die Menge der logischen Symbole, der Junktoren und Quantoren, ist wie folgt gegeben:

nicht: $\neg$ (Negation)

und: $\wedge$ (Konjunktion)

oder: $\vee$ (Disjunktion)

wenn ... dann: $\Rightarrow$ (Implikation)

genau dann ... wenn: $\Leftrightarrow$ (Äquivalenz)

für alle: $\forall$ (Allquantor)

es gibt: $\exists$ (Existenzquantor)

Formeln gehorchen folgenden Bildungsgesetzen:

[12] Die hier gewählte Syntax ist eine der gebräuchlichsten und wird daher aus didaktischen Gründen in dieser Einführung anderen syntaktischen Varianten vorgezogen.

[13] Die Mengen der primitiven Symbole mit ihren Stelligkeiten bilden eine Signatur.

- Alle Atome sind Formeln
- Falls F und G Formeln sind und x ein Variablensymbol, dann sind auch $\neg F, F \wedge G, F \vee G, F \Rightarrow G, F \Leftrightarrow G, \forall x F$ und $\exists x F$ Formeln.

In der Formel $\forall x F$ bzw. $\exists x F$ bindet der Quantor die Variable x. Ist eine Variable in einer Formel nicht gebunden, heißt sie *freie Variable*

Einer Formel kann nun ein Wahrheitswert *wahr* oder *falsch* zugeordnet werden. Wie von Alfred Tarski entwickelt, wird dazu eine Abbildung der Konstanten-, Funktions- und Prädikatssymbole in eine nicht-leere Menge (das Universum, die Domäne) auf Elemente, Abbildungen und Relationen über dieser Menge mit entsprechender Stelligkeit angegeben. Diese Abbildung wird *Interpretation* genannt. Die Wahrheitswerte der Junktoren sind durch die üblichen Wahrheitstabellen festgelegt. Die Formel $\exists x F$ wird von der Interpretation erfüllt, falls es eine Belegung von x mit einem Element der Menge gibt, die F erfüllt. Die Formel $\forall x F$ wird von der Interpretation erfüllt, falls für alle Belegungen von x mit einem Element der Menge F erfüllt ist.

Eine Interpretation, die eine Formel F erfüllt, heißt auch *Modell* von F. Eine Formel F heißt

allgemeingültig (Tautologie), falls sie in *jeder* Interpretation wahr ist,

erfüllbar, falls sie in *mindestens einer* Interpretation wahr ist,

falsifizierbar, falls sie in *mindestens einer* Interpretation falsch ist und

unerfüllbar (Kontradiktion), falls sie in *jeder* Interpretation falsch ist.

Ein Kernbegriff der Logik ist der Begriff der *Folgerung*.

Folgerung: Eine Formel G *folgt* aus einer Formel F (geschrieben $F \models G$), wenn alle Modelle von F auch Modelle von G sind.

In PL1 ist durch die Gültigkeit des *Deduktionstheorems* eine Rückführung der Folgerung auf das Problem des Nachweises der Allgemeingültigkeit einer Formel möglich.

Deduktionstheorem: Für alle Formeln F, G ohne freie Variablen gilt $F \models G$ genau dann, wenn die Formel $F \Rightarrow G$ allgemeingültig ist.

Nun ist es möglich, den Begriff der *tautologischen Äquivalenz* einzuführen. Eine tautologische Äquivalenz ist eine Äquivalenz zweier Formeln, die in jeder Interpretation wahr ist. Als Beispiel sei genannt (seien F und G Formeln):

$$(\neg F \vee G) \Leftrightarrow (F \Rightarrow G)$$

Wegen der Gültigkeit des Deduktionstheorems können nun in Formeln Teile, die einer Seite einer tautologischen Äquivalenz entsprechen, durch die andere Seite ersetzt werden,

ohne daß sich der Wahrheitswert der Formel insgesamt ändert, d.h., jedes Modell der Ausgangsformel ist auch ein Modell der erzeugten Formel und umgekehrt. Diese Umformungen werden daher auch *modellerhaltende Transformationen* genannt.

Ein *Kalkül* ergänzt eine Logik um den syntaktischen Begriff des *Ableitens*, der mit dem semantischen Folgerungsbegriff korreliert. Dazu ist im Fall eines positiven Kalküls eine Menge allgemeingültiger Axiome (Tautologien) bzw. im Fall eines negativen Kalküls eine Menge unerfüllbarer Axiome (Kontradiktionen) gegeben.[14] Zusätzlich zu den Axiomen hat ein Kalkül eine Menge von *Schlußregeln*, mit Hilfe derer durch reine syntaktische Symboltransformation aus einer Menge von Formeln eine neue Formel erzeugt (abgeleitet) werden kann. Wenn aus den Formeln $F_1, \ldots, F_n$ die Formel F durch eine Sequenz von Anwendungen der Schlußregeln abgeleitet werden kann, schreibt man

$$F_1, \ldots, F_n \vdash F$$

Bei einem *deduktiven* Kalkül wendet man die Schlußregeln solange an, bis die zu beweisende (allgemeingültige bzw. unerfüllbare) Formel abgeleitet werden konnte. Bei einem *Testkalkül* wendet man die Schlußregeln solange auf die zu beweisende Formel an, bis eins der Axiome des Kalküls erzeugt wurde.

Zum Schluß sei noch ein Zusammenhang zwischen Folgerung und Ableitung durch die Begriffe *Korrektheit* und *Vollständigkeit* eines Kalküls gegeben:

Korrektheit eines Kalküls: Ein Kalkül ist *korrekt*, wenn gilt:
Wenn $F_1, \ldots, F_n \vdash F$, dann $F_1, \ldots, F_n \models F$.

Vollständigkeit eines Kalküls: Ein Kalkül ist *vollständig*, wenn gilt:
Wenn $F_1, \ldots, F_n \models F$, dann $F_1, \ldots, F_n \vdash F$.

1.1.3 Die Entwicklung eines Modells

Generelle Überlegungen wie im vorigen Abschnitt haben die Aufgabe, einen Rahmen aufzubauen, der die konkrete Arbeit an einer Modellierungsaufgabe bestimmt; sie sind nicht geeignet, auch tieferliegende Probleme vorwegzunehmen. Das geschieht besser durch die Analyse eines übersichtlichen – aber nicht-trivialen – Beispiels.

Was kann man von der Analyse eine konkreten Beispiels erwarten? Zunächst wird man sich einen Ansatz verschaffen, mit dem eine zentrale Eigenschaft der Domäne dargestellt wird. An diesem Ausgangspunkt wird man Aufschluß über die Verbindung von Axiomen und den Fragen gewinnen, die man an dieses kleine Modell stellen kann. Daraus kann man die

[14] Da in PL1 eine Formel genau dann allgemeingültig ist, wenn ihre Negation unerfüllbar ist, sind beide Arten von Kalkülen gleichwertig.

für die Aufgabe wesentlichen Dienstleistungen eines operationalen Systems gewinnen und eventuell Revisionen im Modell vornehmen, wenn diese Dienstleistungen nicht befriedigend zu erbringen oder zu interpretieren sind.

Durch interne Kritik am vorliegenden Modell wird man zu einem neuen Modell gelangen, das dem vorigen funktional entspricht, aber intern unter Umständen anders strukturiert ist. Durch Abstraktion wird man mit geringem Aufwand Möglichkeiten der Übertragung auf fremde, aber verwandte Domänen nachbilden.

Schließlich wird man Modell und Dienstleistungen an der Praxis erproben und Erweiterungen des inneren Aufbaus des Modells vornehmen, bis ein Zustand erreicht ist, der die gestellte Aufgabe zu lösen gestattet.

Hier ist das Szenario:

> Gegeben ist ein technisches Gerät (in diesem Fall ein Staubsauger). Das Modell soll wenigstens zwei Aufgaben unterstützen: Zunächst einmal soll der Grund für ein eventuelles Nichtfunktionieren des Geräts aus dem Modell angegeben werden können; auf der anderen Seite soll es möglich sein, Schritte anzugeben, mit denen aus den einzelnen Komponenten eines Staubsaugers ein funktionsfähiges Ganzes aufgebaut werden kann.

Verfällt man vorübergehend in eine weniger verpflichtende Sprechweise, so sind also Aufgaben der *Diagnose* bzw. *Konfiguration* technischer Systeme zu lösen.

Es muß bei dieser Verwendung des Begriffs „Diagnose“ vorab allerdings deutlich gemacht werden, daß wir ihn hier in einer Art verwenden, die von der gebräuchlichen Verwendung etwa bei [Reiter 87] abweicht. Diagnose wird in diesem Abschnitt als logische Schlußfolgerung dargestellt und nicht als Problem, die beobachteten (Fehler-) Symptome mit einer maximalen Menge von *Normalitätsannahmen* konsistent zu bekommen. Realisiert man die Diagnose-Dienstleistung in der Form, wie sie hier angegeben ist, zeigt sich, daß es in der Tat sehr viele solche Diagnosen gibt, die dann gegeneinander bewertet werden müssen.

Betrachtet man nun wieder das Szenario, so erkennt man, daß die Domäne hier deutlich eingeschränkt wurde auf „Staubsauger“, was zunächst ganz angenehm aussieht. Verschaffen wir uns also zunächst einen Überblick über den gegebenen Staubsauger:

Wir erkennen eine kleine Kontrollampe am Gehäuse und einen Stecker, der durch ein Kabel mit dem Staubsauger verbunden ist. Ferner erfahren wir durch Öffnen des Gehäuses, daß sich im Innern des Staubsaugers ein Motor befindet, der letzlich für die eigentliche Saugleistung verantwortlich ist.

Einige dieser Beobachtungen lassen sich leicht konzeptionalisieren:

$$(S1)\quad \forall staubi : Staubsauger(staubi) \Rightarrow \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi, l) \wedge Motor(m) \wedge HatAlsTeil(staubi, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s)$$

Die Entscheidung, in (S1) den Implikationspfeil nur in einer Richtung verlaufen zu lassen, treffen wir nur vorläufig.

1.1.3.1 Notwendige und hinreichende Bedingungen

Unser Modell besteht nun aus dem Axiom (S1) und wir wollen untersuchen, was dieses Modell auf verschiedene Fragen antworten kann. Wir nehmen also das Axiom (S1) als gegeben an.

Erste Frage: *Was folgt aus* $Staubsauger(staubi_1)$*?*. Die Antwort:

$$(A1)\quad \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi_1, l) \wedge Motor(m) \wedge HatAlsTeil(staubi_1, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi_1, s)$$

Mit etwas Phantasie erkennt man an dieser Antwort bereits den Ansatz eines formalen Konstruktionsplans für einen Staubsauger. Leider erlaubt es der Implikationspfeil in (S1) nicht, (A1) als vollständige Antwort auf die Frage *Wie baut man einen Staubsauger zusammen?* zu interpretieren; dazu müßte der Pfeil andersherum verlaufen! Formal: (A1) ist notwendig, aber nicht hinreichend.

Zweite Frage: *Was folgt aus* $\neg Staubsauger(staubi_1)$*?*

Antwort: *Nichts Neues!* (S1) und $\neg Staubsauger(staubi_1)$ sind miteinander verträglich und auch wenn $staubi_1$ eine Lampe, einen Motor und einen Stecker besitzt, ist das kein Widerspruch zu $\neg Staubsauger(staubi_1)$. Darauf hatten wir schon geachtet.

Dritte Frage: *Was folgt aus der Formel*

$$\neg\exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi_1, l) \wedge Motor(m) \wedge HatAlsTeil(staubi_1, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi_1, s)$$

Antwort: $\neg Staubsauger(staubi_1)$, wie man an der logischen Kontraposition von (S1) ablesen kann.

Vierte Frage: *Was folgt aus der Formel*

$$\exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi_1, l) \wedge \\ Motor(m) \wedge HatAlsTeil(staubi_1, m) \wedge \\ Stecker(s) \wedge HatAlsTeil(staubi_1, s)$$

Antwort: Wieder nichts Interessantes!

Zusammenfassend kann man sagen, daß unser Axiom (S1) ganz gut Fragen nach notwendigen Teilen eines Staubsaugers beantwortet, Dinge, die keine Staubsauger sind, zur Kenntnis nimmt und einwandfreie „Nichtstaubsauger" meldet. Das ist nicht eben viel, aber – wie man sieht – gut im Einklang mit der Anschauung.

Welche Auswirkungen hätte es, wenn wir in (S1) den Pfeil doch in beide Richtungen zeigen lassen?

$$(S2) \quad \forall staubi : Staubsauger(staubi) \Leftrightarrow \\ \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi, l) \wedge \\ Motor(m) \wedge HatAlsTeil(staubi, m) \wedge \\ Stecker(s) \wedge HatAlsTeil(staubi, s)$$

Stellen wir dieselben vier Fragen an (S2), so bekommen wir Antworten, die wesentlich präziser sind:

Erste Frage: Die Antwort ist wiederum (A1), aber wir dürfen dies als vollständigen Konstruktionsplan eines Staubsaugers auffassen.

Zweite Frage: Die Antwort ist eine recht präzise Diagnose:

$$\neg\exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi_1, l) \wedge \\ Motor(m) \wedge HatAlsTeil(staubi_1, m) \wedge \\ Stecker(s) \wedge HatAlsTeil(staubi_1, s)$$

Dritte Frage: Das Ergebnis ist dasselbe wie bei (S1).

Vierte Frage: Nun folgt brav $Staubsauger(staubi_1)$.

An dieser Stelle sind einige Tatsachen festzustellen:

- Die Antworten des Modells sind präziser und aussagekräftiger geworden.
- Das Modell hingegen ist – so wie es aussieht – nicht mehr unbedingt im Einklang mit der Realität.

Formal ist das natürlich kein Wunder: Axiom (S2) besteht formal aus einer logischen Konjunktion von (S1) und einem Axiom, das logisch von (S1) unabhängig ist. Der Übergang von (S1) zu (S2) hat einige Tarski-Modelle ausgeschieden, und in den verbleibenden gelten

natürlich mehr Sätze.

Der „Realitätsverlust“ von (S2) besteht darin, daß – salopp ausgedrückt – alles, was eine Lampe, einen Motor und einen Stecker als Teil hat, als ein Staubsauger erkannt wird, auch wenn es z.B. tatsächlich ein Mixer sein sollte. Es ist eins der Hauptprobleme der Modellbildung, festzustellen, in welchem Ausmaß dies schädlich ist. Letzlich wird man sich für eine von zwei Positionen entscheiden müssen:

1. Die Beschreibung von Domäne und Aufgabe stellen klar fest, daß über Staubsauger geredet wird. Das Modell darf problemlos auf diese Domäne eingeschränkt werden; die gesteigerte Aussagefähigkeit wird mit Freuden zur Kenntnis genommen.
2. Es ist wahr, daß in der Problemstellung lediglich von Staubsaugern die Rede ist. Es ist aber zu erwarten, daß viele Teile des Modells auf andere Domänen generalisiert werden können. Eine solche Modellierung, wie (S2) sie darstellt, darf nicht unbesehen akzeptiert werden.

1.1.3.2 Abgeleitete Konzeptionalisierungen

Nun war im Szenario, von dem ausgegangen wurde, nicht davon die Rede, ein System zu realisieren, daß Staubsauger erkennt. Wir erweitern unser Modell entsprechend um beobachtbare oder bekannte Zusammenhänge, die z.B. nicht-triviale diagnostische Fragestellungen erlauben. Wir notieren zunächst zwei Axiome, die wahrnehmbare Tatsachen (Symptome) mit dem inneren Aufbau des Objekts und generell bekannten Wissenseinheiten in Verbindung setzen:

$$
\begin{aligned}
(L1)\quad & \forall l : Leuchtet(l) \Leftarrow \\
& \quad Lampe(l) \wedge \\
& \quad\quad [\exists staubi, s, sd : Staubsauger(staubi) \wedge HatAlsTeil(staubi, l) \wedge \\
& \quad\quad\quad Stecker(s) \wedge HatAlsTeil(staubi, s) \wedge \\
& \quad\quad\quad Steckdose(sd) \wedge StecktIn(s, sd)]
\end{aligned}
$$

$$
\begin{aligned}
(M1)\quad & \forall m : Brummt(m) \Leftarrow \\
& \quad Motor(m) \wedge \\
& \quad\quad [\exists staubi, s, sd : Staubsauger(staubi) \wedge HatAlsTeil(staubi, m) \wedge \\
& \quad\quad\quad Stecker(s) \wedge HatAlsTeil(staubi, s) \wedge \\
& \quad\quad\quad Steckdose(sd) \wedge StecktIn(s, sd)]
\end{aligned}
$$

Hier wird einiges an Details darüber dargestellt, was über elementare Zusammenhänge innerhalb technischer Systeme bekannt ist. Nehmen wir an, es wären folgende Tatsachen

über den aktuellen Zustand der Welt bekannt:

$$(F) \quad Motor(m_1) \wedge Staubsauger(staubi_1) \wedge HatAlsTeil(staubi_1, m_1) \wedge Stecker(s_1) \wedge HatAlsTeil(staubi_1, s_1) \wedge Steckdose(sd_1).$$

Was in dieser Situation den Motor m_1 vom Brummen abhält, ist die Tatsache, daß wir das fehlende Fakt „StecktIn(s_1,sd_1)“ nicht beweisen können. In der Tat läßt sich aus

- dem „Weltmodell“ (M1),
- dem „Fakt“ (F) und
- dem „Fehlersymptom“ $\neg Brummt(m_1)$

die „Diagnose“ $\neg StecktIn(s_1,sd_1)$ beweisen. Problematisch ist dabei lediglich, daß die Idee eines Beweises erfordert, daß der zu beweisende Satz bekannt ist, bevor man seine Gültigkeit nachweisen kann. Unsere diagnostische Aufgabe erfordert eine gewisse Modifikation dahingehend, daß wir alle möglichen Diagnosen geliefert haben wollen, die aus Modell, Fakten und Symptomen folgen.

Dennoch zeigt das Modell einige Eigenschaften, die leicht zu beheben sind. Das auffällige ist, daß Teile der verwendeten Formeln auf eine gemeinsame Gegebenheit der Domäne hindeuten:

$$\exists staubi, s, sd : Staubsauger(staubi) \wedge HatAlsTeil(staubi, x) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s) \wedge Steckdose(sd) \wedge StecktIn(s, sd)$$

Anscheinend stellt diese Formel eine eigenständige Konzeptionalisierung der Domäne dar, die es verdient, separat behandelt zu werden; z.B.:

$$(A1) \quad \forall x : Angeschlossen(x) \Leftarrow \exists staubi, s, sd : Staubsauger(staubi) \wedge HatAlsTeil(staubi, x) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s) \wedge Steckdose(sd) \wedge StecktIn(s, sd)$$

Dies ergibt einige Vereinfachungen in den Axiomen über Lampen und Motoren:

$$(L2) \quad \forall l : Leuchtet(l) \Leftarrow Lampe(l) \wedge Angeschlossen(l)$$

$$(M2) \quad \forall m : Brummt(m) \Leftarrow Motor(m) \wedge Angeschlossen(m)$$

Es ist nicht einfach, die Nützlichkeit solcher abgeleiteter Konzeptionalisierungen zu be-

urteilen. Sie spielen eine Rolle, die der *Modularisierung* bei der „klassischen" Software-Entwicklung ähnlich ist: Sie beseitigen *Redundanzen* in der Darstellung, sie erlauben *lokale Erweiterungen*, sie entlasten andere Konzepte von irrelevanten Details und machen das Modell übersichtlicher. Andererseits will man von eigenständigen Konzepten des Modells eine gewisse sachliche Realität verlangen, die es möglich macht, über den Inhalt und die konkrete Ausformung des Konzepts zu diskutieren.

1.1.3.3 Abstraktion

Beim Axiom (A1) handelt es sich offenbar um ein ziemlich generelles Konzept, das mit dem Begriff *angeschlossen* assoziiert worden ist. In der vorliegenden Ausprägung allerdings sieht es ausgesprochen merkwürdig aus, weil lediglich Teile eines *Staubsaugers* als Teilnehmer an diesem Konzept auftreten können. Eine generelle Beschreibung dieses Konzepts wird sich von diesem Umstand, der ja aus der gegebenen Domänenbeschreibung resultiert, trennen wollen. Nimmt man diese Notwendigkeit als gegeben an, so nimmt man in Kauf, die darzustellende Domäne freiwillig zu verlassen und – wenigstens teilweise – mögliche andere ähnliche Domänen zu antizipieren.

Die Generalisierung dieses Axioms überträgt das Konzept *angeschlossen* auf eine Domänenbeschreibung, die von Staubsaugern weitgehend abstrahiert:

$$(A2)\quad \forall x : Angeschlossen(x) \Leftarrow \exists egeraet, s, sd : ElektrischesGeraet(egeraet) \wedge HatAlsTeil(egeraet, x) \wedge Stecker(s) \wedge HatAlsTeil(egeraet, s) \wedge Steckdose(sd) \wedge SteckktIn(s, sd)$$

Das neue Konzept *ElektrischesGeraet* stellt eine *Generalisierung* des bekannten Staubsaugers dar, was wir explizit im Modell verankern müssen:

$$(E1)\quad \forall x : ElektrischesGeraet(x) \Leftarrow Staubsauger(x)$$

Die Axiome (M1), (L1), (A2) und (E1) stellen nun das neu strukturierte Modell der Domäne dar, das nun im Hinblick auf sein Verhalten im Problemlösungsprozeß untersucht werden kann.

1.1.3.4 Performanz

Untersuchen wir zunächst die diagnostischen Fähigkeiten des Modells. Sie haben sich durch die internen Umstrukturierungen im Modell nicht verändert, auch nicht zum besseren; lediglich die Ableitungen sind länger geworden, weil mehr Zwischenschritte mit den neuen Konzepten auftreten. Das bedeutet in der Praxis, daß das System länger braucht, um eine Antwort aufzubauen.

Im nächsten Schritt wird man dieses Verhalten an einem realen Staubsauger überprüfen, dessen Motor nicht brummt. Die Diagnose „¬ StecktIn(s_1,sd_1)“ deutet darauf hin, daß der Stecker nicht eingesteckt ist, was man schnell überprüfen kann. Wie auch immer diese Prüfung im Szenario abläuft, hinterher wird man sicher sein können, daß er nun in der Steckdose steckt. Nun schnell am Motor gehorcht und festgestellt, daß er immer noch nicht brummt. Also fügen wir zu den Fakten (F) hinzu $StecktIn(s_1,sd_1)$ und das Symptom $\neg Brummt(m_1)$ zur Diagnose gestellt!

Das nun auftretende Verhalten der erforderlichen Dienstleistung ist formal schwierig zu spezifizieren, denn nun ist offenbar ein logischer Widerspruch aufgetreten: Aus den Axiomen, (F) und $StecktIn(s_1,sd_1)$ folgt $Brummt(m_1)$, was zusammen mit dem Symptom $\neg Brummt(m_1)$ den klassischen Widerspruch ergibt.

Das Auftreten einer solchen *Inkonsistenz* ist ein deutlicher Hinweis darauf, daß das Modell keine ausreichende Übereinstimmung mit den tatsächlichen Gegebenheiten aufweist. Hier helfen keine neuen Konzeptionalisierungen und auch keine Abstraktion; das Modell muß inhaltlich revidiert werden.

1.1.3.5 Modellrevision

Zunächst muß man einmal herausbekommen, warum dieser Motor tatsächlich nicht brummt. Die Ursache wird man dann ins Modell einbringen und hoffen, daß sich die erforderliche Revision eingrenzen läßt. Ein Interview mit einem Spezialisten für Motoren ergibt die Tatsache, daß Motoren kaputt gehen können. Kaputte Motoren brummen auch dann nicht, wenn sie korrekt angeschlossen sind.

Wir konzeptionalisieren diesen Umstand in ein einstelliges Prädikat Ok, das den Wert *wahr* annimmt, wenn das Argument intern in Ordnung ist:

$$(M3) \quad \forall m : Brummt(m) \Leftarrow Motor(m) \wedge Angeschlossen(m) \wedge Ok(m)$$

Eine kurze Überlegung liefert dann auch für die Kontrollampe ein entsprechend modifiziertes Axiom:

$(L3) \quad \forall l : Leuchtet(l) \Leftarrow Lampe(l) \wedge Angeschlossen(l) \wedge Ok(l)$

Ersetzen wir (M2) durch (M3), liefert das Modell als Antwort auf die diagnostische Aufgabe keinen Widerspruch mehr, sondern die Diagnose $\neg Ok(m_1)$. Das neue Modell bezieht also mehr Eigenschaften der Domäne ein als das alte und paßt sich so den beobachteten Phänomenen besser an. Um ein Schlagwort für diesen Unterschied zur Hand zu haben, nennen wir das Modell unter Einbeziehung von (M3) und (L3) „tiefer" als das mit (M2) und (L2).

1.1.3.6 Modellvertiefung

Was ist durch die Modifikation des Modells im letzten Abschnitt geschehen? Zunächst ein Blick auf die Semantik der beiden Axiomatisierungen. Wir erhalten eine Idee davon, wenn wir (M3) anders formulieren:

$(M3) \quad \forall m : (Brummt(m) \Leftarrow Motor(m) \wedge Angeschlossen(m)) \vee \neg Ok(m)$

Formal geht das „tiefere" Axiom (M3) also durch Hinzufügen einer *Disjunktion* aus (M2) hervor. Das bedeutet, daß jedes Tarski-Modell einer Axiomatisierung mit (M3) auch Tarski-Modell einer Axiomatisierung mit (M2) ist; genau anders herum als im obigen Fall der Umkehrung des Implikationspfeils beim Übergang von (S1) zu (S2).

Dann müsste das tiefere Modell in bestimmten Fällen weniger präzise antworten. Probieren wir das aus, indem wir nun wieder den Stecker herausziehen, also das Symptom $\neg Brummt(m_1)$ ohne $SteckstIn(s_1,sd_1)$ und unter Verwendung von (M3) diagnostizieren lassen: Die Antwort ist $\neg SteckstIn(s_1, sd_1) \vee \neg Ok(m_1)$, also tatsächlich schwächer als vorher.

Das ist an sich nichts Aufregendes, denn letztlich ist das genau das, was man erreichen wollte: Die „stärkste" Aussage, die ein Modell liefern kann, ist ja die Meldung einer Inkonsistenz, und das möchte man gerade vermeiden. Andererseits ist die resultierende Schwäche der Antworten unschön, und dieses Dilemma ist das Hauptproblem bei der Modellierung einer Domäne bei gegebener Aufgabe.

Als weiteres Detail eines solchen Modells kann nun die *Annahme* hinzugefügt werden, daß für alle Geräte und Komponenten der Wert $Ok(x)$ auf *wahr* evaluiert. Da eine solche Annahme nicht immer gehalten ist, muß sie aber bei Widerspruch zurückgenommen werden. Ein Mechanismus, mit rücknehmbaren Annahmen zu schließen, wird durch nichtmonotone Logiken zur Verfügung gestellt, die in dem Beitrag über „Nichtmonotones Schließen" ausführlicher dargestellt werden.

1.1.3.7 Anomalien

Die bisherigen Axiomatisierungen zeigen bei genauerem Hinsehen allerdings Probleme, die auf den ersten Blick vielleicht nicht sehr offensichtlich sind. Zurück zur Konfiguration eines Staubsaugers mit (S1):

$$(S1) \quad \forall staubi : Staubsauger(staubi) \Rightarrow \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi, l) \wedge Motor(m) \wedge HatAlsTeil(staubi, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s)$$

Eine sehr „sparsame“ Konfiguration von *zwei* Staubsaugern besteht aus der Formel:

$$Lampe(teil_1) \wedge Motor(teil_1) \wedge Stecker(teil_1) \wedge HatAlsTeil(staubi_1, teil_1) \wedge HatAlsTeil(staubi_2, teil_1).$$

Das ist offensichtlich nicht intendiert. Man wird versuchen, derartige Anomalien entweder auf der Modellebene oder der Ebene des logischen Formalismus auszuschließen. Man analysiert, daß in der angegebenen Formel im wesentlichen zwei typische Anomalien bei logischer Rekonstruktion auftreten:

Implizite Annahme über Typen: Natürlich gibt es ein solches Universalteil wie q, das gleichzeitig Lampe, Motor und Stecker ist, in der Realität nicht. Diese implizite Domäneneigenschaft muß im Modell explizit gemacht werden.

Implizite Annahmen über Beziehungen: Die Vorstellung, zwei voneinander verschiedene Staubsauger könnten sich ihre Komponenten teilen, ist in der Realität nicht zu beobachten.

Was wird man tun? Der einfachste Weg besteht darin, alle diese Dinge explizit im Modell auszuschließen. Das führt zu einer Anzahl von Axiomen, die z.B. die Unverträglichkeit verschiedener Typen ausdrücken. Diese Axiome sind strukturell alle gleich und haben eine Form wie z.B.:

$$(D1) \quad \forall x : Lampe(x) \Rightarrow \neg Motor(x).$$

Die zweite Anomalie hingegen läßt sich mit einem einzigen Axiom beseitigen:

$$(T) \quad \forall x, y, z : ((x \neq y \wedge HatAlsTeil(x, z)) \Rightarrow \neg HatAlsTeil(y, z)).$$

1.1.3.8 Axiomenschemata

Nach einer umfangreichen Analyse der logischen Rekonstruktion eines sehr kleinen Gegenstandsbereichs wird man das, was man durch ein mehr oder weniger strukturiertes Vorgehen im Einzelfall erreicht hat, einer generellen Analyse unterziehen. Zunächst geht es hier darum, nach Phänomenen zu suchen, die sich zu generellen Modellcharakteristika verallgemeinern lassen.

Das Ergebnis dieses Schritts wird auf den ersten Blick lediglich eine *Notation* sein, die kurze Schreibweisen für häufig vorkommende Axiome des logischen Modells ermöglicht. Allerdings werden wir darauf achten, daß die generalisierten Axiome eine Charakterisierung auf einer sprachlichen Ebene erhalten, die Brachman als „epistemologische Ebene" bezeichnet hat [Brachman 79].

Im folgenden Schritt kann dann der Übergang zu Formalismen vorbereitet werden, die es erlauben, unter Verwendung aller oder einiger dieser neuen Bausteine logisch konstruierte Modelle zu realisieren und die antizipierten Dienstleistungen zur Verfügung zu stellen. Im nächsten Abschnitt wird dann an einigen Beispielen diese Realisierung durchgeführt.

Wir haben bei der Entdeckungsreise durch die logische Rekonstruktion eines Staubsaugers eine große Anzahl von Phänomenen beobachtet, die sich dem naiven Herangehen an das Problem in den Weg gestellt haben. Zu jedem der Probleme sind uns Axiome eingefallen, die es beseitigen. Es erscheint an der Zeit, diese Axiome einmal näher anzuschauen, um in genereller Art eine Charakterisierung zu gewinnen, die es erlaubt, den Übergang von Axiomen zu *Repräsentationsformalismen* vorzubereiten.

Die erste Klasse von Axiomenschemata, die aufgetreten ist, wird etwa durch das Axiom (E1) repräsentiert. Es ist eine Instanz des Schemas

$$(G) \quad \forall x : A(x) \Leftarrow B(x),$$

das eine begriffliche Hierarchisierung zwischen A und B darstellt.

Das Axiom (D1) gibt Anlaß zu einer völlig anderen epistemologischen Kategorisierung; es ist eine Instanz des Schemas

$$(D) \quad \forall x : C(x) \Rightarrow \neg D(x),$$

in dem die Unverträglichkeit zweier Begriffe formuliert wird.

Vervollständigt man diese Axiomenschemata durch ein drittes, das allerdings in unserem Modell nicht aufgetreten ist, erreicht man das Schema $\forall x : E(x) \Leftarrow \neg F(x)$, dessen epistemologische Charakterisierung etwas schwieriger zu formulieren ist. Es deutet darauf hin,

daß die beteiligten Begriffe die gesamte Domäne *überdecken*, d.h. jedes Individuum ist entweder ein E oder ein F oder beides.
Man erkennt, warum dieses Schema in unserem Modell (und wahrscheinlich in fast allen anderen) nicht vorkommt: Es ist sehr stark, schränkt die Domäne wesentlich ein und erlaubt auf der anderen Seite den Beweis sehr ungewöhnlicher Aussagen. Man verwendet ein solches Überdeckungsaxiom bestenfalls als *relativierte* Variante, etwa in einem Schema:

$$(RC) \quad \forall x : (\neg A(x) \vee B(x)) \Leftarrow \neg C(x).$$

Die Aussage dieses Schemas besteht darin, daß die Überdeckung von B und C nur relativ zu A besteht, was man daran erkennt, daß die Aussage $A(s) \wedge \neg C(s)$ die Konsequenz $B(s)$ hat. Andererseits folgt aus $\neg B(s) \wedge \neg C(s)$ kein Widerspruch, sondern lediglich das akzeptable $\neg A(s)$.
Ähnlich, aber etwas komplizierter gestaltet sich die Charakterisierung des Axioms (T), das – nach einigen Umformungen – die folgende Gestalt bekommt:

$$(T) \quad \forall x, y, z : (HatAlsTeil(x, z) \wedge HatAlsTeil(y, z)) \Rightarrow (x = y).$$

Schematisierung liefert

$$(IF) \quad \forall x, y, z : (R(x, z) \wedge R(y, z)) \Rightarrow (x = y).$$

Formal bedeutet das nicht mehr als daß die inverse Relation R^{-1} eine partielle Funktion ist.
Nehmen wir für einen Moment an, daß diese Schemata für die Praxis der logischen Modellierung in der Tat relevant sind und oft vorkommen, ist es eine Frage der Ökonomie, ihre Notation etwas zu vereinfachen. Wir notieren also die Instanzen eines Schemas durch Angabe des Schema-Namens und der beteiligten Prädikate bzw. Relationen. Also z.B.

- *G[ElektrischesGeraet,Staubsauger]* als Notation für (E1),
- *D[Lampe,Motor]* als Notation für (D1) und
- *IF[HatAlsTeil]* als Notation für (T).

Hier nun eine Liste von Axiomenschemata, die in der praktischen Arbeit der logischen Rekonstruktion auftreten und eine relevante epistemologische Charakterisierung besitzen:

- G[A,B]: $\forall x : (A(x) \Leftarrow B(x))$
- DEF[A,B,C]: $\forall x : (A(x) \Leftrightarrow B(x) \wedge C(x))$, die *Definition*
- D[A,B]: $\forall x : (A(x) \Rightarrow \neg B(x))$

- RC[A,B,C]: $\forall x : (\neg A(x) \vee B(x)) \Leftarrow \neg C(x)$
- DC[A,B,C]: RC[A,B,C] und D[B,C]; die *disjunkte Überdeckung*
- IF[R]: $\forall x, y, z : (R(x, z) \wedge R(y, z)) \Rightarrow (x = y)$
- F[R]: $\forall x, y, z : (R(x, y) \wedge R(x, z)) \Rightarrow (y = z)$, eine *funktionale* Relation
- TRANS[R]: $\forall x, y, z : (R(x, y) \wedge R(y, z)) \Rightarrow R(x, z)$
- INV[R,S]: $\forall x, y : R(x, y) \Leftrightarrow S(y, x)$
- SYMM[R]: $\forall x, y : R(x, y) \Rightarrow R(y, x)$ oder auch INV[R,R]

Gelingt es nun, Axiome des logischen Modells durch geeignete Schemata zu formulieren, läßt sich ein Submodell unseres Modells mit Hilfe dieser Schemata aufschreiben:

G[ElektrischesGeraet,Staubsauger]

IF[HatAlsTeil]

D[Motor,Lampe]

D[Lampe,Stecker]

D[Motor,Stecker]

Eine Familie solcher Schemata, die der KL-ONE Familie (s. [Schmolze, Woods 92]) zugerechnet und als logik-basierte Framesprache interpretiert werden kann, wird im folgenden Abschnitt mit einer etwas abstrakteren Notation eingeführt und mit ihren Eigenschaften etwas genauer diskutiert.

1.1.4 Repräsentationsformalismen, Inferenzalgorithmen, und Berechenbarkeitseigenschaften

Die Fragen der konzeptionellen Modellierung sind im letzten Abschnitt angesprochen. Der nächste Schritt besteht darin, die konzeptionellen Einheiten dieses Modells in eine Form zu bringen, die es erlaubt, die gewünschten Dienstleistungen möglichst effizient zu erbringen.

Dabei stellt sich das Problem, daß unter Umständen *mehrere* Ausdrucksmittel zur Verfügung stehen, die sich in einigen wichtigen Punkten voneinander unterscheiden. Die Diversifikation des Instrumentariums der Wissensrepräsentation ist einerseits bedauerlich, andererseits wohl unvermeidlich, um zum einen eine *natürliche Darstellung* und zum anderen eine *effiziente Verarbeitung* des dargestellten Wissens anzubieten.

Nach den Bemerkungen in den vorhergehenden Abschnitten liegt es nahe, die Prädikatenlogik erster Stufe als *den* Repräsentationsformalismus zu wählen und dazu ein geeigne-

tes Deduktionssystem auf dem Rechner zu implementieren. Tatsächlich ist dies auch eine mögliche Vorgehensweise, die von verschiedenen Forschern verfolgt wird (siehe auch das Kapitel über Deduktionssysteme). Der große Vorteil dieser Lösung ist, daß die Prädikatenlogik erster Stufe ein sehr ausdrucksstarker Formalismus ist. Natürlich gibt es auch Gründe, die gegen ein solches Vorgehen sprechen, sonst hätte sich diese Lösung bereits als Universallösung durchgesetzt.

Zum ersten ist die Prädikatenlogik erster Stufe von ihrer Ausdruckskraft zu *mächtig*, als daß man *effiziente* Verfahren für ihre Verarbeitung angeben könnte. Tatsächlich ist das Problem, ob eine Aussage aus einer anderen logisch folgt, *unentscheidbar*, d.h. wir können kein Deduktionssystem implementieren, daß korrekt und vollständig ist und immer terminiert. Ein Einwand gegen dieses Argument könnte sein, daß man das Wissen in geeigneter Weise codiert und zusätzlich Kontrollinformationen bereit stellt, so daß man zusichern kann, daß ein System nach endlicher Zeit eine Antwort gibt. In diesem Fall würde man jedoch nicht mehr von Wissensrepräsentation, sondern von Programmierung sprechen. PROLOG ist zum Beispiel ein System, das genau diese Strategie der oft auch „prozedural“ genannten Repräsentation von Wissen unterstützt.

Zum zweiten ist die Prädikatenlogik erster Stufe, obwohl zu mächtig für eine effiziente Verarbeitung, manchmal nicht ausdrucksstark genug. Beispielsweise können wir in der Prädikatenlogik erster Stufe nicht die transitive Hülle einer Relation definieren oder das Konzept der natürlichen Zahlen vollständig charakterisieren. Auch entziehen sich Schlußweisen des Alltags, die auf unsichere Informationen aufbauen, der Behandlung durch die Prädikatenlogik erster Stufe (siehe auch das Kapitel über „Nicht-monotone Logiken“).

Schließlich ist die Ausdruckskraft der Prädikatenlogik erster Stufe, die wir oben als Vorteil genannt haben, manchmal auch ein Nachteil, da diese einhergeht mit „epistemischer Neutralität“. Konkret bedeutet dies, daß wir zwar viele Bereiche des Alltagswissens, wie zum Beispiel Wissen über Zeit und Überzeugungen, innerhalb der Prädikatenlogik erster Stufe repräsentieren können, daß diese Dinge aber jeweils „neu erfunden“ werden müßten. An dieser Stelle könnte man einwenden, daß durch die Schaffung von „Theoriebibliotheken“ diesem Mißstand abgeholfen werden könnte. Dies ist aber bisher aber nicht geschehen und wegen des oben genannten Problems der Ineffizienz hätte es vermutlich auch nicht viel Sinn.

Aus den oben genannten Gründen (und auch einigen anderen, wie z.B., daß „Logik nichts mit den kognitiven Prozessen des Menschen gemein habe“ [Minsky 75]) hat man in der KI spezielle Repräsentationsformalismen entwickelt, wie z.B. *semantische Netze*, *Frames*, *Produktionsregeln* und spezielle Formalismen zur Repräsentation zeitlicher und räumlicher Aspekte [Brachman, Levesque 85]. Während in der Frühzeit der KI solche Repräsentati-

onsformalismen als etwas angesehen wurden, das verschieden von Logik ist, indem man kognitive Aspekte und Effizienzaspekte in den Vordergrund stellte [Minsky 75], herrscht heutzutage eine etwas andere Meinung vor. Der wesentliche Grund dafür ist, daß Logik das für Wissen am besten geeignete *analytische Werkzeug* zu sein scheint.

Newell [Newell 82] hat den Term *Wissensebene* (knowledge level) dafür geprägt, um Wissen und repräsentiertes Wissen auf einer sehr abstrakten, von Maschinen unabhängigen Weise zu analysieren. Das geeignete Werkzeug dafür ist seiner Meinung nach Logik. Newell geht sogar noch einen Schritt weiter und sagt, daß – frei übersetzt – „eine nicht-logische Analyse von Wissen Unfug ist".

Im folgenden werden wir einen Blick auf die verschiedenen Aspekte der Verbindung von Repräsentationsformalismen und Logik werfen und herausarbeiten, welche Vorteile eine solche Verbindung hat. Wir werden beispielhaft einen Repräsentationsformalismus einführen, der es uns erlaubt, einen Teil der Beispieldomäne zu beschreiben. Anhand dieses Formalismus werden wir dann auf den Aspekt der formalen Bedeutung, der *Semantik*, von Repräsentationssprachen eingehen und darauf aufbauend den *algorithmischen Aspekt* auf einer relativ abstrakten Ebene betrachten. Schließlich werden die *Berechenbarkeitseigenschaften* von Repräsentationsformalismen und -diensten beleuchtet.

1.1.4.1 Ein Repräsentationsformalismus

Als Beispiel eines Repräsentationsformalismus wollen wir eine *frame-basierte* Sprache betrachten, die den Sprachen der KL-ONE Familie [Nebel 90a; Schmolze, Woods 92] sehr nahe steht. Mit Hilfe dieser Sprache können wir Abstraktionsaspekte der Staubsaugerdomäne beschreiben und das Konstruieren und Erkennen von Staubsaugern als Dienst des Repräsentationssystems anbieten. Für die Diagnose müßten jedoch zusätzliche Repräsentationsmittel zur Verfügung gestellt werden.

Frame-Sprachen sind geeignet, Begriffe zu beschreiben und in Beziehung zu setzen. Jeder Frame beschreibt einen Begriff durch Angabe von allgemeineren Begriffen und Beschreibungen von Beziehungen zu anderen Begriffen. Im folgenden führen wir eine kleine Repräsentationssprache ein, die dieses (und noch etwas mehr) unterstützt. Unsere Repräsentationssprache enthält *Begriffssymbole*, die mit B bezeichnet werden und Objekte beschreiben, sowie *Rollensymbole*, die mit R bezeichnet werden und Beziehungen zwischen jeweils zwei Objekten beschreiben. Zudem gibt es verschiedene Operatoren, mit denen wir Rollen und Begriffe kombinieren können, um neue *Begriffsausdrücke* (im folgenden mit C bezeichnet) zu gewinnen. Die Repräsentationssprache hat, formuliert in einer (abstrakten)

BNF-Notation, folgende Gestalt:[15].

C	$\rightarrow$	B	Begriffssymbole
	$\mid$	$C \sqcap C'$	Begriffskonjunktion
	$\mid$	$C \sqcup C'$	Begriffsdisjunktion
	$\mid$	$\neg C$	Begriffsnegation
	$\mid$	$\forall R\colon C$	Werterestriktion
	$\mid$	$\exists R\colon C$	Existentielle Restriktion

Die Begriffskonjunktion bezeichnet einen Begriff, der die Bedingungen beider Begriffe enthält. Beispielsweise bezeichnet die Begriffskonjunktion (*Mann* $\sqcap$ *Elternteil*) den Begriff *Vater*. Ebenso verhält es sich mit der Begriffsdisjunktion und -negation. Die Werterestriktion bezeichnet Begriffe, bei denen die *Rollenfüller* der angegebenen Rolle R die Restriktionen des gegebenen Begriff's C erfüllen. So bezeichnet der Begriffsausdruck ($\forall$ *Kinder*: *Mann*) den Begriff „jemand, der nur männliche Kinder hat (unter der Voraussetzung, daß überhaupt Kinder vorhanden sind)“. Schließlich können mit Hilfe der Existenzrestriktion Begriffe beschrieben werden, bei denen es mindestens einen Rollenfüller gibt, der zu einem bestimmten Begriff gehört.

Nehmen wir an, daß unser Vokabular von Begriffssymbolen die folgenden Symbole enthält *Komponente, Gerät, E-Gerät, Mixer, Staubsauger, Stecker, Lampe, Motor, Stecker, Saugrohr, Mixstab* und unser Vokabular von Rollensymbolen das Symbol *Teil*. Dann könnten wir z.B. einen Staubsauger in folgender Weise beschreiben:

$$\textit{E-Gerät} \sqcap \forall\, \textit{Teil}\colon \textit{Komponente} \sqcap$$
$$\exists\, \textit{Teil}\colon \textit{Motor} \sqcap \exists\, \textit{Teil}\colon \textit{Lampe} \sqcap \exists\, \textit{Teil}\colon \textit{Saugrohr}.$$

Damit haben wir zwar einen Staubsauger beschrieben, aber die in der Beschreibung benutzten Begriffssymbole undefiniert gelassen. Eine vollständige Beschreibung aller oben erwähnten Begriffssymbole, die wir im folgenden *Terminologie* nennen wollen, könnte folgendermaßen aussehen:

$$\begin{aligned} \textit{Komponente} &\sqsubseteq \textit{Ding} \\ \textit{Motor} &\sqsubseteq \textit{Komponente} \end{aligned}$$

[15] Die folgende Notation entspricht der „Standardnotation“ für *terminologische Logiken*, in der Literatur auch *Begriffssprachen*, *Termsubsumptionssprachen*, *Beschreibungslogiken* oder *KL-ONE-basierte Sprachen* genannt, wie sie auf dem „International Workshop on Terminological Logics“, 1991, diskutiert wurde [Baader et al. 91]. Die spezielle Sprache, die wir hier betrachten, trägt den Namen $\mathcal{ALC}$ [Schmidt-Schauß, Smolka 91].

$$
\begin{array}{rcl}
\mathit{Lampe} & \sqsubseteq & \mathit{Komponente} \sqcap \neg \mathit{Motor} \\
\mathit{Stecker} & \sqsubseteq & \mathit{Komponente} \sqcap \neg \mathit{Motor} \sqcap \\
 & & \neg \mathit{Lampe} \\
\mathit{Saugrohr} & \sqsubseteq & \mathit{Komponente} \sqcap \neg \mathit{Motor} \sqcap \\
 & & \neg \mathit{Lampe} \sqcap \neg \mathit{Stecker} \\
\mathit{Mixstab} & \sqsubseteq & \mathit{Komponente} \sqcap \neg \mathit{Motor} \sqcap \\
 & & \neg \mathit{Lampe} \sqcap \neg \mathit{Stecker} \sqcap \\
 & & \neg \mathit{Saugrohr} \\
\mathit{Gerät} & \sqsubseteq & \forall \mathit{Teil}\colon \mathit{Komponente} \sqcap \mathit{Ding} \sqcap \neg \mathit{Komponente} \\
\mathit{E\text{-}Gerät} & \doteq & \mathit{Gerät} \sqcap \exists \mathit{Teil}\colon \mathit{Stecker} \\
\mathit{Staubsauger} & \doteq & \mathit{E\text{-}Gerät} \sqcap \forall \mathit{Teil}\colon \mathit{Komponente} \sqcap \\
 & & \exists \mathit{Teil}\colon \mathit{Motor} \sqcap \exists \mathit{Teil}\colon \mathit{Lampe} \sqcap \\
 & & \exists \mathit{Teil}\colon \mathit{Saugrohr} \\
\mathit{Mixer} & \doteq & \mathit{E\text{-}Gerät} \sqcap \forall \mathit{Teil}\colon \mathit{Komponente} \sqcap \\
 & & \exists \mathit{Teil}\colon \mathit{Motor} \sqcap \exists \mathit{Teil}\colon \mathit{Lampe} \sqcap \\
 & & \exists \mathit{Teil}\colon \mathit{Mixstab}
\end{array}
$$

Die neu aufgetretenen Operatoren „$\sqsubseteq$“ und „$\doteq$“ bedeuten „partielle Definition“ bzw. „vollständige Definition“. Zum Beispiel ist eine *Lampe* partiell dadurch definiert, daß sie eine *Komponente* ist, die allerdings kein *Motor* ist, während ein *E-Gerät* vollständig dadurch definiert ist, daß es ein *Gerät* ist und einen *Stecker* besitzt. Die oben eingeführten Begriffe, die die *Terminologie* unserer Anwendungsdomäne bilden, können wir jetzt aufgrund ihrer Definitionen in einer Spezialisierungshierarchie, auch *Subsumptionshierarchie* genannt, anordnen.

Um über konkrete Objekte zu reden, erweitern wir unsere Repräsentationssprache um Mittel zur Beschreibung von Objekten. Wir schreiben $o \in C$, wenn das Objekt o zum Begriff C gehört. Also z.B. $s \in \mathit{Staubsauger}$, falls s ein *Staubsauger* ist. $\langle x, y \rangle \in R$ schreiben wir, wenn das Objekt y ein Rollenfüller für die Rolle R des Objektes x ist. $\langle s, st \rangle \in \mathit{Teil}$ bedeutet z.B., daß das Objekt st ein *Teil* von s ist.

Die Konstruktionsaufgabe könnte jetzt folgendermaßen formalisiert werden. Gegeben unsere obige Terminologie sowie die Aussage

$$s \in \mathit{Staubsauger},$$

bedeutet unter der gegebenen intuitiven „Semantik“ unseres Repräsentationsformalismus,

daß es andere Objekte X_1, X_2, X_3, X_4 geben muß, so daß gilt:

$$\begin{array}{lcllcl} X_1 & \in & \mathit{Stecker} & \langle s, X_1 \rangle & \in & \mathit{Teil} \\ X_2 & \in & \mathit{Motor} & \langle s, X_2 \rangle & \in & \mathit{Teil} \\ X_3 & \in & \mathit{Lampe} & \langle s, X_3 \rangle & \in & \mathit{Teil} \\ X_4 & \in & \mathit{Saugrohr} & \langle s, X_4 \rangle & \in & \mathit{Teil}. \end{array}$$

Dies ist aber genau die Bauanleitung für unseren Staubsauger!

Andersherum funktioniert es auch. Gegeben die vier Objekte mit den genannten Zusicherungen sowie der Aussage $s \in \mathit{Gerät}$ führt zu der Aussage $s \in \mathit{E\text{-}Gerät}$ und $s \in \mathit{Staubsauger}$. Mit anderen Worten, falls eine Beschreibung eines Objektes und seiner Teile mit der Definition eines Begriffs übereinstimmt, können wir dieses Objekt dem Begriff zuordnen. Ein Teil dieser Aufgabe ist offensichtlich identisch mit der Aufgabe eine Spezialisierungshierarchie der Begriffe aufzubauen. Wenn wir z.B. erfahren wollen, zu welchem Begriff das Objekt c gehört, daß beschrieben ist als

$$c \in (\mathit{Gerät} \sqcap \exists\, \mathit{Teil}\colon \mathit{Stecker}),$$

so stellen wir fest, daß wir lediglich die Oberbegriffe der Konzeptbeschreibung finden müssen.

Wenn wir versuchen, die Diagnoseaufgabe zu lösen, bemerken wir, daß verschiedene Dinge dieses ersteinmal verhindern. Zum einen brauchen wir neue Begriffssymbole. Dann können wir nicht ohne weiteres den Begriff *Angeschlossen* in unserer Sprache repräsentieren. Dazu wäre so etwas wie eine *inverse Rolle* und die Zusicherung der *Identität von Rollenfüllern* erforderlich. Wir wollen das an dieser Stelle aber nicht weiter verfolgen, sondern versuchen die Bedeutung unseres Repräsentationsformalismus und die Dienste des Repräsentationssystems präziser zu fassen.

1.1.4.2 Semantik von Repräsentationsformalismen

Ein wesentlicher Schwachpunkt von frühen Repräsentationsformalismen ist es, daß die Bedeutung dieser Formalismen nur intuitiv und/oder durch das Systemverhalten gegeben ist. Beispielsweise kritisierte Woods [Woods 75], daß es keine Semantik der *semantischen Netze* gäbe. Dies führte dazu, daß relativ viele (fruchtlose) Debatten über die Ausdrucksfähigkeit von Formalismen geführt wurden. Schlimmer noch, es war auch nicht klar, *was* denn nun tatsächlich mit Hilfe eines Repräsentationsformalismus repräsentiert werden kann.

Eine Lösung dieses Problems ist es, eine *formale Semantik* anzugeben. Dabei gibt es natürlich erst einmal beliebige Möglichkeiten. Als Standardwerkzeug dafür haben sich jedoch logische Methoden bewährt. Wir haben damit also die Situation, daß zwar Logik

selbst nicht als Repräsentationsformalismus benutzt wird, daß die verschiedenen Repräsentationsformalismen aber jeweils eine logische Semantik besitzen. Damit ist es möglich,

- die Bedeutung von Repräsentationsformalismen zu kommunizieren,
- die intendierte Semantik mit der formalen Spezifikation zu vergleichen [Hanks, McDermott 87],
- qualifizierte Diskussionen über die Ausdrucksfähigkeit von Repräsentationsformalismen zu führen [Baader 90a],
- verschiedene Repräsentationsformalismen miteinander zu vergleichen [Nebel, Smolka 91],
- das Systemverhalten mit der formalen Spezifikation der Repräsentationssprache zu vergleichen [Touretzky 86],
- die Berechenbarkeitseigenschaften von Repräsentationsformalismen zu bestimmen [Levesque, Brachman 87],
- Inferenzalgorithmen zu entwickeln und deren *Korrektheit* und *Vollständigkeit* zu beweisen [Schmidt-Schauß, Smolka 91; Hollunder et al. 90].

In unserem Fall können wir die Semantik unseres kleinen Repräsentationsformalismus durch eine direkte Übersetzung in die Prädikatenlogik erster Stufe angeben. Jedes Begriffssymbol B entspricht einem einstelligen Prädikat $B(x)$. Jedes Rollensymbol entspricht einem zweistelligem Prädikat $R(x, y)$. Ein komplexer Begriffsausdruck C entspricht einer Formel $C(x)$ mit einer freien Variablen x, die sich wie folgt ergibt:

Repräsentationssprache	Logik
$C \sqcap C'$	$C(x) \wedge C'(x)$
$C \sqcup C'$	$C(x) \vee C'(x)$
$\neg C$	$\neg C(x)$
$\forall R\colon C$	$\forall y\colon R(x, y) \Rightarrow C(y)$
$\exists R\colon C$	$\exists y\colon R(x, y) \wedge C(y)$

Die Definitionsoperatoren werden als universeller Abschluß einer Implikation bzw. eines Bikonditionals interpretiert:

Repräsentationssprache	Logik
$B \sqsubseteq C$	$\forall x\colon B(x) \Rightarrow C(x)$
$B \doteq C$	$\forall x\colon B(x) \Leftrightarrow C(x)$

Schließlich werden die Beschreibungen von Objekten als Prädikationen über Konstanten aufgefaßt:

Repräsentationssprache	Logik
$a \in C$	$C(a)$
$\langle a, b \rangle \in R$	$R(a, b)$

Nach dieser Angabe einer formalen Semantik sollte klar sein, daß die in dem vorhergehenden Abschnitt angegebenen Schlußfolgerungen sich als logische Folgerungen ergeben. D.h. die Dienste „Konstruktion“und „Erkennung“ lassen sich als Spezialfälle des logischen Folgerungsbegriffs auffassen. Dies ist nicht notwendigerweise immer der Fall. Es lassen sich eine Menge von Fällen denken, bei denen zwar der Repräsentationsformalismus eine an die Prädikatenlogik angelehnte Form der Semantik besitzt, bei denen wir aber an Diensten interessiert sind, die nicht dem klassischen logischen Folgerungsbegriff entsprechen. Änderungen von Wissensbasen, bei denen wir etwas löschen oder modifizieren wollen, fallen z.B. in diese Kategorie, ebenso wie das Schließen unter unsicheren Annahmen.

Schließlich ergibt sich, daß unsere Repräsentationssprache einen Teil der „epistemologischen“ Schemata abdeckt, die wir weiter oben aufgestellt haben. Welche Unterschiede sich ergeben, insbesondere in Hinblick auf die Berechenbarkeitseigenschaften, werden wir weiter unten diskutieren.

1.1.4.3 Inferenzalgorithmen

Eine Repräsentationssprache gibt uns die Ausdrucksmittel, um das Wissen einer Anwendungsdomäne zu beschreiben. Die mit der Sprache assoziierte Semantik verrät uns, welche formale Bedeutung ein gegebener Repräsentationsausdruck hat und wie die Dienste eines Repräsentationssystems zu interpretieren sind. Was uns jetzt noch fehlt, ist die Mechanisierung, die *Algorithmisierung*, der Dienste eines Repräsentationssystems. Wie muß ein Algorithmus aussehen, der *Staubsauger* erkennt oder, allgemeiner, Objekte Begriffen zuordnet? Wie muß ein Algorithmus aussehen, der *Staubsauger* konstruiert oder, allgemeiner, der die Bedingungen für die Zugehörigkeit eines Objekts zu einem Begriff berechnet? Wie sieht ein Algorithmus aus, der eine Spezialisierungshierarchie einer Menge von Begriffen berechnet? Und angenommen, wir haben auf alle diese Fragen Antworten gefunden, wie können wir zeigen, daß die Algorithmen tatsächlich die in sie gesetzten Hoffnungen erfüllen?

Wir wollen mit der letzten Frage beginnen. Angenommen, wir haben einen *Entscheidungsalgorithmus* (ein Algorithmus, der „ja“ oder „nein“ ausgibt) spezifiziert, von dem wir glauben, daß er entscheidet, ob ein beliebiges Objekt a zum Begriff C gehört, dann müssen wir beweisen, daß

- immer, wenn der Algorithmus eine positive Antwort gibt, tatsächlich $a \in C$ gilt,
- immer, wenn $a \in C$ gilt, der Algorithmus eine positive Antwort gibt und

- der Algorithmus tatsächlich immer terminiert.

Die erste Bedingung heißt *Korrektheit*, die zweite *Vollständigkeit* des Algorithmus. Die dritte Bedingung ist eine notwendige Bedingung dafür, daß es sich überhaupt um einen Algorithmus handelt. Wird die dritte Bedingung nicht erfüllt, sprechen wir von einer *Prozedur*. Allerdings kann es vorkommen, daß nicht alle drei Bedingungen zusammen erfüllt werden können. Schlußfolgerung in der Prädikatenlogik erster Stufe ist beispielsweise unentscheidbar, d.h., daß man keinen *korrekten* und *vollständigen* Algorithmus angeben kann, der entscheidet, ob eine Aussage aus einer anderen folgt. Man kann allerdings *vollständige* und *korrekte* Prozeduren für dieses Problem angeben, z.B. *Resolution*. Offensichtlich können wir diese Eigenschaften nur dann untersuchen, wenn wir die Semantik des Repräsentationsformalismus formal angegeben haben. Außerdem ist es offensichtlich, daß wir einen Beweis nur dann führen können, wenn der Algorithmus kompakt genug ist, so daß ein Beweis der Korrektheit und Vollständigkeit auch tatsächlich möglich ist. Aus diesem Grund wählt man eine möglichst abstrakte Form bei der Angabe des Algorithmus.

Wir wollen beispielhaft für den oben angegebenen Repräsentationsformalismus einen abstrakten Algorithmus angeben, der *Subsumption zwischen zwei Begriffen entscheidet*, d.h. der bestimmt, ob ein Begriff spezieller als ein anderer ist. Obwohl dies ja nur einer der Dienste ist, für die wir uns interessieren, ist die Aufgabenstellung doch so beschaffen, daß dieser Algorithmus verallgemeinert werden kann, um auch die anderen Dienste zu realisieren.

Zuerst einmal fassen wir die Subsumptionsrelation etwas formaler. Ein Begriff „C wird in der Terminologie T von C' subsumiert“, in Symbolen $C \preceq_T C'$, genau dann, wenn jedes Objekt, das ein C ist, auch ein C' ist, also

$$C \preceq_T C' \quad \text{genau dann wenn} \quad T \models \forall x\colon C(x) \Rightarrow C'(x).$$

Als ersten Schritt zur Berechnung von $\preceq_T$ geben wir jetzt ein einfaches Verfahren an, um bei der Subsumptionsbestimmung von der Terminologie zu abstrahieren. Dazu formen wir die Terminologie T in eine andere Terminologie T^* um, in der der $\sqsubseteq$ Operator nicht mehr auftaucht. Zu diesem Zweck führen wir für jedes partiell definierte Begriffssymbol ein neues Begriffssymbol ein, daß auf der linken Seite der Definition konjunktiv hinzugenommen wird, also z.B.:

$$\begin{aligned} \mathit{Komponente} &\doteq \mathit{Ding} \sqcap \mathit{Komponente}^* \\ \mathit{Motor} &\doteq \mathit{Komponente} \sqcap \mathit{Motor}^* \\ \mathit{Lampe} &\doteq \mathit{Komponente} \sqcap \neg \mathit{Motor} \sqcap \mathit{Lampe}^*. \end{aligned}$$

Die neuen Symbole $\mathit{Komponente}^*, \mathit{Motor}^*, \mathit{Lampe}^*, \ldots$ bleiben innerhalb der Terminologie undefiniert. Wie man sich leicht klar macht, ändert man durch diesen Übergang von T nach

T^* nichts an der Bedeutung der ursprünglichen Begriffssymbole. Insbesondere bleibt die Subsumptionsbeziehung zwischen Begriffsausdrücken, die nur Symbole der ursprünglichen Terminologie verwenden, von dieser Transformation unberührt.

Wenn wir jetzt einen Begriffsausdruck C gegeben haben, so wollen wir mit $E(C)$ seine *Expansion* bezüglich der Terminologie T^* bezeichnen, wobei mit „Expansion" die Ersetzung von Begriffssymbolen durch ihre Definition gemeint ist, die so lange durchgeführt wird, bis nur noch undefinierte Begriffssymbole auftauchen. Beispielsweise ist $E(Motor)$:

$$Ding \sqcap Komponente^* \sqcap Motor^*.$$

Die Funktion $E(\cdot)$ ist natürlich nur dann wohl-definiert, wenn

1. jedes Begriffssymbol höchstens einmal auf der rechten Seite einer Definition auftaucht und
2. die Terminologie keine „Zyklen" enthält.

Dieses sind aber zwei Voraussetzungen, die man bei Repräsentationsformalismen der KL-ONE Familie normalerweise macht (siehe aber [Baader 90b; Nebel 91]).

Außerdem wollen wir mit $\preceq$ die Subsumptionsrelation in der *leeren* Terminologie, d.h. der Terminologie, die keine Definitionen enthält, bezeichnen. $C \preceq C'$ bedeutet also, daß die Formel $\forall x\colon C(x) \Rightarrow C'(x)$ allgemeingültig ist. Wollen wir jetzt berechnen, ob $C \preceq_T C'$ gilt, so können wir statt dessen auch berechnen, ob $E(C) \preceq E(C')$ gilt, wie man sich anhand der Semantik des Repräsentationsformalismus leicht klar macht. Wir haben somit die Subsumption innerhalb einer Terminologie auf Subsumption zwischen Begriffsausdrücken (in der leeren Terminologie) reduziert. Damit können wir uns jetzt darauf konzentrieren, einen Algorithmus für $\preceq$ zu entwerfen.

Eine erste Idee für solch einen Algorithmus könnte sein, die beiden Ausdrücke $E(C)$ und $E(C')$ strukturell miteinander zu vergleichen. Dies ist auch tatsächlich eine Möglichkeit, wie Levesque und Brachman [Levesque, Brachman 87] gezeigt haben. Sie haben für eine Teilmenge des oben angegebenen Formalismus einen vollständigen und korrekten Subsumptionsalgorithmus angeben, der auf strukturellem Vergleich basiert. Allerdings stellt sich heraus, daß die Verallgemeinerung dieses Vorgehens für mächtigere Formalismen, wie z.B. für den hier betrachteten Formalismus, aus mehreren Gründen nicht gangbar ist. Das Hauptproblem besteht darin, daß die *Vollständigkeit* des Algorithmus sich nur schwer beweisen läßt.

Wir wollen hier einen anderen Weg gehen, der an die *Constraint-Lösungsmethode* von Schmidt-Schauß und Smolka [Schmidt-Schauß, Smolka 91] angelehnt ist. Dazu machen wir uns zuerst klar, daß man Subsumption zwischen zwei Begriffsausdrücken auf *Inkonsistenz*

eines Begriffsausdruck reduzieren kann, wobei ein Begriffsausdruck *inkonsistent* oder auch *leer* genannt wird, wenn man damit keine Objekte beschreiben kann, oder formal, wenn der existentielle Abschluß der Übersetzung in die Prädikatenlogik, also $\exists x\colon C(x)$, unerfüllbar ist. Nun gilt folgende Äquivalenz:

$$C \preceq C' \quad \text{genau dann wenn} \quad C \sqcap \neg C' \text{ inkonsistent.}$$

Dies gilt, da die Negation einer allgemeingültigen Formel immer unerfüllbar ist (und umgekehrt) und ja folgende Beziehungen gelten:

$$\begin{array}{rlr} C \preceq C' & \text{genau dann wenn} & \models \forall x\colon C(x) \Rightarrow C'(x) \\ \neg\forall x\colon C(x) \Rightarrow C'(x) & \text{ist logisch äquivalent zu} & \exists x\colon C(x) \wedge \neg C'(x) \\ \exists x\colon C(x) \wedge \neg C'(x) \text{ unerfüllbar} & \text{genau dann wenn} & C \sqcap \neg C' \text{ inkonsistent.} \end{array}$$

Wenn wir also einen Algorithmus für Subsumption gefunden haben, können wir auch Inkonsistenz von Begriffsausdrücken entscheiden und umgekehrt.[16] Da Algorithmen für Inkonsistenz einfacher zu entwickeln und verifizieren sind, werden wir uns darauf konzentrieren, einen Inkonsistenzentscheidungsalgorithmus zu entwickeln. Die Idee dahinter ist relativ simpel. Man versucht auf allen möglichen Wegen ein Objekt zu konstruieren, das ein Beispiel für ein Objekt ist, das zu dem zu testenden Begriff gehört. Gelingt dies, ist der Begriffsausdruck konsistent. Gelingt dies nicht – und haben wir tatsächlich alle Möglichkeiten für die Konstruktion eines solchen Objekts ausgeschöpft – dann ist der Begriffsausdruck inkonsistent.

Um den Algorithmus möglichst einfach zu halten, werden wir den Begriffsausdruck vorher in eine Normalform, die sogenannte *Negationsnormalform*, überführen. Dies ist eine Form, in der der Negationsoperator nur direkt vor Begriffssymbolen steht. Ein beliebiger Begriffsausdruck kann in seine Negationsnormalform überführt werden, in dem man folgende Äquivalenzen ausnutzt:

$$\begin{array}{rcl} \neg(C \sqcap C') & \sim & \neg C \sqcup \neg C' \\ \neg(C \sqcup C') & \sim & \neg C \sqcap \neg C' \\ \neg(\neg C) & \sim & C \\ \neg(\forall R\colon C) & \sim & \exists R\colon \neg C \\ \neg(\exists R\colon C) & \sim & \forall R\colon \neg C. \end{array}$$

[16] Voraussetzung dafür ist, daß wir Begriffskonjunktion und Negation in unserer Sprache haben.

Um zu bestimmen, ob ein Ausdruck in Negationsnormalform inkonsistent ist, führen wir jetzt den Begriff eines *Constraints* ein. Ein Constraint ist entweder ein Paar der Art $(x{:}\,C)$ oder ein Tripel (xRy) mit der intuitiven Bedeutung, daß das Objekt x zum Begriff C gehört bzw. daß die beiden Objekte x und y in der R-Beziehung zueinander stehen. Eine Menge von Constraints wird *Constraint-System* genannt und im folgenden mit $\mathcal{C}$ bezeichnet. Gegeben ein Begriffsausdruck C, dann nennen wir das System $\{(x{:}\,C)\}$ ein *jungfräuliches* Constraint-System und die Variable x wird *Wurzelvariable* genannt. Ein Constraint-System heißt *erfüllbar*, wenn der existentielle Abschluß über die Konjunktion aller Constraints eine erfüllbare Formel ist, *unerfüllbar* sonst. Wir geben jetzt eine Menge von Transformationsregeln an, die die Erfüllbarkeitseigenschaft invariant lassen und entweder ein Objekt konstruieren, das alle Constraints erfüllt oder zeigen, daß dies nicht möglich ist [Hollunder et al. 90].

1. $\mathcal{C} \cup \{x{:}\,(C \sqcap C')\} \rightarrow_{\sqcap} \mathcal{C} \cup \{x{:}\,(C \sqcap C'), x{:}\,C, x{:}\,C'\}$, falls nicht schon $\{x{:}\,C, x{:}\,C'\} \subseteq \mathcal{C}$.
2. $\mathcal{C} \cup \{x{:}\,(\exists R{:}\,C)\} \rightarrow_{\exists} \mathcal{C} \cup \{x{:}\,(\exists R{:}\,C), xRy, y{:}\,C\}$, wobei y eine neue Variable ist, falls es nicht schon eine Variable z gibt, so daß $\{xRz, z{:}\,C\} \in \mathcal{C}$.
3. $\mathcal{C} \cup \{x{:}\,(\forall R{:}\,C), xRy\} \rightarrow_{\forall} \mathcal{C} \cup \{x{:}\,(\forall R{:}\,C), xRy, y{:}\,C\}$, falls nicht schon $(y{:}\,C) \in \mathcal{C}$.

Es sollte klar sein, daß ein Constraint-System $\mathcal{C}$ nach der Anwendung einer der obigen Regeln erfüllbar ist, genau dann, wenn das System vorher erfüllbar war.

Was uns jetzt noch fehlt, ist eine Regel für die Begriffsdisjunktion. Im Gegensatz zu den anderen drei Regeln muß diese jedoch *nicht-deterministisch* sein. Um zu prüfen, ob $x{:}\,(C \sqcup C')$ erfüllbar ist, müssen wir entweder zeigen, daß $x{:}\,C$ oder $x{:}\,C'$ erfüllbar ist. Die entsprechende Regel lautet daher:

4. $\mathcal{C} \cup \{x{:}\,(C \sqcup C')\} \rightarrow_{\sqcup} \begin{cases} \mathcal{C} \cup \{x{:}\,(C \sqcup C'), x{:}\,C\} \text{ oder} \\ \mathcal{C} \cup \{x{:}\,(C \sqcup C'), x{:}\,C'\}, \end{cases}$
   falls nicht schon $(x{:}\,C) \in \mathcal{C}$ oder $(x{:}\,C') \in \mathcal{C}$.

Falls ein Constraint-System erfüllbar ist, dann muß wenigstens eine der beiden Möglichkeiten der Regel $\rightarrow_{\sqcup}$ zu einem erfüllbaren Constraintsystem führen. Umgekehrt, falls ein Constraint-System unerfüllbar ist, wird das System nach der Anwendung von $\rightarrow_{\sqcup}$ unerfüllbar sein, egal welche Möglichkeit wir gewählt haben.

Wenden wir diese vier Regeln auf ein jungfräuliches Constraint-System solange an, bis keine Regel mehr anwendbar ist, erhalten wir ein *vollständiges* Constraint-System, und ob ein vollständiges Constraint-System erfüllbar ist, kann man sehr einfach feststellen. Ein Paar von Constraints der Art $(x{:}\,B), (x{:}\,\neg B)$, wobei B ein Begriffssymbol ist, wollen wir *elementaren Widerspruch* nennen. Offensichtlich ist ein Constraint-System nicht erfüllbar, falls es einen elementaren Widerspruch enthält. Gibt es keinen elementaren Widerspruch, dann

können wir das vollständige Constraint-System benutzen, um eine Struktur aufzubauen, so daß die Wurzelvariable ein Objekt bezeichnet, das zu dem ursprünglichen Begriffsausdruck gehört.

Mit den obigen Aussagen folgt daraus, daß der Begriffsausdruck C inkonsistent ist, genau dann, wenn jede mögliche Vervollständigung des jungfräulichen Constraint-Systems $\{x{:}\,C\}$ einen elementaren Widerspruch enthält.

Obwohl dieses Verfahren erst einmal nicht wie ein Algorithmus ausschaut, kann man sich leicht klar machen, daß es tatsächlich immer terminiert und damit ein Entscheidungsalgorithmus ist. Die wesentliche Idee bei einem Terminierungsbeweis für das oben angegebene Verfahren ist, daß die Transformationsregeln komplexe Ausdrücke in einfachere Ausdrücke zerlegen. Die nicht-deterministische Komponente kann man leicht beseitigen, indem man jeweils beide Möglichkeiten der Disjunktionsregel durchspielt. Natürlich wird man bei einer konkreten Implementierung nicht unbedingt ein Constraint-System und Regelanwendungen benutzen, um den Inkonsistenztest zu realisieren, da dieses nicht sehr effizient ist. Statt dessen wird man vermutlich andere Möglichkeiten, wie z.B. rekursiven Abstieg über die Struktur des Begriffsausdrucks, benutzen. Für die Verifikation des Verfahrens ist die obige abstrakte Form eines Algorithmus jedoch ideal.

Am Schluß dieses Abschnitts wollen wir noch einmal die Entwicklung unseres Inferenzalgorithmus zusammenzufassen. Wir waren ausgegangen von der Frage, wie wir $C \preceq_T C'$ bestimmen können. Dieses Problem haben wir reduziert auf die Berechnung von $E(C) \preceq E(C')$ – auf den Subsumptionstest von Begriffsausdrücken in der leeren Terminologie. Subsumption in der leeren Terminologie haben wir auf den Test der Inkonsistenz der Negationsnormalform von $E(C) \sqcap \neg E(C')$ reduziert, was wiederum auf die Unerfüllbarkeit von Constraint-Systemen reduziert wurde. Das letzte Problem wurde schließlich gelöst, in dem wir eine Menge von vier Transformationsregeln angegeben haben, die angewendet auf auf ein jungfräuliches Constraint-System entweder immer zu einem elementaren Widerspruch führen und damit die Inkonsistenz beweisen, oder aber zu einem vollständigen Constraint-System ohne elementaren Widerspruch, das es uns erlaubt, eine Struktur zu konstruieren, die den ursprünglichen Konzeptausdruck erfüllt.

1.1.4.4 Berechenbarkeitseigenschaften

Man kann sich an dieser Stelle fragen, was denn nun besonderes an unserem Repräsentationsformalismus ist. Offensichtlich ist es doch möglich, ihn vollständig in Prädikatenlogik einzubetten. Da wir wissen, wie vollständige und korrekte Deduktionsverfahren für Prädikatenlogik aussehen, könnte man jetzt hergehen und solche Verfahren anwenden, statt

sich den Kopf über spezielle Inferenzalgorithmen zu zerbrechen. Tatsächlich geht ein solches Argument jedoch am Kern der Sache vorbei. Um ein Beispiel aus der Informatik zu geben, würde man ja auch die Untersuchung der Eigenschaften von kontextfreien Sprachen und die Entwicklung von Parsingalgorithmen für spezielle kontextfreie Sprachen nicht mit dem Argument abtun, daß es sich hierbei nur um Spezialfälle von kontextsensitiven Sprachen handelt.

Was uns an dieser Stelle interessiert, ist also die spezielle Struktur von Repräsentationsformalismen, die unter Umständen *effizientere* Verfahren zuläßt als allgemeine Beweisverfahren für die Prädikatenlogik erster Stufe. Unter Umständen sind diese Verfahren so beschaffen, daß wir gewisse *Garantien* für die Inferenzverfahren abgeben können, z.B., daß wir für jede Eingabe nach endlicher Zeit eine Antwort bekommen oder sogar schon nach einer Zeit, die durch eine Funktion in der Länge der Eingabe begrenzt ist.

Als Beispiel können wir hier unseren Repräsentationsformalismus betrachten, der offensichtlich die Eigenschaft hat, daß Subsumption *entscheidbar* ist, da wir ja einen Entscheidungsalgorithmus angeben können. Das heißt, wir haben gegenüber der Prädikatenlogik tatsächlich etwas gewonnen – auf dem Gebiet der Berechenbarkeitseigenschaften. Allerdings haben wir natürlich auch etwas von der Ausdrucksfähigkeit der Prädikatenlogik eingebüßt. Solche Betrachtungen gehören zu einer der wichtigsten Aufgaben der Wissensrepräsentation, nämlich den richtigen Kompromiß zwischen Ausdrucksfähigkeit und Berechenbarkeitseigenschaften zu finden [Levesque, Brachman 87].

Hierbei ist zu beachten, daß die spezielle syntaktische Struktur der Repräsentationssprache nur eine relativ untergeordnete Rolle spielt. Würden wir statt der linearen Form unseres Repräsentationsformalismus eine graphische Form wählen, die an semantischen Netzen orientiert ist, so mag man einen Vorteil bei der *Präsentation* des Wissens haben und einen heuristischen Vorteil beim Nachdenken über repräsentiertes Wissen und Inferenzverfahren zur Verarbeitung des Wissens haben. Solange aber die *Ausdrucksfähigkeit* von Formalismen gleich ist, das heißt salopp gesprochen, daß alles, was wir in dem einen Formalismus ausdrücken können, mit ungefähr dem gleichem Aufwand auch in dem anderen Formalismus ausdrückbar ist – ein Punkt, den man mit Hilfe der formalen Semantik überprüfen kann [Baader 90a] – solange werden auch Inferenzverfahren ungefähr den gleichen Aufwand benötigen.

Nun ist die Unterscheidung in entscheidbare und unentscheidbare Probleme aber sehr grob und meist nicht sehr aussagekräftig. Viele *im Prinzip* entscheidbare Probleme können nichtsdestoweniger sehr kompliziert sein – so kompliziert, daß ein Algorithmus mehr Laufzeit benötigt als die zu erwartende Lebensdauer der Erde beträgt (auch wenn wir von sehr viel schnelleren Computern ausgehen). Wenn wir z.B. Brettspiele wie Schach, Go oder

Dame betrachten, so ist das Problem, den besten Zug zu finden, im Prinzip entscheidbar, da es nur endlich viele Brettkonfigurationen gibt und somit auch nur endlich viele Spielverläufe. Die Anzahl der möglichen Brettkonfigurationen ist aber so groß, daß diese prinzipielle Entscheidbarkeit keine Relevanz für die Praxis besitzt.

Eine feinere Unterteilung zur Beurteilung der Berechenbarkeitseigenschaften bietet die Komplexitätstheorie [Garey, Johnson 79]. Man unterteilt hierbei die Menge aller Probleme in solche, für die Algorithmen existieren, deren Laufzeit in allen Fällen durch eine polynomiale Funktion in der Länge der Eingabe nach oben abgeschätzt werden können und solche, bei denen keine Algorithmen mit dieser Eigenschaft existieren. Die ersteren werden als *handhabbare* Probleme bezeichnet. Der Grund für diese Art der Unterteilung liegt darin, daß bei man bei handhabbaren Problemen in der Regel davon ausgehen kann, daß auch relativ große Probleme noch in vernünftiger Zeit gelöst werden können, insbesondere, wenn der Grad des Polynoms niedrig ist (≤ 3). Bei nicht-handhabbaren Problemen steigt die notwendige Rechenzeit jedoch meist so schnell, daß nur relativ kleine Eingaben in vernünftiger Zeit bearbeitet werden können.

Man kann sich die praktischen Auswirkungen des Unterschieds zwischen handhabbaren und nicht-handhabbaren Problemen leicht vor Augen führen, wenn man die beiden Funktionen $f(n) = n^2$ und $g(n) = 2^n$ betrachtet.

n	$f(n)$	$g(n)$
10	100	1024
20	400	10^6
30	900	10^9
40	1600	10^{12}
50	2500	10^{15}

Angenommen, diese Funktionen beschreiben das Laufzeitverhalten zweier Algorithmen in Abhängigkeit von der Eingabegröße, und angenommen, die zugrundeliegende Zeiteinheit sind msec., dann sieht man anhand der obigen Tabelle schnell, daß man im Falle der exponentiellen Laufzeitfunktion Eingaben der Größe 10 noch in einer Sekunde behandeln kann, daß bei Eingaben der Größe 30 die Laufzeit aber völlig indiskutabel ist (277 Stunden). Dagegen können bei der polynomialen Laufzeitfunktion Eingaben der Größe 30 innerhalb einer Sekunde bearbeitet werden und selbst bei einer dreifach größeren Eingabe liegt die Laufzeit immer noch im Rahmen des Möglichen (10 Sekunden). Auch die Weiterentwicklung der Rechnertechnologie oder massive Parallelverarbeitung kann an dieser Stelle nicht viel weiter helfen, wie man sich schnell klar machen kann.

Natürlich wäre es wünschenswert, daß die Inferenzprobleme von Repräsentationsformalismen handhabbar sind, daß man also Inferenzalgorithmen angeben kann, die ein polyno-

miales Laufzeitverhalten haben. Levesque und Brachman [Levesque, Brachman 87] gehen sogar so weit, daß sie fordern, jeder Repräsentationsformalismus müsse diese Eigenschaft haben. Dies schränkt jedoch die Ausdrucksfähigkeit und/oder die Art der möglichen Dienste sehr radikal ein, so daß diese Forderung nicht immer eingehalten werden kann [Doyle, Patil 91b].

Wir wollen an dieser Stelle wieder unsere Repräsentationssprache und die Berechnung der Subsumptionsrelation als Beispiel heranziehen. Wenn wir den Constraint-Lösungsalgorithmus, der im letzten Abschnitt entwickelt wurde, analysieren, so stellen wir fest, daß er einige Stellen enthält, die dazu führen, daß im schlechtesten Fall die Laufzeit (und der notwendige Speicherplatz) exponentiell mit der Größe der Eingabe wächst. Natürlich stellt sich sofort die Frage, ob man es nicht besser machen könnte – ob das Subsumptionsproblem handhabbar ist und wir lediglich einen schlechten Algorithmus entworfen haben.

Bei der Suche nach einer Antwort stellt man fest, daß auf der einen Seite ein Algorithmus mit polynomialem Laufzeitverhalten nicht zu finden ist und auf der anderen Seite man aber nicht beweisen kann, daß es keinen solchen Algorithmus gibt. Diese Situation ist typisch für ganze Klassen von Problemen. Man hat unabhängig von diesem Dilemma aber Charakterisierungen gefunden, um diese Probleme von ihrer Berechenbarkeitskomplexität her zu beschreiben und nimmt an, daß diese Probleme nicht in polynomialer Zeit lösbar sind.

Eine dieser Charakterisierung erfolgt mit Hilfe von *nicht-deterministischen* Maschinenmodellen. Diese Maschinen kann man sich vorstellen als normale Computer, die an bestimmten Punkten nicht-deterministisch eine Entscheidung fällen. Man definiert dann, daß ein Problem gelöst wird, falls es eine Folge von solchen nicht-deterministischen Entscheidungen gibt, die zur Lösung führen. Dies entspricht z.B. einem PROLOG-Interpreter, der an jedem Entscheidungspunkt die „richtige“ Entscheidung trifft, und der deshalb nie zurücksetzen muß. Aufbauend auf dieses neue Maschinenmodell kann man die Klasse der Probleme definieren, die auf nicht-deterministischen Maschinen in polynomialer Zeit gelöst werden können. Diese Klasse von Problemen, die auch mit NP bezeichnet wird, ist offensichtlich eine Oberklasse der weiter oben eingeführten Klasse der handhabbaren Probleme, die mit P bezeichnet wird. Das oben beschriebene Dilemma, daß wir weder die Existenz eines Algorithmus mit polynomialer Laufzeit noch das Gegenteil beweisen können, findet sich an dieser Stelle wieder als Frage danach, ob P=NP oder P$\neq$NP.

Wie oben schon erwähnt wurde, ist diese Frage noch nicht beantwortet worden. Man ist allerdings in der Lage, *schwierigste* Probleme innerhalb der Klasse NP zu identifizieren, die sogenannten *NP-vollständigen Probleme*. Diese Probleme haben die Eigenschaft, daß sie alle dann, und nur dann, in polynomialer Zeit auf deterministischen Maschinen gelöst

werden können, wenn P=NP ist. Da man davon ausgeht, daß dies nicht der Fall ist, werden die NP-vollständigen Probleme als nicht-handhabbar angesehen.

Wenn man jetzt den Inferenzalgorithmus noch einmal anschaut, stellt man fest, daß er selbst auf einer nicht-deterministischen Maschine unter Umständen mehr als polynomial viel Zeit verbraucht. Das Zusammenspiel der $\rightarrow_\exists$ und der $\rightarrow_\forall$ Regel kann dazu führen, daß exponentiell viele Variablen eingeführt werden. Man kann in der Tat zeigen, daß das Subsumptionsproblem in unserem Formalismus zu den schwierigsten Problemen einer noch umfassenderen Klasse als NP gehört, nämlich der Klasse PSPACE [Schmidt-Schauß, Smolka 91; Schild 91]. Die Probleme dieser Klasse sind dadurch charakterisiert, daß der Speicherplatzbedarf höchstens polynomial mit der Eingabegröße wächst.

Diese sehr präzise Charakterisierung der Komplexität des Subsumptionsproblems macht zweierlei deutlich. Erstens wissen wir jetzt, daß wir tatsächlich keinen im Prinzip besseren Algorithmus als den oben angegebenen finden können.[17] Zweitens ist jetzt klar, daß wir nicht damit rechnen können, daß wir ein Repräsentationssystem bauen können, das unter allen Umständen eine korrekte Antwort in vernünftiger Zeit liefert.

Es gibt verschiedene Methoden, um mit diesem Problem fertig zu werden. Die einfachste ist, das Komplexitätsproblem zu ignorieren, und zu hoffen, daß der Benutzer keine zu großen Begriffsausdrücke eingibt oder daß die Fälle, die zu exponentieller Laufzeit führen, nicht auftreten. Diese Vogel-Strauß-Strategie hat jedoch den offensichtlichen Nachteil, daß wir keine Aussagen über die Performanz von Repräsentationssystemen machen können, und dem Benutzer keine Garantien für das Funktionieren des Systems geben können.

Eine Variation dieses Themas ist die Analyse der Ursachen der Komplexität mit der Absicht, die Fälle, die sich in polynomialer Zeit behandeln lassen, zu identifizieren. Dies kann sich auf die Struktur des repräsentierten Wissen beziehen [Cheeseman et al. 91] oder auf Teilmengen der Repräsentationssprache. Solche Analysen helfen, den Rahmen abzustecken, indem man Antworten in polynomialer Zeit von einem Repräsentationssystem erwarten kann, d.h. man kann für bestimmte Fälle die Garantie geben, daß das Repräsentationssystem funktionieren wird.

Schließlich kann man die Vollständigkeits- und/oder die Korrektheitsforderung aufgeben und *Approximationsalgorithmen* entwerfen, die immer in polynomialer Zeit eine Antwort finden, die aber nicht notwendig korrekt oder vollständig sind. In diesem Fall ist es aber notwendig, formale und/oder intuitive Kriterien zu geben, die das Antwortverhalten des

[17] Besser in dem Sinne, daß man mit polynomialer Laufzeit auskommt. Es kann durchaus sein, daß ein anderer Algorithmus in allen (oder fast allen) Fällen polynomial besser als der angegebene ist. Wegen des extremen Wachstums exponentieller Funktionen heißt dies jedoch nur, daß wir Eingaben, die etwas größer sind, in vernünftiger Zeit verarbeiten können.

Algorithmus beschreiben.

Eine natürliche Frage, die sich jetzt ergibt, ist, ob es vielleicht andere terminologische Logiken gibt, bei denen das Subsumptionsproblem eine geringere Komplexität besitzt, oder sogar in P liegt [Levesque, Brachman 87]. Eine terminologische Logik mit dieser Eigenschaft wird nicht für alle Probleme geeignet sein [Doyle, Patil 91b]. Aber wenn man mit einer solchen Repräsentationssprache auskommt, kann man *garantieren*, daß der Inferenzdienst immer in angemessener Zeit eine Antwort ausgibt. Donini und andere [Donini et al. 91a; Donini et al. 91b] haben dieses Problem sehr gründlich analysiert und die Komplexität des Subsumptionsproblems für fast alle denkbaren terminologischen Logiken präzise identifizieren können. Insbesondere waren sie in der Lage, die Grenze zwischen handhabbaren und nicht-handhabbaren terminologischen Logiken zu lokalisieren.

Allerdings muß man an dieser Stelle bemerken, daß diese Analysen sich immer auf die Subsumption von Begriffsausdrücken in der leeren Terminologie bezieht, also auf das Problem $E(C) \preceq E(C')$. Bezieht man die Möglichkeit, Begriffe mit Hilfe des Operators $\doteq$ zu definieren, in die Analyse mit ein, so stellt sich heraus, daß selbst wenn das Problem $E(C) \preceq E(C')$ in polynomialer Zeit lösbar ist, das Problem $C \preceq_T C'$ mindestens so schwierig wie die NP-vollständigen Probleme ist [Nebel 90b]. Dieses überraschende Ergebnis kann man damit erklären, daß der Ausdruck $E(C)$ unter ungünstigen Umständen sehr groß werden kann (exponentiell in der Größe der Terminologie). Allerdings scheint dieses im Normalfall nicht zu passieren, was vermutlich damit zusammenhängt, daß die „Definitionstiefe“ in Terminologien meist sehr beschränkt ist.

Dieses Ergebnis zeigt, daß die Analyse der Berechenbarkeitseigenschaften nicht einfach eine gradlinige Anwendung der Komplexitätstheorie auf die Repräsentationsformalismen und Inferenzdienste ist, sondern daß man sehr genau die Nebenbedingungen untersuchen muß, unter denen die Formalismen angewandt und die Dienste in Anspruch genommen werden.

Ein anderes Beispiel für dieses Phänomen ist die Analyse des Inferenzdienstes *temporale Projektion*. Hierbei handelt es sich darum, den Zustand der Welt nach dem Stattfinden einer Menge von Ereignissen vorherzusagen. Mit Hilfe dieses Inferenzdienstes ist es unter anderem möglich, festzustellen, ob ein von einem Planungssystem erzeugter Plan tatsächlich zu dem gewünschten Ergebnis führt – eine Aufgabe, die *Planvalidierung* genannt wird. Dean und Boddy [Dean, Boddy 88] haben nun gezeigt, daß temporale Projektion für partiell geordnete Ereignismengen NP-vollständig ist, auch wenn man sehr starke Restriktionen bei der Formulierung der Änderungsregeln in Kauf nimmt. Aus diesem Grund haben Dean und Boddy einen Approximationsalgorithmus für das Problem der temporalen Projektion entwickelt. Untersucht man nun das Planvalidierungsproblem, stellt sich allerdings heraus,

daß dieses Problem in den meisten Fällen handhabbar ist. Das heißt, daß unter der Annahme, daß Planvalidierung die wesentliche Motivation hinter temporaler Projektion ist, es günstiger ist, Planvalidierung selbst als Inferenzdienst anzubieten [Nebel, Bäckström 94].

Abschließend wollen wir noch bemerken, daß der Entwurf von Repräsentationsformalismen, die handhabbare Inferenzalgorithmen erlauben, nicht nur vom technischen Standpunkt her eine wichtige Aufgabe der Wissensrepräsentation ist [Levesque 86], sondern auch vom kognitiven Standpunkt her seine Berechtigung hat. Viele der kognitiven Fähigkeiten, wie Sprachverstehen und Bilderkennung, sind offensichtlich so realisiert, daß wir sie ohne großen Aufwand durchführen. Das heißt aber, daß wir *handhabbare* Methoden dafür einsetzen. Das heißt aber auch, daß unsere internen Repräsentationen solche Verfahren zulassen müssen, eine Schlußfolgerung, die Repräsentationen, die nicht-handhabbare Verfahren erfordern als *physikalisch nicht realisierbar* ausschließt [Levesque 88].

1.1.5 Systeme

Logische Analyse bzw. konzeptuelle Modellierung und Operationalisierung bilden im wesentlichen das Fundament, auf dem konkrete Anwendungen wissenbasierter Techniken aufgebaut werden. Es bleibt einem konkreten Repräsentationssystem vorbehalten, die Möglichkeiten der Darstellung dem Nutzer zur Verfügung zu stellen. Zuerst soll hier der noch abstrakte Begriff des „Nutzers“ kurz dargestellt werden.

In einem System wird die Funktionalität des Formalismus in Form von Dienstleistungen des Kerns und von zusätzlichen Dienstleistungen zur Verfügung gestellt. Beide Formen von Dienstleistungen sollen anschließend analysiert werden.

In gewissem Sinn verwendet dieser Abschnitt vornehmlich das Vokabular der Kerninformatik. Das ist leicht verständlich, bezieht sich ein großer Teil des Materials, das hier behandelt wird, auf konkrete Fragen der Realisierung, Sicherheit und Nutzbarkeit von großen Softwareprodukten. Die Informatik hat Qualitätsmerkmale und Verfahren entwickelt, die genau auf diese Fragen eingehen. Einige davon werden hier aufgenommen.

1.1.5.1 Typen von Systembenutzern

In ganz striktem Sinn sind Typen von Systemen (s.u.) von den beteiligten Nutzern kaum zu trennen. Experimentalsysteme werden zum größten Teil von den verantwortlichen Projektmitgliedern verwendet, die entweder für das System oder für die Domäne bzw. Aufgabe hoch qualifiziert sind. Dennoch kann man drei Tätigkeitsklassen bilden, die einen entsprechenden Personenkreis mit gewissen Anforderungen und Qualifikationen bestimmen:

Systemrealisierung: Dies sind alle Tätigkeiten, die mit der konkreten Implementation zusammenhängen. Merkmale solcher Tätigkeiten sind insbesondere Zugriffe auf Datenstrukturen, die allen anderen Nutzern unzugänglich sind, Eingriffe auf der Ebene unzugänglichen Codes und Einsatz von vorhandenen bzw. selbstgeschriebenen Mitteln der Fehlersuche und -beseitigung. Im Idealfall verfügen Systeme über eigene Umgebungen, die den beteiligten Personen diese Arbeiten ermöglichen.

Modellierung: Der zentrale Teil eines wissensbasierten Systems ist das operationalisierte Modell, das für die Lösung der gestellten Aufgabe verwendet wird. Aufbau, Wartung und Weiterentwicklung dieses Modells und seiner internen Repräsentation obliegt einem Personenkreis, der mit den Repräsentationkonstrukten soweit vertraut ist, wie das System sie sichtbar werden läßt. Die benötigten Hilfsmittel sind auf diese Personen und das äußere Erscheinungsbild des Repräsentationskerns abzustimmen.

Zwecknutzung: Letztlich dient das gesamte System mit dem operationalisierten Modell einem bestimmten Zweck. Dieser Zweck kann irgendwo zwischen einem abstrakten Begriff wie „Erkenntnisgewinn“ und einer konkreten Aufgabe wie „Kreditwürdigkeits-Analyse“ liegen. Prinzipiell wird man für eine solche Gruppe von Nutzern einen Zugang zum System anbieten, der von allen internen Strukturen und Abläufen abstrahiert und sich lediglich auf der Ebene von Domäne und Aufgabe abspielt.

1.1.5.2 Dienstleistungen

In diesem Abschnitt geht es darum, zu klären, was ein Benutzer von einem System erwarten darf, das einen oder mehrere Formalismen zur Repräsentation von Wissen enthält. Am Anfang sollen diese Dienstleistungen in drei Klassen eingeteilt werden, die wir dann separat analysieren werden:

Integration: Stehen mehrere unterschiedliche Repräsentationssysteme nebeneinander, so ist es eine Aufgabe des Systems, diese Einzelfunktionen so zu koordinieren, daß das dargestellte Wissen möglichst übergreifend für Inferenzen verwendet wird. Diese Dienstleistungen sind bereits im vorigen Abschnitt besprochen worden.

Einbindung: In vielen Fällen sind die Funktionen des Systems in einen größeren Softwarekomplex eingebunden, in dem etwa die Interaktion des Problemlösers mit dem Benutzer oder der Zugriff auf externe Komponenten (z.B. Datenbanken) vorgenommen werden. Solche Anwendungssoftware betrachtet den Repräsentationskern in erster Linie als eine Sammlung externer Dienstleistungen, die bei Bedarf abgerufen werden sollen.

Wartung: Die Inhalte der Wissensbasis müssen – wenigstens in bestimmten Phasen der Entwicklung der Anwendung – dem Benutzer zugänglich gemacht werden. Neben eventuell automatisierten Wartungsfunktionen (etwa Klassifikation) sind dem Benutzer Funktionen zur Inspektion und Modifikation zur Verfügung zu stellen.

Bei diesem Katalog handelt es sich um eine Zusammenstellung von Anforderungen, die geeignet ist, aus einem formalen Repräsentationsschema ein vollständiges System zu machen, das zwei Eigenschaften hat:

- Es ist *nützlich*, d.h. es stellt eine Performanz zur Verfügung, die zur Lösung eines gegebenen Problems notwendig ist.
- Es ist *integrierbar*, d.h. es kann alle anderen vorhandenen Informationsquellen des Benutzers zur Problemlösung heranziehen. Das Phänomen des „computational island", also eines alleinstehenden Systems ohne Bezug zu anderen vorhandenen Rechensystemen, ist eins der Haupthindernisse bei der Verbreitung wissensbasierter Techniken.

1.1.5.3 Klassifikation von Systemen

Zunächst sollen zwei Klassen von Systemen skizziert werden, die dem Anforderungskatalog in verschiedener Weise folgen. Mit beiden Klassen kann man in der Praxis konfrontiert werden.

Reife Systeme: Hierbei handelt es sich um Softwareprodukte, die nach anerkannten professionellen Entwurfs-, Konstruktions- und Evaluierungsmethoden entstanden sind. Sie genügen formalen Spezifikationen, die z.B. von Werkzeugen des *Software Engineering* überwacht werden. Sie sind ausgestattet mit Testsuiten, also Sätzen repräsentativer Testdaten; sie sind intern und extern wohldokumentiert und unterliegen einer Wartungs- und Erweiterungsstrategie des Herstellers.

Prototypen: Dies sind Softwareprodukte, die parallel zum Problemverständnis des „Schöpfers" entstehen. Sie enthalten die wesentlichen Algorithmen, Datenstrukturen und Benutzerschnittstellen, die zur Evaluierung der Korrektheit beim Aufbau großer Wissensbasen benötigt werden.

Ob Prototypen sinnvoll sind, ist jedoch umstritten. Eine der vertretenen Ansichten besteht darin, daß ohne formale Entwurfs- und Konstruktionsmethoden kein brauchbares nichttriviales Softwareprodukt entstehen kann. Dennoch gibt es wenigstens zwei Gründe für die Erstellung von Prototypen:

1. Für bestimmte Probleme sind die benötigten Algorithmen und Datenstrukturen nur theoretisch bekannt. Details sind nur durch prototypische Implementationen und ih-

ren Vergleich zu entwickeln.

2. Die notwendige „Lernphase“ am Anfang eines Projekts wird durch einen Prototypen unter Umständen drastisch verkürzt. Am Prototypen kann die Kommunikation zwischen Konstrukteur und späterem Nutzer schnell aufgenommen werden; Mißverständnisse und Unklarheiten werden schnell sichtbar und können bereits zu Beginn des Projekts beseitigt werden.

Grade im Bereich wissensbasierter Systeme wird dieser Weg häufig eingeschlagen, insbesondere wenn es sich um Systeme handelt, die neue Formalismen realisieren sollen. Es ist allerdings zu beobachten, daß Prototypen sich oft in Richtung auf reife Systeme entwickeln (sollen), was nicht ohne Probleme ist. Eins der Kennzeichen eines Prototyps besteht gerade darin, daß er bei Bedarf durch formale Spezifikationen ersetzt wird, die dann mit geeigneten Methoden, Personen und Werkzeugen zu einem reifen System entwickelt werden.

Es bleibt noch festzustellen, daß durch Einsatz moderner Entwicklungsumgebungen und geeigneter Programmiersprachen und -techniken (LISP, Prolog, Objekt-orientierte Programmierung) Prototypen mit beachtlicher Funktionalität schnell entstehen können. Dies kann soweit gehen, daß sich der potentielle Nutzer des Systems bereits mit einem guten Prototypen zufriedengibt und so das gesamte Projekt in ein nicht geringes „moralisches“ Dilemma stürzt: Soll man den Prototypen zur Benutzung freigeben oder auf ein mit nicht geringer Verzögerung entstehendes reifes System verweisen? Allerdings haben Prototypen oft eine Menge von unerwünschten Eigenschaften – z.B. Ineffizienz und mangelende Robustheit [Heinsohn et al. 94] – so daß Redesign und Neuimplementation unausweichlich sind.

1.1.5.4 Integration

Eins der Hauptmerkmale wissensbasierter Systeme ist die parallele Verwendung verschiedener Repräsentationsformalismen, die sich syntaktisch unterscheiden und auch verschiedene Dienstleistungen anbieten. Einer der Gründe, sich mehrerer solcher *Sub-Notationen* zu bedienen, besteht darin, daß sich diese Formalismen verschiedenen Wissenskategorien zuordnen lassen und so eine adäquate Formulierung des Wissens zulassen.

In der Praxis hat dies zur Folge, daß Wissenseinheiten entweder in verschiedenen Formalismen dargestellt werden (Redundanz), oder daß sie Bezug auf andere Wissenseinheiten in anderer Darstellung nehmen (Interaktion).

Beide Phänomene stellen Anforderungen an ein System, die über die reine Repräsentation und reine Inferenzen hinausgehen. Am Anfang wurde dargestellt, daß eine gemeinsame semantische Basis der verwendeten Repräsentationskonstrukte ein geeigneter Ansatz für

die Lösung des entstehenden Integrationsproblems ist.

Die technische Realisierung der Integration verschiedener Repräsentationskonstrukte beruht in erster Linie auf der Vorstellung, daß die vorhandene Aufteilung dem Nutzer möglichst verborgen bleiben soll. Das ist in der Praxis natürlich leichter gesagt als getan.

1.1.5.5 Hybridizität als Hauptproblem

Die Kombination verschiedener Repräsentationsformalismen zu einem geschlossenen System (Hybridizität) birgt wenigstens zwei Probleme in sich, die kurz beschrieben werden sollen.

Eins der Problem läßt sich unter dem Schlagwort *inferenzielle Lücken* beschreiben: Beide Subformalismen mögen isoliert betrachtet korrekt und vollständig implementiert sein; der kombinierte hybride Formalismus kann trotzdem unvollständig sein. In einem solchen Fall erfordert die Kombination der Formalismen zusätzlichen Aufwand, der in der Implementation übergreifender Inferenzmechnismen besteht. Auch wenn beide Subformalismen „gutartig“ sind, ist natürlich nicht in jedem Fall sichergestellt, daß solche übergreifenden Mechanismen überhaupt realisierbar sind und, wenn doch, daß sie effizient realisierbar sind.

Das andere Problem entsteht durch eine eventuelle *Redundanz* der Repräsentation. Das Hauptproblem beim Auftreten von Redundanz besteht – technisch gesprochen – darin, daß u.U. widersprüchliche Schlußfolgerungen gezogen werden können, oder daß Modifikationen desselben Inhalts an mehreren Stellen ausgeführt werden müssen.

Das Problem der Redundanz ist nicht prinzipiell neu und schon gar keines, das als genuines Problem wissensbasierter Systeme zu betrachten ist. Es tritt unter anderen Bezeichnungen – z.B. „Datenabhängigkeit“ – als Problem des *Datenbankentwurfs* auf. Der dort (immer noch) eingeschlagene Weg besteht darin, Redundanzen zu verbieten (Schlagwort: Normalformen) und die Pflege weitergehender Datenabhängigkeiten speziellen Prozeduren („Datenbank-Prozeduren“) überläßt.

Ausgedrückt mit dem Vokabular der Logik geht es bei der Behandlung dieses Problembereichs darum, eventuell auftretendende *globale Inkonsistenzen* innerhalb der Wissensbasis zu erkennen und eventuell zu beseitigen. Global sind diese Inkonsistenzen deshalb, weil sie zwischen Wissenseinheiten auftreten können, die in unterschiedlichen Subsystemen dargestellt sind. Ein kleines Beispiel mag das Problem erläutern:

Das vorliegende System möge bestehen aus

Subsystem 1: einer terminologisch/assertorischen Komponente, in der also Begriffe, Beziehungen zwischen Begriffen sowie Aussagen über Instanzen von Begriffen repräsen-

tiert werden, und

Subsystem 2: einem „normalen“ Hornklausel-Beweiser, namentlich Prolog.

In Subsystem 1 können Begrifflichkeiten folgender Art dargestellt sein:

- **Staubsauger** und **Mixer** sind **E-Geräte**
- **Staubi** ist ein **Staubsager**
- Jedes **E-Gerät** ist ein **Staubsauger** oder ein **Mixer**
- Nichts ist gleichzeitig **Staubsauger** und **Mixer**

In Subsystem 2 finden sich ähnliche Dinge mit einer Abweichung:

- **Staubsauger** und **Mixer** sind **E-Geräte**
- **Staubi** ist ein **Staubsauger**
- Jeder **Staubsauger** ist ein **Mixer**

Man überzeugt sich schnell davon, daß Subsystem 1 und Subsystem 2 auf die Frage

Ist **Staubi** *ein* **Mixer***?*

widersprüchliche Antworten liefern. Offenbar ist das globale logische Weltmodell in sich inkonsistent, obwohl beide Teilmodelle der jeweiligen Subsysteme sehr wohl konsistent sind. In diesem Beispiel kann man den Grund für die globale Inkonsistenz schnell lokalisieren; es ist aber nicht zu sehen, wie man globale logische Konsistenz im allgemeinen Fall garantieren kann. Ein tieferer Grund der Schwierigkeiten liegt natürlich darin, daß in beiden Subsystemen identische Symbole (Prädikate, Relationen und Konstanten) auftreten, die dann auf der Oberfläche mit identischen Interpretationen belegt werden. Dies natürlich umso mehr, wenn der Unterschied zwischen den Subsystemen dem Benutzer verborgen ist!

Es scheint drei Möglichkeiten zu geben, damit fertig zu werden:

1. *Ignorieren:* Man läßt die Dinge so problematisch wie sie sind und verläßt sich darauf, daß diese Probleme in der Praxis nicht auftreten. Insbesondere nimmt man in Kauf, daß Wissen redundant dargestellt ist und entfernt sich damit von der ursprünglichen Idee, daß verschiedene Subsysteme eindeutig epistemischen Kategorien zugeordnet werden.
2. *Verbieten:* Man stellt sicher, daß sich verschiedene Subsysteme keine Symbole teilen.
3. *Integrieren:* Man läßt ein gemeinsames Vokabular der Subsysteme zu, bildet aber alle sprachlichen Konstruktionen der jeweiligen Subsysteme in ein gemeinsames semantisches System ab, in dem die Widerspruchsfreiheit geprüft wird.

Die Idee der globalen Integration ist natürlich bestechend, stößt aber irgendwann an prinzipielle Grenzen der Entscheidbarkeit. In der Praxis wird man sich mit einem Kompromiß

zufriedengeben müssen, der im einen Extrem die erste Möglichkeit (Ignorieren), im anderen Extrem die zweite (Verbieten) wählt. Zwischen beiden Extremen sind praktikable Möglichkeiten angesiedelt, die allerdings von den beteiligten Subformalismen abhängen. Wesentlich bei solchen Einschränkungen ist grundsätzlich eine Anforderung:

> *Alle Restriktionen der in einem Subsystem verwendbaren Ausdrücke müssen syntaktisch verifizierbar sein.*

Der Grund ist einfach: Will man potentielle Widersprüche durch solche Einschränkungen ausschalten, so dürfen solche gefährlichen Ausdrücke im entsprechenden Subsystem niemals eingetragen werden. Die einzige Handhabe, das zu erkennen, liefert lediglich die syntaktische Struktur des Ausdrucks, insbesondere die darin verwendeten Symbole und ihre Position innerhalb des Ausdrucks.

Im Beispiel oben besteht eine einfache Möglichkeit z.B. darin, *Konstanten* (wie Staubi) genau einem der beiden Subsysteme zuzuordnen, man verbietet also übergreifende Beschreibungen von Individuen.

Wenn sich die Subformalismen genauer charakterisieren lassen, kann man auch weniger restriktive Einschränkungen formulieren. Im gegebenen Beispiel kann man etwa folgende syntaktische Einschränkungen vereinbaren:

- Einstellige Prädikatensymbole und zweistellige Relationssymbole, die in Subsystem 1 auftreten, dürfen in einer Hornklausel des Subsystems 2 nur in *negativer Form* auftreten. Auf Deutsch besagt dies, daß ein solches Symbol nur in der *Bedingung* einer Regel auftreten darf.
- Einstellige Prädikatensymbole und zweistellige Relationssymbole, die in Subsystem 1 nicht auftreten, sowie Symbole höherer Stelligkeit dürfen in Subsystem 2 ohne Einschränkungen repräsentiert werden.
- Individuenkonstanten dürfen zwischen beiden Systemen geteilt werden.

Zur Unifikation von Ausdrücken, in denen Symbole des Subsystems 1 auftreten, müssen dann natürlich die entsprechenden Fakten dieses Subsystems verwendet werden. Im Prinzip haben diese Vereinbarungen lediglich den Sinn, Schlüsse auf Zugehörigkeit zu einem Begriff oder Beteiligung an einer Rolle exklusiv dem Subsystem 1 zu überlassen.

1.1.5.6 Technische Realisierung

Hat man sich bei der Verbindung zwischen mehreren Repräsentationsformalismen einmal für eine Kopplungs-Strategie entschieden, so besteht „nur noch“ das Problem, diese Aufteilung so zu realisieren, daß

- die gewünschte Funktionalität, insbesondere die Aufgabenteilung zwischen den Komponenten, zustandekommt und
- der Benutzer davon möglichst nichts merkt.

Die einfachste Möglichkeit besteht darin, alle auftretenden Repräsentationskomponenten über eine abstrakte Schnittstelle mit dem Benutzer zu verbinden: Funktionen wie etwa **TELL** und **ASK** bieten einen ersten Anhalt. Die Zuordnung zu den einzelnen Formalismen wird – ähnlich wie bei *BABYLON* [Christaller et al. 89] – von einer eigenständigen Komponente vorgenommen, die *syntaktisch* entscheiden kann, welchem Formalismus die Eingabe zuzuordnen ist. Dies kann im obigen Beispiel in zwei Schritten geschehen:

1. Feststellen der „epistemischen Kategorie“ des Ausdrucks, also z.B. die Einordnung als *begrifflichen* Ausdruck oder als *Aussage*.
2. Sollte das zur Bestimmung des verantwortlichen Subsystems nicht ausreichen, so sind die syntaktischen Eigenschaften des Ausdrucks mit den gegebenen Einschränkungen zu vergleichen.

Daß das problematisch ist, sieht man am Beispiel: *Menschen sind Lebewesen*, dessen Zuordnung so nicht festgelegt werden kann, wenn der Begriff **Lebewesen** im Subsystem 1 noch nicht vorhanden ist. Hier wird man eine präzisere Kennzeichnung des Ausdrucks verlangen müssen.

1.1.5.7 Einbindung

Wissensbasierte Systeme unterscheiden sich in ihrem inneren Aufbau nicht prinzipiell von anderen Softwaresystemen: Sie realisieren bestimmte Algorithmen auf bestimmten Datenstrukturen, konstruieren neue Datenstrukturen und beantworten Fragen.

Unter dieser Prämisse wird man sich fragen, in welcher Form ein wissensbasiertes System

- Dienstleistungen anderer Softwarekomponenten (vornehmlich etwa Datenbanken) in Anspruch nehmen kann und
- seine Dienstleistungen anderen Softwarekomponenten zur Verfügung stellen kann.

Es ist in der Tat so, daß sich Hersteller reifer Systeme – also i.w. universeller *Expertensystem-Schalen* – aus kommerziellen Gründen mit diesen Fragen befassen: Ohne eine Integration mit existierender Software wird auf Dauer keine wesentliche Verbreitung dieser Technik zu erwarten sein.

1.1.5.8 Wartung

In dieser Systemkomponente wird dem Nutzer ein Zugang zu den vorhandenen Wissenseinheiten angeboten, der es erlaubt, den Inhalt der Wissensbasen zu inspizieren und zu modifizieren. Ebenso erwartet man hier Funktionen, die alle Stadien des Entwurfs, der Realisierung, des Tests und der Weiterentwicklung einer wissensbasierten Anwendung tragen.

Hier spiegelt sich am deutlichsten die im System verwendete Repräsentationsstruktur wider: Objektorientierte (hierarchische) Strukturen erfordern eine andere Darstellung als etwa Regeln oder Datenbankinhalte. Es ist zu beobachten, daß in dieser Komponente ein großer Anteil an Phantasie und Prinzipien des „user centered system design“ eine Rolle spielen; von einer durchgängigen Standardisierung kann aber heute noch keine Rede sein.

Es sollen zunächst die hauptsächlichen Funktionen einer solchen Wartungskomponente charakterisiert werden; dann kann man kurz über gebräuchliche Techniken nachdenken, die hier verwendet werden.

Inspektion: Der Inhalt der Wissensbasis soll unter verschiedenen Kriterien betrachtet werden. Diese Kriterien reichen von einem generellen Überblick über die Anzahl der Wissenseinheiten, ihre gegenseitigen Beziehungen, Gruppierung und Hierarchisierung bis zum Einblick in aufgebaute Datenstrukturen, den Inhalt einzelner Wissenselemente bis hin zu begleitender Information, wie Autorenschaft oder Dokumentation.

Modifikation: Wissenseinheiten permanenter Natur (etwa Klassendefinitionen, Regeldefinitionen, Begriffsbeschreibungen) erweisen sich oft als fehlerhaft, unvollständig oder hinfällig. Der Benutzer erwartet Hilfsmittel, die entsprechende Modifikationen ermöglichen. Hat die Revision eines Wissenselements größere *nicht-lokale* Konsequenzen, so liegt die „Verantwortung“ bei der Wartungskomponente, diese Effekte darzustellen oder (besser) im voraus darauf aufmerksam zu machen.

Unterstützung beliebiger Entwurfsmethoden: Was im klassischen Software Engineering lange gang und gäbe ist, wird sich aller Voraussicht nach über kurz oder lang auch in Entwurf und Realisierung wissensbasierter Systeme einbürgern: *kooperativer Entwurf*, d.h. die Entwicklung eines großen Domänenmodells durch ein Team von Personen, die u.U. dediziert an einzelnen Teilen des Modells arbeiten. Software Engineering Werkzeuge stellen ein Repertoire von Mechanismen zur Verfügung, um die Probleme des kooperativen Entwurfs etwas zu mildern.

1.1.5.9 Inspektion und Modifikation

Eine umfangreiche Wissensbasis stellt eine Konstruktion mit einer gewissen Komplexität dar. Ein Repräsentationsformalismus mit einer großen Anzahl *strukturbildender Operatoren* und den entsprechenden Inferenzmechanismen wird Strukturen aufbauen, die u.U. nicht vorherzusehen sind, nicht unmittelbar erklärbar oder schlicht nicht beabsichtigt. Ein Anwendungssystem, das auf einer solchen Wissensbasis operiert, wird – gewöhnlich selbst ziemlich komplex – unter gewissen Bedingungen eine Funktionalität zeigen, die a priori nicht erwartet und a posteriori nicht erklärbar ist. Die Tatsache, daß es sich auf eine komplexe Daten- und Prozeßstruktur abstützt, macht es unvermeidbar, Mittel zu schaffen, mit denen der Inhalt der Wissensbasis inspiziert und gegebenenfalls modifiziert werden kann.

Natürlich wird man sich beim Entwurf dieser Mittel am Profil des entsprechenden Nutzers orientieren müssen; unter dem Stichwort „Software-Ergonomie“ [Lauter 87; Raasch 91] sind „Checklisten“ entwickelt worden, die es erlauben, die meisten Fehler im Entwurf von Benutzerschnittstellen zu vermeiden. Davon soll hier also nicht die Rede sein. Vielmehr wird man sich Gedanken darüber machen, ob es irgendwelche Aspekte solcher Funktionalität gibt, die speziell für wissensbasierte Systeme interessant sind.

Grundsätzlich führt die Vielzahl von Eigenschaften der beteiligten Konstrukte dazu, daß die Hilfsmittel der Inspektion sich eine dieser Eigenschaften als primären Blick auf diese Konstrukte wählen und die Darstellung gemäß dieser Eigenschaft strukturieren. In der so entstehenden Struktur, die noch weitgehend von den Details abstrahiert, stehen dann Möglichkeiten der *Navigation* und der *Fokussierung* zur Verfügung.

Unter *Navigation* soll hier die Möglichkeit verstanden werden, große Strukturen schnell zu durchlaufen, um einzelne Objekte zu finden, an derem inneren Aufbau man interessiert ist. *Fokussierung* ändert die Darstellung eines gewählten Objekts oder Ausschnitts, um Details anderer möglicher Aspekte sichtbar zu machen.

Die Wahl dieser primären Struktur, in der navigiert wird, ist unter Umständen abhängig vom Ziel der Inspektion und der Sicht, die der Benutzer auf den Inhalt der Wissensbasis hat. Überläßt man die Wahl dieser primären Struktur dem Benutzer, so ist ein vergleichsweise großer Aufwand bei der Implementation die Folge. Hier einige Sichtweisen, die sich in der Praxis als nützlich erwiesen haben:

Nicht-hierarchische Formalismen. Darunter sind in erster Linie regel-basierte Formalismen zu verstehen. Regeln können aus Gründen der Übersichtlichkeit oder der effizienteren Verarbeitung zu *Regelmengen* zusammengefaßt werden. Diese *Gruppierung* ist eine der Eigenschaften, die zur Übersicht über einen Regelsatz verwendet werden kann. Andere Sichten, die unter bestimmten Bedingungen erwünscht sind, sind Zu-

sammenfassungen von Regeln, die ein bestimmtes Fakt erschließen oder bestimmte Fakten in ihrer Prämisse erwähnen.

Hierarchische Formalismen. Hierzu zählen etwa terminologische, objekt-orientierte oder frame-artige Formalismen. Kennzeichnend dafür ist die Tatsache, daß diese Formalismen Konzepte, Objekte oder Frames in Hierarchien einordnen. Damit ist im allgemeinen die primäre Struktur vorgegeben, in der navigiert wird. Das Hilfsmittel dazu ist der bekannte *Browser*, der Bestandteil aller solcher Systeme ist. Andere mögliche primäre Sichtweisen hängen von möglichen Unterscheidungen des Formalismus auf der epistemologischen Ebene ab, so kann man sich etwa vorstellen, daß in KL-ONE-artigen Systemen eine Darstellung lediglich der *primitiven Konzepte*, die in gewisser Weise das Skelett des Modells bilden, nützlich sein kann.

Funktionen zur Modifikation eines Teils der Wissensbasis sind erfahrungsgemäß algorithmisch aufwendiger, als man annimmt. Der Grund liegt offenbar darin, daß es sich – im Sinne der Logik – bei Modifikationen um *nicht-monotone* Operationen mit allen unschönen Konsequenzen handelt. Sind vom System auf der Basis der nun modifizierten Wissenseinheit bereits Schlüsse gezogen worden, die unter Umständen globale Auswirkungen gehabt haben, so kann das zur Folge haben, daß eine kleine Modifikation globale Konsequenzen für den Inhalt der Wissensbasis hat; im schlimmsten Fall kann sie inkonsistent werden. Neben den Problemen, die mit der Realisierung dieser Funktionalität verbunden sind, bleibt die Aufgabe eines Systems die, den Benutzer auf den Effekt seines Modifikationswunsches aufmerksam zu machen, was ausgesprochen schwierig zu sein scheint.

Das Repertoire der Modifikationsmöglichkeiten festzulegen, ist ein Problem, das vielleicht einer kurzen Analyse bedarf. Mag es z.B. sinnvoll sein, eine Regel zu löschen, die einen unzulässigen Schluß ausgeführt hat, mag es auch sinnvoll sein, ein Frame zu löschen, weil es keine Notwendigkeit mehr dafür gibt, so ist die Frage, ob man in einem terminologischen System ein Konzept „löschen“ kann, doch einen kurzen Gedanken wert.

Auf der epistemischen Ebene besteht ein Konzept aus einer Beschreibung (bzw. einer Definition) und einem Namen dafür. Von einem passenden Inferenzmechanismus ist dafür gesorgt worden, daß die Beschreibung in sich nicht widersprüchlich ist, d.h., es kann durchaus Objekte geben, auf die diese Beschreibung paßt. Es kann höchstens sein, daß der mit der Beschreibung verknüpfte Name nicht zu der gegebenen Beschreibung paßt, aber das kann man auch durch Umbenennen und Angabe einer anderen Beschreibung für den alten Namen lösen (vgl. auch [Nebel 90a]).

1.1.5.10 Kooperative Nutzung

Kooperativer Zugang zu großen Softwaresystemen ist heutzutage nahezu selbstverständlich. Dies gilt für die Entwicklungsphase genauso wie für die Nutzungsphase. Kein nichttriviales Softwareprodukt wird von einzelnen realisiert; es gibt Teams mit wohldefinierten Aufgaben und mehr oder weniger formale „Absprachen" zwischen ihnen. Der Entwurf einer großen Wissensbasis ist von ähnlicher Komplexität wie der Entwurf eines jeden großen Programms. Die Nutzung einer Wissensbasis muß mehreren Teilnehmern parallel möglich sein, um den großen Aufwand zu rechtfertigen. Hier scheint ein gewisser Nachholbedarf im Bereich wissensbasierter Systeme zu bestehen.

Es scheint – gemessen an der Anzahl grundlegender Artikel – vergleichsweise wenig Arbeiten zum Thema des kooperativen Entwurfs und der kooperativen Nutzung wissensbasierter Systeme zu geben. Die Probleme und Lösungsansätze sind aus anderen Gebieten der Informatik, die sich mit der Realisierung großer Softwaresysteme befassen, i.w. bekannt.

Im Rahmen der Entwicklung großer Systeme stellt sich das Problem, Entwickler zu koordinieren, die gleichzeitig an einer gemeinsamen Datenstruktur, namentlich einer großen Wissensbasis arbeiten. Die wesentlichen Probleme dabei enstehen dadurch, daß die Operationen, die von einer Gruppe vorgenommen werden, Auswirkungen auf die Prämissen haben können, unter denen eine andere Gruppe arbeitet. Bei klassischer Software kann man dem durch strikte Trennung der Zuständigkeiten einzelner Komponenten (Modularisierung) entgegenwirken, die über wohldefinierte Schnittstellen miteinander kommunizieren.

Bei großen Modellen ist die Situation oft nicht so, daß die einzelnen Teile keine oder leicht kontrollierbare Verbindungen zueinander haben. In solchen stark vernetzten Modellen unterliegt die Kontrolle der Konsequenzen einzelner Modifikationen dem Verwaltungssystem, das mit Methoden wie *Propagierung* nicht-lokale Änderungen vornehmen kann. Methoden der *Versionskontrolle*, die im Software-Engineering angewendet werden, scheinen in diesen Fällen nicht unmittelbar anwendbar zu sein. Mays [Mays et al. 91] stellt einen Ansatz vor, der durch Markierung einzelner Wissenseinheiten die Koordination einzelner Entwicklungsgruppen ermöglicht.

Kooperative Nutzung wissensbasierter Systeme stellt ähnliche Probleme, die ansatzweise durch Konzepte aus dem Bereich der Datenbanken, insbesondere *Transaktionen*, behandelt werden können. Dennoch sind viele Fragen offen. Insbesondere ist nach Schemata zu suchen, die die Sperrung von Teilen einer stark vernetzten Wissensbasis regulieren.

Literaturverzeichnis

[Allen, Hayes 87] James F. Allen, Patrick J. Hayes. Movements and points in an interval-based temporal logic. Technical Report 180, Univ. of Rochester, 1987.

[Allen 83] James Allen. Mantaining knowledge about temporal intervals. *Communications of the ACM*, 26(10):832–843, 1983.

[Baader 90a] Franz Baader. A formal definition for expressive power of knowledge representation languages. In *Proceedings of the 9th European Conference on Artificial Intelligence*, Stockholm, Schweden, 1990.

[Baader 90b] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pp. 621–626, Boston, MA, 1990.

[Baader et al. 91] Franz Baader and Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, Hans-Jürgen Profitlich. Terminological knowledge representation: a proposal for a terminological logic. In B. Nebel, K. von Luck, C. Peltason (Hrsg.), *International workshop on terminological logics.* DFKI Document D-91-13, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, 1991. Auch als KIT Report, TU Berlin und IWBS Report, IBM Germany, Stuttgart, publiziert.

[Bobrow, Winograd 79] Daniel G. Bobrow, Terry Winograd. KRL—another perspective. *Cognitive Science*, 3(1):29–42, 1979.

[Brachman, Levesque 85] Ronald J. Brachman, Hector J. Levesque, Hrsg.. *Readings in Knowledge Representation.* Morgan Kaufmann, Los Altos, CA, 1985.

[Brachman 79] Ronald J. Brachman. On the epistomological status of semantic networks. In N. Findler, Hrsg., *Associative Networks*, pp. 3–50. Academic Press, New York, NY, 1979.

[Cheeseman et al. 91] Peter Cheeseman, Bob Kanefsky, William M. Taylor. Where the *really* hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pp. 331–337, Sydney, Australien, 1991.

[Christaller et al. 89] Thomas Christaller et al. *Die KI-Werkbank Babylon.* Addison-Wesley, Reading, MA, 1989.

[Davis 90] Ernest Davis. *Commonsense Knowledge*. Morgan Kaufmann, Los Altos, CA, 1990.

[Dean, Boddy 88] Thomas L. Dean, Mark Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36(3):375–400, October 1988.

[Donini et al. 91a] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt. The complexity of concept languages. In James A. Allen, Richard Fikes, Erik Sandewall, Hrsg., *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pp. 151–162, Cambridge, MA, April 1991.

[Donini et al. 91b] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pp. 458–465, Sydney, Australien, August 1991.

[Doyle, Patil 91a] Jon Doyle, Ramesh S. Patil. Two theses of knowledge representation: language restrictions, taxonomic classifications, and the utility of representation services. *Artificial Intelligence*, 48(3):261–297, 1991.

[Doyle, Patil 91b] Jon Doyle, Ramesh S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–298, April 1991.

[Dreyfus 81] Hubert L. Dreyfus. From micro-worlds to knowledge-representation: AI at an impasse. In J. Haugeland, Hrsg., *Mind Design*, pp. 161–204. MIT Press, Cambridge, MA, 1981.

[Eisinger, Ohlbach 87] Norbert Eisinger, Hans-Jürgen Ohlbach. Grundlagen und Beispiele. In Karl-Heinz Bläsius, Hans-Jürgen Bürckert, Hrsg., *Deduktionssysteme*, pp. 23–41. Oldenbourg Verlag, München, 1987.

[Faltings et al. 89] Boi Faltings, Emmanual Baechler, Jeff Primus. Reasoning about kinematic topology. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 1331–1336, Detroit, MI, 1989.

[Garey, Johnson 79] Michael R. Garay, David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[Hanks, McDermott 87] Steve Hanks, Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, November 1987.

[Hayes 79] Patrick J. Hayes. The naive physics manifesto. In Don Michie, Hrsg., *Expert Systems in the Microelectronic Age*. Edinburgh Univ. Press, 1979.

[Hayes 85] Patrick J. Hayes. Naive physics I: Ontology for liquids. In J. R. Hobbs, R. C. Moore, Hrsg., *Formal Theories of the Commonsense World*, pp. 71–108. Ablex, Norwood, NJ, 1985.

[Heinsohn et al. 94] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, Hans-Jürgen Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2): 367–397, 1994.

[Hobbs, Moore 85] Jerry R. Hobbs, Robert C. Moore. *Formal Theories of the Commonsense World*. Ablex, Norwood, NJ, 1985.

[Hobbs et al. 85] Jerry R. Hobbs, Tom Blenko, Bill Croft, Greg Hager, Henry A. Kautz, Paul Kube, Yoav Shoham. Commonsense summer: Final report. CSLI Report CSLI-85-35, Stanford Univ., 1985.

[Hollunder et al. 90] Bernhard Hollunder, Werner Nutt, Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 348–353, Stockholm, Schweden, 1990.

[Kortüm 91] Gerd Kortüm. Temporales Schließen in einem natürlichsprachlichen System. LILOG Report 162, IBM Deutschland, Stuttgart, 1991.

[Lauter 87] Barbara Lauter. *Software-Ergonomie in der Praxis*. Oldenbourg Verlag, München, 1987.

[Lehnert, Wilks 79] Wendy Lehnert, Yorick Wilks. A critical perspective on KRL. *Cognitive Science*, 3(1):1–28, January 1979.

[Lenat, Guha 90] Doug B. Lenat, R. V. Guha. *Building Large Knowledge-Based Systems*. Addison-Wesley, Reading, MA, 1990.

[Levesque, Brachman 87] Hector J. Levesque, Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Levesque 86] Hector J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1):81–108, October 1986.

[Levesque 88] Hector J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[Mays et al. 91] Eric Mays, Robert Dionne, Robert Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, June 1991.

[McCarthy 59] John McCarthy. Programs with common sense. *Proc. Symposium on Mechanisation of Thought Processes 1*, 1959.

[McCarthy 68] John McCarthy. Programs with common sense. In Marvin Minsky, Hrsg., *Semantic Information Processing*, pp. 403–418. MIT Press, Cambridge, MA, 1968.

[McDermott 87] Drew V. McDermott. A critique of pure reason. *Computational Intelligence*, 3(3):151–160, 1987.

[Miller et al. 60] G. A. Miller, E. Galanter, K. H. Pribram. *Plans and the Structure of Behavior*. Holt, Rinehard and Winston, 1960.

[Minsky 75] Marvin Minsky. A framework for representing knowledge. In P. Winston, Hrsg., *The Psychology of Computer Vision*, pp. 211–277. McGraw-Hill, New York, NY, 1975.

[Minsky 91] Marvin Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *The AI Magazine*, 12(2):34–51, 1991.

[Nebel, Bäckström 94] Bernhard Nebel, Christer Bäckström. On the computational complexity of temporal projection, planning, and plan validation. *Artificial Intelligence*, 66(1):125–150, 1994

[Nebel, Smolka 91] Bernhard Nebel, Gert Smolka. Attributive description formalisms ...and the rest of the world. In Otthein Herzog, Claus-Rainer Rollinger, Hrsg., *Text Understanding in LILOG*, Band 546 *Lecture Notes in Artificial Intelligence*, pp. 439–452. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[Nebel 90a] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Band 422 *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1990.

[Nebel 90b] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Nebel 91] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In J. Sowa, Hrsg., *Principles of Semantic Networks*, pp. 331–362. Morgan Kaufmann, San Mateo, CA, 1991.

[Newell 82] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982.

[Quillian 66] M. Ross Quillian. *Semantic Memory*. Carnegie Institute of Technology, Pittsburgh, PA, 1966. BBN Report AFCRL-66-189, Bolt, Beranek, and Newman Inc., October 1966.

[Raasch 91] Jörg Raasch. *Systementwicklung mit Strukturierten Methoden*. Carl Hanser Verlag, München, 1991.

[Reiter 87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

[Schank 73] Roger C. Schank. Identification of conceptualization underlying natural language. In R. C. Schank, K. M. Colby, Hrsg., *Computer Models of Thought and Language*, pp. 187–247. Freeman, San Francisco, CA, 1973.

[Schefe 85] Peter Schefe. Zur Rekonstruktion von Wissen in neueren Repräsentationssprachen der Künstlichen Intelligenz. In Herbert Stoyan, Hrsg., *GWAI-85. 9th German Workshop on Artificial Intelligence*, pp. 230–244. Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[Schild 91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pp. 466–471, Sydney, Australia, 1991.

[Schmidt-Schauß, Smolka 91] Manfred Schmidt-Schauß, Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[Schmolze, Woods 92] James G. Schmolze, William A. Woods. In F. Lehman, *Semantic Networks in Artificial Intelligence*, Pergamon Press, 1992.

[Sloman 85] Aaron Sloman. Why we need many knowledge representation formalisms. In M. Bramer, Hrsg., *Research and Development in Expert Systems*. Cambridge University Press, Cambridge, UK, 1985.

[Touretzky 86] David S. Touretzky. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, Los Altos, CA, 1986.

[Winograd, Flores 86] Terry Winograd, Fernando Flores. *Understanding Computers and Cognition*. Addison-Wesley, Reading, MA, 1986.

[Winston et al. 87] Patrick H. Winston, Roger Chaffin, Douglas Hermann. A taxonomy of part-whole relations. *Cognitive Science*, 11:417–444, 1987.

[Woods 75] William A. Woods. What's in a link: Foundations for semantic networks. In D. G. Bobrow, A. M. Collins, Hrsg., *Representation and Understanding: Studies in Cognitive Science*, pp. 35–82. Academic Press, New York, NY, 1975.