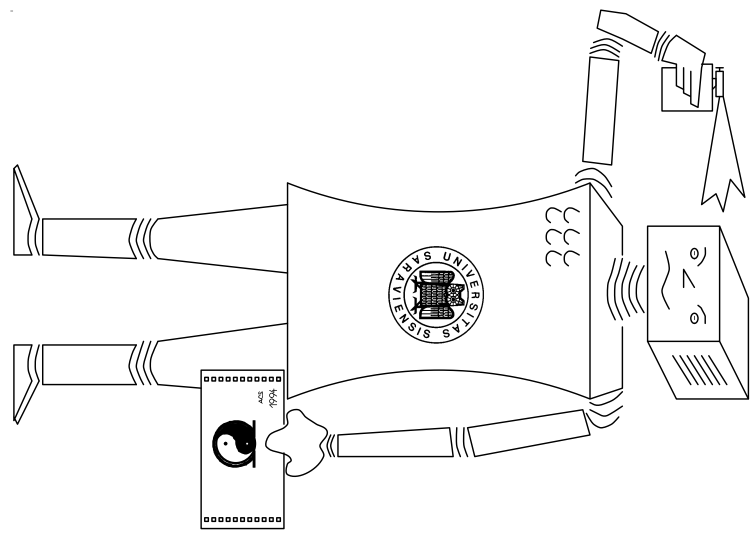
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

WWW: `http://js-sfbsun.cs.uni-sb.de/pub/www/`



Deduktionssysteme

Manfred Kerber and Christoph Weidenbach

Published as: in *KI*, 1992, **3/92**, pages 14–22.

Deduktionssysteme

Manfred Kerber
Fachbereich Informatik
Universität des Saarlandes
6600 Saarbrücken

Christoph Weidenbach
Max-Planck-Institut für Informatik
Universität des Saarlandes
6600 Saarbrücken

Eine wichtige Form des formalen *Schließens* ist die *Deduktion*. Sie ist im Vergleich zu anderen Schlußweisen sehr gut verstanden und in verschiedensten Deduktionssystemen operationalisiert. Das Ausgangsproblem, das man mit Deduktionssystemen lösen will, ist für eine gegebene Formelmenge Δ und eine Formel φ automatisch festzustellen, ob φ aus Δ ableitbar ist. Wir werden dafür im folgenden Beispiele geben und auch zeigen, wie sich diese Problemstellung im Laufe der Zeit erweitert hat (vergleiche Abschnitt „Beantwortung von Anfragen").
Es ist in der KI ein gängiges Problem festzustellen, ob eine Formel φ aus einer Formelmenge Δ ableitbar ist (zum Beispiel in regelbasierten Expertensystemen). Unter Formeln verstehen wir dabei Ausdrücke einer formalen Sprache. Für eingeschränkte Sprachen oder Formelmengen lassen sich zur Lösung des Problems effiziente Algorithmen finden. Für hinreichend mächtige Sprachen gibt es allerdings keinen Algorithmus, der für beliebige Δ und φ entscheidet, ob sich φ aus Δ ableiten läßt. Trotzdem ist es oft angebracht, solch mächtige Sprachen zu benutzen, da viele Probleme nur so angemessen formuliert werden können. Deshalb ist man an Inferenzmechanismen interessiert, die für in der Praxis vorkommende Beispiele gute Ergebnisse liefern, obwohl der allgemeine Fall unentscheidbar ist. Dieses ist die klassische Aufgabe von Deduktionssystemen.
Mit diesem Artikel wollen wir die Grundlagen von Deduktionssystemen kurz einführen und auf die Schwierigkeiten bei Problemlösungen eingehen. Es ist uns dabei wichtiger, einen Eindruck der inhärenten Probleme und der damit verbundenen aktuellen Forschungsfragen zu vermitteln als einen Überblick aller gängigen Methoden anzugeben. Dafür verweisen wir auf die einschlägige Literatur.[1]

[1]insbesondere auf [7], wo viele der im folgenden angesprochenen Themen vertieft werden

Im folgenden werden zuerst die grundlegenden Begriffe erklärt, wobei die Prädikatenlogik erster Stufe (oft mit PL1 abgekürzt) als wohl wichtigste Logik Ausgangspunkt auch zum Vergleich zu anderen Logiken ist. Im darauffolgenden Abschnitt werden zwei für das automatische Beweisen sehr wichtige Kalküle, der Tableaukalkül und das Resolutionsverfahren vorgestellt, sowie Aspekte der Mechanisierung erörtert. Schließlich gehen wir noch auf gängige Erweiterungen der Prädikatenlogik erster Stufe ein: auf Gleichheitsbehandlung, auf Sorten und auf Logik höherer Stufe.

Logik und Deduktion

Deduktion bedeutet die Ableitung einer Aussage aus anderen Aussagen durch *Schlußregeln*. Aussagen werden dabei in einer formalen Sprache, einer *Logik*, beschrieben. Als Beispiel wollen wir die folgende Ableitung in natürlicher Sprache in Logik übersetzen:
Aus „Max ist ein Mensch." und „Wenn Max ein Mensch ist, dann ist der Vater von Max ein Mensch." folgt „Der Vater von Max ist ein Mensch."
In Logik können diese Sätze in die folgenden *Formeln*, das heißt Ausdrücke mit einer festgelegten *Syntax*[2] übersetzt werden.

$$\frac{\begin{array}{l}\alpha\\ \alpha \Rightarrow \beta\end{array}}{\beta}$$

Dabei steht α für „Max ist ein Mensch" und β für „Der Vater von Max ist ein Mensch". Die hier angewandte Schlußregel heißt *Modus Ponens*.
Gewisse Aussagen nimmt man als gegeben an, diese werden *Axiome* genannt. Ein Beispiel ist das Axiom vom ausgeschlossenen Dritten: $\alpha \vee \neg\alpha$, welches anschaulich besagt, daß man für jede Aussage α sie selbst oder aber ihr Gegenteil annehmen kann. Eine Menge von Schlußregeln zusammen mit einer Menge von Axiomen bildet einen *Kalkül*. Ausgehend von den Axiomen und von gewissen Annahmen können mit Hilfe der Schlußregeln neue Aussagen erzeugt werden. Wir bezeichnen die Formelmenge dieser Annahmen, im folgenden Datenbasis genannt, immer mit Δ. Im obigen Beispiel besteht Δ aus den beiden Formeln „α" und „$\alpha \Rightarrow \beta$". Dabei hat man das Ziel, eine Formel φ abzuleiten. Im Beispiel oben ist φ gerade β. Ein solcher Kalkül wird ein *deduktiver Kalkül* genannt. Bei einem Testkalkül geht man umgekehrt von φ aus und führt es mit Hilfe der Schlußregeln auf die Axiome und auf Δ zurück. Eine wichtige Spezialform sind die *negativen Testkalküle*, bei denen man mit den Axiomen, mit Δ und der Negation von φ beginnt und mit Hilfe der Regeln einen Widerspruch ableitet.
Während das obige Beispiel mit Hilfe der Aussagenlogik formalisiert werden kann, ist dies für das folgende nicht der Fall, sondern man benötigt Prädikatenlogik erster Stufe, PL1, mit der man Ausdrücke wie „für alle" ($\forall$) und „es gibt" ($\exists$) formal fassen kann. Betrachten wir den folgenden Schluß: „Max ist ein Mensch." und „Für alle Menschen gilt, daß ihre Väter wieder Menschen sind." also „Der Vater von Max ist ein Mensch."
Die zugehörige Übersetzung in PL1 hat die folgende Gestalt:

[2]Wir gehen auf diese Syntax hier nicht näher ein, sie ergibt sich aber aus den Beispielen.

Δ:
(1) $Mensch(max)$
(2) $\forall x\,(Mensch(x) \Rightarrow Mensch(vater(x)))$

und (3) $\varphi := Mensch(vater(max))$.

Das Symbol „x" in $\forall x\,(Mensch(x) \Rightarrow Mensch(vater(x)))$ heißt eine *Objektvariable*. Das Symbol „max" ist ein *Konstantensymbol*, „*Mensch*" ist ein *Prädikatssymbol* und „*vater*" ein *Funktionssymbol*. Ausdrücke wie „max", „x", „$vater(x)$" oder „$vater(vater(max))$" heißen *Terme*.
Mit einer Instantiierungsregel kann man allquantifizierte Variablen durch Terme ersetzen:

$$\frac{\forall x\,\alpha(x)}{\alpha(t)}$$

Damit ist es möglich, durch Instantiierung von x mit max aus (2) herzuleiten:

(4) $Mensch(max) \Rightarrow Mensch(vater(max))$

Mit dem Modus Ponens erhält man dann aus (1) und (4) die gewünschte Aussage „$Mensch(vater(max)))$".
Prädikatenlogik erster Stufe unterscheidet sich von Prädikatenlogiken höherer Stufen dadurch, daß eine Quantifizierung über Prädikate, wie „*Mensch*", oder über Funktionen wie „*vater*" nicht möglich ist. Daher lassen sich gewisse Sachverhalte wie die Induktionsaxiome oder die Frameaxiome in Prädikatenlogik erster Stufe nur annähern, aber nicht vollständig beschreiben.
Im allgemeinen geht man allerdings davon aus, daß der in der Mathematik informell benutzte Begriff „beweisbar" mit dem Begriff „beweisbar in PL1" zusammenfällt (Hilbertthese [4, S.41]). PL1 ist darüberhinausgehend die ausdrucksstärkste Logik für die wichtige Struktursätze gelten [32].[3] Somit erscheint es prinzipiell hinreichend für PL1 Deduktionssysteme zu bauen. In der Tat lassen sich alle Erweiterungen von PL1, für die es korrekte und vollständige Kalküle gibt, auch in PL1 ausdrücken. Allerdings ist diese Charakterisierung mit der Turingäquivalenz von Programmiersprachen vergleichbar. Die gängigen höheren Programmiersprachen sind *prinzipiell* gleichmächtig, für einen *konkreten* Anwendungsbereich ist die Auswahl der Programmiersprache jedoch äußerst wichtig. In erheblich stärkerem Umfang ist dies bei der Auswahl von Logiken der Fall. Für bestimmte Problemklassen ist es eher eine theoretische als eine praktische Option zu versuchen, sie in PL1 automatisch zu lösen. Dies hat vor allem zwei Gründe: zum einen kann die Ausdrucksstärke für eine adäquate Repräsentation zu gering sein, zum anderen ist die Abarbeitung ohne Spezialkonstrukte oft zu ineffizient. Allerdings sollte man die Ausdrucksmöglichkeiten in PL1 auch nicht unterschätzen. Ein Problem kann auf verschiedenste Arten in PL1

[3] Keine Erweiterung der Ausdrucksstärke in diesem Sinne sind nicht-monotone Logiken. PL1 ist monoton, das heißt: Wenn aus einer Formelmenge eine weitere Formel hergeleitet werden kann (wie aus (1) und (2) hergeleitet werden konnte, daß der Vater von Max ein Mensch ist), so kann diese Formel auch dann noch hergeleitet werden, wenn die Ausgangsmenge um zusätzliche Formeln (wie zum Beispiel durch „$Philosoph(max)$") erweitert wird. Dadurch unterscheiden sich die hier beschriebenen Logiken also von denen im Bereich des nicht-monotonen Schließens betrachteten.

formuliert werden, und die geschickte Problemformulierung ist manchmal schon die halbe Lösung [26].
Was Formeln bedeuten, wird durch die *Semantik* festgelegt. Die möglichen Bedeutungen ganzer Formeln sind durch die Wahrheitswerte „wahr“ oder „falsch“ gegeben. Ein *Modell* einer Formel (beziehungsweise einer Formelmenge) ist eine Interpretation der Formel (beziehungsweise aller Formeln in der Formelmenge), die den Wahrheitswert „wahr“ ergibt. Hat Δ ein Modell, so heißt es erfüllbar oder konsistent. Mit Hilfe einer logischen Ableitung einer Formel φ aus einer Formelmenge Δ will man zeigen, daß jedes Modell von Δ auch ein Modell von φ ist. In der Logik wird dies streng formalisiert, wir können allerdings im weiteren nicht näher darauf eingehen, sondern verweisen auf einschlägige Logikbücher wie [17, 39]. Für PL1 wählt man üblicherweise eine Semantik, bei der Terme als Elemente einer nichtleeren Menge, dem *Universum*, interpretiert werden. Wichtig für das folgende ist, daß es für PL1 Kalküle gibt, die exakt der Semantik entsprechen, das heißt, die *korrekt* und *vollständig* sind. Somit ist es möglich die inhaltlichen *semantischen* Aspekte von Schlüssen auszuschalten (das heißt, es ist nicht nötig, Modelle zu betrachten) und man kann rein *syntaktisch* (also auf der Kalkülseite) vorgehen. Leider bedeutet dies aber nicht, daß man damit ein Entscheidungsverfahren dafür hat, ob aus einer Menge Δ eine Formel φ ableitbar ist.

Kalküle

In diesem Abschnitt wollen wir auf die Grundlage der automatischen Beweiser, die Kalküle, eingehen. Historisch wurden dabei zuerst Kalküle mit Regeln wie den oben angegebenen betrachtet. Allerdings hat man für die Automatisierung von PL1 sehr bald andere Verfahren entwickelt. Aus der Vielzahl von Kalkülen, und diese gibt es auch noch in verschiedensten Varianten, haben wir in diesem Abschnitt zwei wichtige Verfahren ausgewählt, nämlich das Tableauverfahren und die Resolution.

Tableauverfahren

Ein negatives Testverfahren für PL1, das besonders für aussagenlogische Formeln sehr effizient ist, ist das Tableauverfahren [45]. Es erfordert im Gegensatz zum Resolutionsverfahren keine Normalform als Ausgangspunkt, vielmehr geht man von Δ und der Negation von φ aus und baut aus diesen Formeln durch die Anwendung von *Dekompositionsregeln* sukzessive einen Baum (mit den Formeln in Δ und der Negation von φ als Startknoten) auf. Ein Zweig (Weg von der Wurzel zu einem Blatt des Baumes) heißt *geschlossen*, wenn sich in ihm syntaktisch komplementäre Formeln befinden. Ist jeder Zweig geschlossen, so hat man φ aus Δ abgeleitet. Das Tableauverfahren ist korrekt und vollständig.
Insgesamt stehen 13 Dekompositionsregeln zur Verfügung, durch die Formeln zerlegt werden können. Charakteristische Beispiele für Regeln sind:

$$\frac{\neg(\alpha \Rightarrow \beta)}{\begin{array}{c}\alpha \\ \neg\beta\end{array}} \qquad (\neg \Rightarrow)$$

Man kann durch diese Regel einen Zweig ergänzen, der die Formel $\neg(\alpha \Rightarrow \beta)$ enthält, in dem man an das Blatt erst die Formel α und dann die Formel $\neg\beta$ anfügt.
Die Regeln

$$\frac{\neg(\alpha \wedge \beta)}{\neg\alpha|\neg\beta} \quad (\neg\wedge) \qquad\qquad \frac{\alpha \Rightarrow \beta}{\neg\alpha|\beta} \quad (\Rightarrow)$$

eröffnen unter dem Blatt zwei neue Zweige durch die nebeneinanderstehenden Formel $\neg\alpha$ und $\neg\beta$ beziehungsweise $\neg\alpha$ und β.
Zur Behandlung von Quantoren gibt es Instantiierungsregeln der Gestalt:

$$\frac{\forall x\, \alpha(x)}{\alpha(t)} \quad (\forall) \qquad\qquad \frac{\neg\forall x\, \alpha(x)}{\neg\ \alpha(c)} \quad (\neg\forall)$$

Das heißt aus Formeln mit Quantoren darf man neue ableiten, in dem man einen Term t instantiiert beziehungsweise, in dem man eine neue Konstante c einführt.
Als Beispiel wollen wir wieder ableiten, daß der Vater von Max ein Mensch ist:

(1) $Mensch(max)$	(Δ)
(2) $\forall x\,(Mensch(x) \Rightarrow Mensch(vater(x)))$	(Δ)
(3) $\neg Mensch(vater(max))$	$(\neg\varphi)$
(4) $Mensch(max) \Rightarrow Mensch(vater(max))$	$(\forall)(2)$
(5) $Mensch(vater(max))$	$(\Rightarrow)(4).(3)$
(6) $\neg Mensch(max)$	$(\Rightarrow)(4).(1)$

Die Zahl hinter dem Punkt gibt an, durch welche Formel der Zweig geschlossen wird. Ein größeres Beispiel für eine Ableitung mit Hilfe des Tableauverfahrens findet man im Abschnitt „Vergleich von Kalkülen“.

Resolution

Resolution ist auch ein negatives Testverfahren, das bei seiner Erfindung einen großen Fortschritt darstellte, da die Instantiierungen nicht mehr wie im Tableauverfahren geraten werden müssen [41]. Sie setzt eine Normalform, die *Klauselnormalform*, voraus, von der ausgehend weitere *Klauseln* hergeleitet werden. Kann man die *leere Klausel*, das einzige Axiom für eine falsche Formel, ableiten, so war die ursprüngliche Klauselmenge und damit die Ausgangsformelmenge unerfüllbar.
Die Klauselnormalform ist eine Konjunktion von Disjunktionen, das heißt, eine Formel der Gestalt:

$$(L_{11} \vee \ldots \vee L_{1n_1}) \wedge \ldots \wedge (L_{m1} \vee \ldots \vee L_{mn_m})$$

wobei die einzelnen L_{ij}, die sogenannten *Literale*, *atomare Formeln* (also Formeln ohne Junktoren und Quantoren) oder deren Negation sind. $(L_{i1} \vee \ldots \vee L_{in_i})$ heißt eine *Klausel*. Üblicherweise notiert man die Klauselnormalform als eine Liste von Klauseln, die implizit mit „$\wedge$“ verknüpft sind. Die einzelnen Klauseln werden dabei

durchnummeriert. In der Klauselnormalform sind alle Variablen implizit allquantifiziert. Existenzquantoren werden durch die *Skolemisierung* eliminiert, zum Beispiel erhält man aus $\forall x\ \exists y\ P(x,y)$ die Formel $\forall x\ P(x,f(x))$, wobei f ein neues Funktionssymbol ist. Zu jeder Formelmenge kann man eine Klauselnormalform berechnen, die genau dann unerfüllbar ist, wenn es die Ursprungsformelmenge war.
Als Beispiel wollen wir wieder die Formelmenge vom Anfang betrachten. Es ist zu zeigen, daß aus (1) und (2) die Formel (3) folgt.
Deshalb werden (1), (2) in Klauselnormalform transformiert:

Δ:
(1) $Mensch(max)$
(2) $\neg Mensch(x) \vee Mensch(vater(x))$

Zu Δ wird nun die Klauselnormalform der Negation von φ hinzugefügt:

(3) $\neg Mensch(vater(max))$

Es ist nun zu zeigen, daß diese Klauselmenge unerfüllbar ist.
Die Resolutionsregel angewendet auf (1)1 und (2)1 – die zweite Zahl soll dabei das entsprechende Literal angeben – liefert in diesem Fall die *Resolvente*

(4) $Mensch(vater(max))$

Diese wiederum resolviert mit Klausel (3)1, was die leere Klausel

(5) $\square$

ergibt. Also war die Ausgangsklauselmenge unerfüllbar und damit ist φ aus Δ ableitbar.
Die allgemeine Resolutionsregel hat die Gestalt:

$$\frac{\begin{array}{ll} L \vee K_1 \vee \ldots \vee K_n & \\ \neg L' \vee M_1 \vee \ldots \vee M_m & \qquad \sigma(L) = \sigma(L') \end{array}}{\sigma(K_1) \vee \ldots \vee \sigma(K_n) \vee \sigma(M_1) \vee \ldots \vee \sigma(M_m)}$$

Dabei ist σ die allgemeinste Einsetzung die L und L' gleich macht, der *allgemeinste Unifikator*. Im obigen Beispiel ist σ gleich $(x \leftarrow max)$.
Zusammen mit der Faktorregel, mit der man unter bestimmten Bedingungen zwei Literale innerhalb einer Klausel zu einem verschmelzen kann und der leeren Klausel als Axiom für eine widersprüchliche Formel erhält man einen widerlegungsvollständigen Kalkül.
Eine wichtige Eigenschaft der Resolution ist, daß, wegen der Berechnung der allgemeinsten Unifikatoren, es nicht notwendig ist, Einsetzungen für universell quantifizierte Variablen zu raten.
Eine Verfeinerung für das Resolutionsprinzip stellt das sogenannte *Klauselgraphverfahren* dar, bei dem die möglichen Resolutionsschritte durch Kanten zwischen Literalen in einem Graphen, der aus Klauseln besteht, explizit repräsentiert werden. Dieses Verfahren ist besonders wegen spezieller Löschungsregeln reizvoll, die es auf syntaktischer Ebene erlauben, sinnlose Schritte zu erkennen [29, 35].

Mechanisierung

Wir werden exemplarisch an den oben eingeführten Kalkülen (Tableau und Resolution) die folgenden Grundaussagen zeigen:

- Das Problem der Mechanisierung von Kalkülen, d.h. eine Anfrage an eine Datenbasis logischer Formeln automatisch zu beantworten, ist ein hartes Problem.
- Verschiedene Kalküle, Repräsentationen und Implementierungen verhalten sich signifikant anders bezüglich ihrer Perfomanz auf verschiedenen Formelklassen.
- Die adäquate Repräsentation von speziellen Strukturen verlangt spezielle Sprachkonstrukte und spezielle Inferenzmechanismen.

Beantwortung von Anfragen

Ausgangspunkt für das Problem der Beantwortung von Anfragen ist eine Datenbasis Δ logischer Formeln erster Stufe. Ein Beispiel für eine solche Datenbasis ist:

Δ:
(1) $Mensch(max)$
(2) $\forall x\,(Mensch(x) \Rightarrow Mensch(vater(x)))$
(3) $\forall x, y\,(Mensch(x) \wedge Mensch(y) \Rightarrow Liebt(x, y))$

Die erste Formel bedeutet, daß Max ein Mensch ist, die zweite Formel beschreibt, daß Väter von Menschen wieder Menschen sind und die dritte Formel sagt, daß alle Menschen einander lieben[4]. Man kann folgende Anfragetypen unterscheiden:

(a) Ist eine Formel φ ableitbar aus der Datenbasis Δ? Zum Beispiel: Ist $\varphi := \forall y\,(Mensch(y) \Rightarrow Liebt(vater(max), y))$ aus Δ ableitbar?
(b) Welche zusätzlichen Annahmen werden benötigt, um φ aus der Datenbasis ableiten zu können? Zum Beispiel: Welche Annahmen rechtfertigen $\varphi := Liebt(max, maria)$ aus Δ abzuleiten?
(c) Welche neuen Formeln sind ableitbar, falls man eine Formel in die Datenbasis einfügt? Zum Beispiel: Was ist ableitbar wenn man $Mensch(maria)$ in Δ einfügt?
(d) Wie könnte ein Modell für die Datenbasis Δ aussehen?

Für alle vier Anfragetypen ist es schwierig automatisch eine Antwort zu generieren. Man kann beweisen, daß es keinen Algorithmus gibt, der für eine beliebige Datenbasis Δ und eine beliebige Formel φ die Anfragen (a), (b), (c) oder (d) nach endlicher Zeit beantwortet.
Die Fragestellungen sind auch nicht unabhängig voneinander. So kann man mit Hilfe von Anfrage (c) eine Anfrage (a) beantworten, in dem man die Formel $\neg\varphi$ zur Datenbasis Δ hinzufügt und überprüft, ob sich ein Widerspruch ableiten läßt.
Wir werden nun die verschiedenen Fragestellungen für unser Beispiel mit der Resolutionsmethode lösen. Dazu überführen wir Δ in Klauselnormalform:

[4] Paradiesische Zustände

Δ:
(1) $Mensch(max)$
(2) $\neg Mensch(x) \vee Mensch(vater(x))$
(3) $\neg Mensch(x) \vee \neg Mensch(y) \vee Liebt(x,y)$

Um nun (a) zu beantworten, d.h. ob sich aus Δ die Formel $\varphi := \forall y\,(Mensch(y) \Rightarrow Liebt(vater(max),y))$ ableiten läßt, negieren wir φ, wandeln sie in Klauselnormalform um und fügen sie zu Δ:

(4) $Mensch(a)$
(5) $\neg Liebt(vater(max),a)$

wobei a eine neue Konstante ist, die durch Skolemisierung des Existanzquantors in $\exists y\, \neg(Mensch(y) \Rightarrow Liebt(vater(max),y))$ entsteht. Die Beantwortung der Anfrage sieht dann so aus:

(6) $\neg Mensch(vater(max)) \vee \neg Mensch(a)$
Resolution zwischen den Literalen (5)1 und (3)3
(7) $\neg Mensch(vater(max))$
Resolution zwischen den Literalen (6)2 und (4)1
(8) $\neg Mensch(max)$
Resolution zwischen den Literalen (7)1 und (2)2
(9) $\Box$
Resolution zwischen den Literalen (8)1 und (1)1

Die Lösung war in vier Schritten relativ einfach zu berechnen. Das war jedoch nur der Fall, weil die richtigen Schritte ausgewählt wurden. Schon in diesem einfachen Beispiel gibt es für die betrachtete Anfrage unendlich viele nutzlose Resolutionsschritte der Klausel (2) mit sich selbst, die Klauseln der Form

$$\neg Mensch(x) \vee Mensch(vater^i(x)) \qquad i > 1$$

erzeugen. Nur wenn man die richtigen Schritte weiß, findet man schnell eine Lösung. Es gibt aber keine Intuition, die sich automatisieren ließe und die nur die guten Schritte auswählt. Deshalb arbeiten letztendlich alle Deduktionssysteme mit einem fairen Verfahren, d.h. nach endlicher Zeit wird jeder mögliche Schritt ausgeführt. Eine Lösung wird nur dann gefunden, falls man die leere Klausel $\Box$ ableiten kann, bevor die Anzahl der zu betrachtenden Klauseln zu stark angestiegen ist. Für schwierige Beispiele müssen Suchräume mit Millionen von Klauseln aufgebaut werden. Wie stark der Suchraum anwächst, hängt von vielen Faktoren ab, unter anderem vom verwendeten Kalkül, worauf wir im nächsten Abschnitt näher eingehen.
Die Härte des Suchproblems läßt sich durch die Parameter Verzweigungsrate b und Lösungstiefe l charakterisieren. Die Lösungstiefe l ist die Länge der kürzesten Ableitungsfolge, die zur Lösung führt. Die Verzweigungsrate b gibt an, wie schnell der Suchraum wächst, d.h. wie viele neue Formeln bei vollständiger Anwendung der Inferenzregeln auf die aktuelle Formelmenge berechnet werden können. Damit läßt sich die Anzahl der zu erzeugenden Formeln, um die Lösung zu finden, durch b^l approximieren. Im allgemeinen ist das eine untere Schranke, da die Verzweigungsrate eher anwächst als abnimmt. Nehmen wir dieses sehr grobe Maß, so ergibt sich für das

Eingangsbeispiel eine Lösungstiefe von 4 und eine Verzweigungsrate von 10 (die Anzahl möglicher Resolventen zwischen allen Klauseln im ersten Schritt). Somit hätten wir bei einer blinden Aufzählung aller möglichen Resolventen $10^4 = 10000$ Klauseln zu erzeugen, bevor die leere Klausel garantiert gefunden werden kann. Natürlich arbeitet kein existierendes System auf diese Art und Weise. Es gibt spezielle Regeln, Heuristiken und Strategien um den Suchraum einzuschränken. Im konkreten Beispiel wurde mit der Anfrage gestartet. Man kann sich leicht überlegen, daß nur Resolventen zwischen der Anfrageformel (und deren Nachfolgern) und der Klauselmenge betrachtet werden müssen. Diese Strategie (genannt „Set of Support") schränkt für das Beispiel bereits den Suchraum auf $4^4 = 256$ Klauseln ein.

Der Anfragetyp (a) ist das ursprüngliche Problem des automatischen Beweisens. Hier gibt es eine Vielzahl von klassischen einführenden Büchern wie [13, 33] und eine Menge von Büchern, die auch neuere Entwicklungen berücksichtigen [7, 6]. Beispiele für Deduktionssysteme, die mit einem Resolutionskalkül arbeiten, sind das MKRP-System [35] oder das OTTER-System [34].

Um nun die Anfrage (b) zu beantworten, d.h. finde eine Formel γ, so daß aus Δ erweitert um γ die Formel $\varphi := Liebt(max, maria)$ folgt, kann man im Prinzip so verfahren wie bei (a). Man startet mit (4) $\neg Liebt(max, maria)$ und bildetet nun alle möglichen Resolventen zwischen dieser Klausel, ihren Nachfahren und den Klauseln in Δ:

(5) $\neg Mensch(max) \vee \neg Mensch(maria)$
Resolution zwischen dem Literal (3)3 und (4)1

(6) $\neg Mensch(maria)$
Resolution zwischen den Literalen (5)1 und (1)1

Mit Klausel (6) sind nun keine Schritte mehr möglich und die gesuchte Formel ist deshalb $\gamma := Mensch(maria)$, d.h. wenn wir wissen, daß $Mensch(maria)$ gilt, können wir $Liebt(max, maria)$ ableiten. Obwohl das Verfahren nur die Antwort $Mensch(maria)$ generiert, gibt es noch weitere mögliche Antworten. So wären auch die Formeln $Liebt(max, maria)$, $Mensch(max) \wedge Mensch(maria)$, $Mensch(x)$, etc. mögliche Antworten gewesen. Man nennt $Liebt(max, maria)$ die triviale Antwort, $Mensch(max) \wedge Mensch(maria)$ eine redundante Antwort weil die Information $Mensch(max)$ bereits in Δ enthalten ist und $Mensch(x)$ eine unspezifische Antwort weil $Mensch(maria)$ als Antwort bereits ausreicht. Ist man nur an nicht trivialen, mit Δ konsistenten, nicht redundaten, möglichst spezifischen Antworten interessiert, so ist bezüglich dieser Charakterisierung $Mensch(maria)$ die einzige Antwort.

Wir haben hier eine vereinfachte Version der Arbeit von Demolombe und Fariñas [16] vorgestellt. Weitere Arbeiten gibt es hier auch von Stickel [47].

Auch die Beantwortung von (c), welche zusätzlichen Formeln ableitbar sind, falls eine neue Formel zu Δ hinzugefügt wird, ist mit der Resolutionsmethode prinzipiell möglich. Wir starten mit (4) $Mensch(maria)$ und leiten alle möglichen Resolventen aus dieser Klausel her:

(5) $\neg Mensch(y) \vee Liebt(maria, y)$
Resolution zwischen dem Literal (3)1 und (4)1
(6) $\neg Mensch(x) \vee Liebt(x, maria)$
Resolution zwischen dem Literal (3)2 und (4)1
(7) $Liebt(max, maria)$
Resolution zwischen den Literalen (6)1 und (1)1
(8) $\neg Mensch(x) \vee Liebt(vater(x), maria)$
Resolution zwischen den Literalen (6)1 und (2)2
$\vdots$

Die Resolutionsmethode terminiert nicht für dieses Beispiel und wir können unendlich viele Folgerungen aus $Mensch(maria)$ ableiten.
Die Fragestellung (d), wie könnte ein Modell für Δ aussehen, ist die in der Literatur am wenigsten behandelte und vielleicht die praktisch schwierigste. Ein naiver Ansatz wäre die Resolutionsmethode allgemein auf Δ anzuwenden und zu hoffen, daß das Verfahren terminiert und eine vollständige endliche Menge von möglichen Resolventen berechnet. Enthält diese Menge die leere Klausel, so ist Δ inkonsistent, enthält sie die leere Klausel nicht, so ist Δ konsistent und aus den generierten Klauseln läßt sich ein Modell erzeugen. Wie das kleine Beispiel jedoch zeigt, terminiert die Resolutionsmethode nur in sehr wenigen Fällen. Deshalb wurden spezielle Verfahren entwickelt die erfolgreicher sind [12, 38]. Ein Modell für Δ könnte so aussehen, daß Max der einzige Mensch ist, er sich selbst liebt und er sein eigener Vater ist[5].

Vergleich von Kalkülen

Im vorherigen Abschnitt wurde bereits die Beantwortung von Anfragen mit Hilfe der Resolutionsmethode erörtert. Alternativ wird nun die Beantwortung von Anfrage (a) unter Benutzung des Tableaukalküls diskutiert. Wir erhalten folgenden Baum:

[5] Diese Interpretation hat mit der Realität wenig zu tun. Selbst die Gentechnologie dürfte damit ihre Probleme haben.

(1)$Mensch(max)$	(Δ)
(2)$\forall x\,(Mensch(x) \Rightarrow Mensch(vater(x)))$	(Δ)
(3)$\forall x, y\,(Mensch(x) \wedge Mensch(y) \Rightarrow Liebt(x, y))$	(Δ)
(4)$\neg\forall y\,(Mensch(y) \Rightarrow Liebt(vater(max), y))$	$(\neg\varphi)$
(5)$\neg(Mensch(a) \Rightarrow Liebt(vater(max), a))$	$(\neg\forall)(4)$
(6)$Mensch(a)$	$(\neg \Rightarrow)(5)$
(7)$\neg Liebt(vater(max), a)$	$(\neg \Rightarrow)(5)$
(8)$Mensch(max) \Rightarrow Mensch(vater(max))$	$(\forall)(2)$
(9)$\neg Mensch(max)$	$(\Rightarrow)(8).(1)$
(10)$Mensch(vater(max))$	$(\Rightarrow)(8)$
(11)$Mensch(vater(max)) \wedge Mensch(a) \Rightarrow Liebt(vater(max), a)$	$(\forall)(3)$
(12)$Liebt(vater(max), a)$	$(\Rightarrow)(11).(7)$
(13)$\neg(Mensch(vater(max)) \wedge Mensch(a))$	$(\Rightarrow)(11)$
(14)$\neg Mensch(a)$	$(\neg\wedge)(13).(6)$
(15)$\neg Mensch(vater(max))$	$(\neg\wedge)(13).(10)$

Das Beispiel illustriert einige für die Mechanisierung wichtige Unterschiede zwischen dem Resolutions- und dem Tableaukalkül:

1. Das Tableauverfahren folgt der Struktur der Eingabeformeln, während Resolution auf der Klauselnormalform arbeitet.

2. Quantoren werden explizit instantiert: bei Allquantoren können beliebige Terme eingesetzt werden, bei Existenzquantoren wird eine neue Konstante generiert. Resolution instantiiert Variablen nur so weit als nötig durch den Unifikationsalgorithmus.

3. Fallunterscheidungen werden explizit durch Verzweigungen im Baum ausgeführt, z.B. bei der Anwendung der $(\Rightarrow)$ Regel. Das entspricht bestimmten Restriktionen bei der Anwendung der Resolutionsregel.

Die Punkte 1 und 3 führen dazu, daß Tableaubeweiser im allgemeinen auf aussagenlogischen Problemen effizienter sind als Resolutionsbeweiser. Die Notwendigkeit, Quantoren vollständig zu instantiieren, ist ein Nachteil des Tableauverfahrens, weil

völlig unklar ist, welcher Term eingesetzt werden soll. So wäre es auch möglich gewesen die Formel (2) im Tableaubeweis nicht mit max zu instantiieren (Formel (8)), sondern mit $vater(max)$, $vater(vater(max))$, ...
Es gibt aber auch tiefere Unterschiede zwischen den Kalkülen, so läßt sich eine Formel der Form

$$(5)\ Liebt(vater^{2^i}(max), maria)$$

wobei die Datenbasis um die Formel (4) $Mensch(maria)$ erweitert wird, im Resolutionskalkül in $i+4$ Schritten beweisen, indem i mal die Klausel (2) mit sich selbst resolviert und dann (1), (3), (4) und (5) angewendet wird. Im Tableaukalkül braucht man hingegen mehr als 2^i Schritte, das heißt exponentiell mehr Schritte als im Resolutionskalkül.
Eine genauere Untersuchung der relativen Komplexität von Kalkülen findet man in [18, 30]. Bekannte Systeme die nach dem Tableau- oder verwandten Verfahren arbeiten sind das SETHEO-System [31] oder das HARP-System [36].

Erweiterungen der Prädikatenlogik erster Stufe

In einigen Bereichen werden Spezialfälle von PL1 untersucht, da diese für bestimmte Anwendungen hinreichend und günstiger sind. So betrachtet man bei der Untersuchung taxonomischen Wissens ein Fragment von PL1, das noch entscheidbar ist. Im logischen Programmieren wird PL1 zur sogenannten Hornlogik eingeschränkt, die zwar auch nicht entscheidbar ist, für die es aber möglich ist, sehr effiziente Abarbeitungen zu erreichen [49]. Wir werden diese Ansätze im weiteren nicht diskutieren, sondern in diesem Abschnitt auf *Spracherweiterungen* von PL1 eingehen.
Anhand von drei Beispielen werden wir belegen, daß die Standardkalküle der Prädikatenlogik für viele Anwendungen zu schwach sind, um eine effiziente Mechanisierung zu ermöglichen. Es müssen vielmehr Erweiterungen berücksichtigt werden, die eine adäquate Repräsentation bestimmter Sachverhalte gewährleisten.
Abstrakt gesehen geht es darum, Theorien, d.h. Prädikate oder Funktionen mit einer festgelegten Bedeutung, effizient in den Kalkül zu integrieren. Hier gibt es prinzipiell zwei Möglichkeiten: (i) der Kalkül kann um neue Inferenzregeln erweitert werden, z.B. die Paramodulationsregel zur Integration der Gleichheit in das Resolutionsverfahren, (ii) der Standardunifkationsalgorithmus kann durch einen Theorieunifikationsalgorithmus ausgetauscht werden, z.B. die sortierte Unifikation zur Integration von Sorten.
Für beide Ansätze gibt es eine Vielzahl von Arbeiten. Zu (i) wären als generelle Arbeiten [46] oder [11] zu nennen. Des weiteren gibt es Arbeiten, die erläutern, wie eine spezielle Theorie integriert werden kann (siehe unten). Aus der Idee, Theorien in den Unifikationsalgorithmus einzubauen, hat sich ein eigenes Gebiet entwickelt: die Unifikationstheorie [43].

Gleichheit

Das vielleicht beste Beispiel für die Notwendigkeit die Sprache der Prädikatenlogik erster Stufe zu erweitern ist die Gleichheit, d.h. die Kodierung des zweistelligen

Prädikats „=“ mit der Semantik, daß $t_1 = t_2$ genau dann gilt, wenn t_1 und t_2 als das gleiche Element des Universums interpretiert werden.
Man kann die Gleichheit im Prinzip wie jedes andere Prädikat, z.B. das Prädikat *Mensch* oder *Liebt* im Eingangsbeispiel, formalisieren. Für das Eingangsbeispiel müßten die Formeln (5)–(10) zur Datenbasis Δ hinzugefügt werden:

Δ:
(1) $Mensch(max)$
(2) $\forall x\,(Mensch(x) \Rightarrow Mensch(vater(x)))$
(3) $\forall x, y\,(Mensch(x) \wedge Mensch(y) \Rightarrow Liebt(x, y))$
(4) $maxerl = max$
Axiome der Gleichheit:
(5) $\forall x\,(x = x)$
(6) $\forall x, y\,(x = y \Rightarrow y = x)$
(7) $\forall x, y, z\,(x = y \wedge y = z \Rightarrow x = z)$
(8) $\forall x, y\,(x = y \Rightarrow vater(x) = vater(y))$
(9) $\forall x, y\,(x = y \Rightarrow (Mensch(x) \Rightarrow Mensch(y)))$
(10) $\forall x, y, v, z\,(x = y \wedge v = z \Rightarrow (Liebt(x, v) \Rightarrow Liebt(y, z)))$

Die Formeln (5)–(7) codieren die Reflexivität, Symmetrie und Transitivität der Gleichheitsrelation. Die Formeln (8)–(10) bedeuten, daß gleiche Objekte als Argumente von Funktionen und Prädikaten zum gleichen Ergebnis führen. Die Formel (4) ist eine echte Erweiterung der Datenbasis und besagt, daß Max den Spitznamen Maxerl hat.
Wir wollen die Formel $\varphi := Liebt(maxerl, max)$ ableiten. Dazu überführen wir Δ in Klauselnormalform:

Δ:
(1) $Mensch(max)$
(2) $\neg Mensch(x) \vee Mensch(vater(x))$
(3) $\neg Mensch(x) \vee \neg Mensch(y) \vee Liebt(x, y)$
(4) $maxerl = max$
Axiome der Gleichheit:
(5) $x = x$
(6) $x \neq y \vee y = x$
(7) $x \neq y \vee y \neq z \vee x = z$
(8) $x \neq y \vee vater(x) = vater(y)$
(9) $x \neq y \vee \neg Mensch(x) \vee Mensch(y)$
(10) $x \neq y \vee v \neq z \vee \neg Liebt(x, v) \vee Liebt(y, z)$

und fügen die Klauselnormalform von $\neg\varphi$

(11) $\neg Liebt(maxerl, max)$

zu Δ. Wenden wir die vorher erwähnte Strategie des Set of Support an, so gibt es initial zwei mögliche Resolventen:

(12) $\neg Mensch(maxerl) \vee \neg Mensch(max)$
Resolution zwischen den Literalen (11)1 und (3)3
(13) $x \neq maxerl \vee v \neq max \vee \neg Liebt(x, v)$
Resolution zwischen den Literalen (11)1 und (10)4

Von diesem Zustand an hat sich die Verzweigungsrate auf $l = 11$ erhöht. Da wir noch 5 Schritte bis zur Lösung brauchen, z.B. in dem wir mit (12) nacheinander die Klauseln (1), (9), (1), (6), (4) resolvieren, ergibt sich eine Abschätzung für den Suchraum von $11^5 = 161051$ Klauseln. Das bedeutet, daß dieses simple Beispiel bei einer axiomatischen Behandlung der Gleichheit zu einem schwierigen Problem wird. Die Gleichheitsaxiome blähen in einer solch dramatischen Weise den Suchraum auf, daß es praktisch unmöglich ist, überhaupt noch etwas mit Resolution zu zeigen. Deshalb wurden für die Gleichheit neue Ableitungsregeln erfunden, insbesondere die Paramodulation:

$$\frac{\begin{array}{ll} t = s \lor K_1 \lor \ldots \lor K_n & \\ L[\ldots t' \ldots] \lor M_1 \lor \ldots \lor M_m & \sigma(t') = \sigma(t) \end{array}}{\sigma(L[\ldots s \ldots]) \lor \sigma(M_1) \lor \ldots \lor \sigma(M_m) \lor \sigma(K_1) \lor \ldots \lor \sigma(K_n)}$$

wobei σ der allgemeinste Unifikator von t' und t ist. Bei der Anwendung der Paramodulationsregel können t und s ihre Rollen tauschen. Es hat sich gezeigt, daß man sich bei t' auf Terme beschränken kann, die keine Variablen sind. Die Schreibweise $L[\ldots t' \ldots]$ bedeutet, daß t' als Unterterm in dem Literal L vorkommt. Diese Regel ist um ein Vielfaches effizienter als der axiomatische Ansatz, weil man die Annahmen (6)–(10) nun nicht mehr braucht. Wir erhalten somit nach dem ersten Schritt nur noch die Klausel (12) bei einer Verzweigungsrate von 3. Außerdem ist die Lösungstiefe geringer, da wir (12)1 mit (4)1 paramodulieren:

(13) $\neg Mensch(max) \lor \neg Mensch(max)$

und dann zweimal mit (1) resolvieren. Es ergibt sich also ein Suchraum von $3^3 = 27$ Klauseln gegenüber 161051 im obigen Fall. Das heißt, durch Einführung einer speziellen Technik ist der Suchraum näherungsweise um den Faktor 6000 geschrumpft. Trotzdem ist der Suchraum auch bei Verwendung der Paramodulationsregel im allgemeinen immer noch sehr groß. Deshalb wurden effizientere Inferenzregeln entwickelt. Heute ist auch die Paramodulationsregel weitgehend durch sogenannte Vervollständigungsverfahren abgelöst. Hier wird zusätzlich die Anwendung der Paramodulation gerichtet, d.h. man wendet Gleichungen nur noch in eine Richtung an. Zusätzlich werden mächtige Techniken zur Reduktion des Suchraums bereitgestellt.
Die Paramodulation wurde zuerst von Robinson und Wos [40] vorgestellt. Das Vervollständigungsverfahren wurde zuerst von Knuth und Bendix [27] eingeführt und dann weiter verallgemeinert und verbessert [22, 52, 3].

Sorten

Ein zweites Beispiel für die Integration einer Theorie ist die Erweiterung des Resolutionskalküls um Sorten, eine besondere Behandlung einstelliger Prädikatensymbole. Eine sortierte Formalisierung des Eingangsbeispiels ist

Δ':
(1′) $max \lessdot Mensch$
(2′) $\forall x_{Mensch}\, vater(x_{Mensch}) \lessdot Mensch$
(3′) $\forall x_{Mensch}, y_{Mensch}\, Liebt(x_{Mensch}, y_{Mensch})$

Das Symbol $\lessdot$ bedeutet die Elementbeziehung zwischen einem Term und einer Sorte (z.B. $max \lessdot Mensch$) und wir erlauben, Variablen auf Sorten einzuschränken (z.B. x_{Mensch}).
Die sortierte Kodierung ist viel näher an der menschlichen Intuition als die am Anfang verwendete. Die sortierte Formel (3′) sagt „alle Menschen lieben einander“, während die unsortierte Formel (3) sagt „für alle zwei Dinge gilt, wenn beide Menschen sind, dann lieben sie einander“, eine Formulierung, die niemand wählen würde. Man sieht hier schon, daß die unsortierte Formulierung auch nicht so effizient handhabbar sein kann wie die sortierte, weil erst einmal alle Elemente des Universums als mögliche Einsetzung für die Variablen betrachtet werden und erst dann geprüft wird, ob sie das Prädikat Mensch erfüllen. In der sortierten Version werden a priori nur Menschen für eine Einsetzung in die Variablen zugelassen. Würden wir z.B. zur unsortierten Datenbasis Δ noch die Klausel (4) $\neg Liebt(2,4)$[6] hinzufügen, so wäre Resolution mit Klausel (3) möglich. Mit der Resolvente (5) $\neg Mensch(2) \vee \neg Mensch(4)$ sind jedoch keine weiteren Resolutionsschritte mehr möglich, weil ja 2 und 4 nicht als Menschen deklariert wurden (z.B. $Mensch(2)$). Die Klausel (5) kann nicht zu einer Ableitung der leeren Klausel beitragen. In der sortierten Fassung wird das a priori erkannt und der Resolutionsschritt zwischen (3′) und (4) erst gar nicht zugelassen.
Man kann beweisen, daß bei einem sortierten Kalkül für die sortierten Variablen nur solche Terme eingesetzt werden müssen, die die entsprechende Sorte haben, z.B. darf für die Variable x_{Mensch} nur ein Term der Sorte *Mensch* eingesetzt werden, d.h. ein Term der sich über die Deklarationen der Sorte *Mensch* ($max \lessdot Mensch$, $vater(x_{Mensch}) \lessdot Mensch$) konstruieren läßt (z.B. max, $vater(x_{Mensch})$, $vater(max)$, ...). Solche Terme heißen *wohlsortiert*. Der Test welche Sorte ein Term hat, ist für praktisch alle bekannten Sortenlogiken effizient entscheidbar.
Der Vorteil der sortierten Fassung liegt in einer geringeren Verzweigungsrate (denn es werden nur wohlsortierte Einsetzungen für die Variablen betrachtet) und einer geringeren Lösungstiefe des Suchraums. Der Grund dafür ist das „Verschwinden“ einiger Literale durch die Sortierung der Variablen und die Einschränkung der zugelassenen Unifikatoren auf wohlsortierte Unifikatoren.
Fügen wir nun die Klauseln aus der Anfrage (a) (Abschnitt „Beantwortung von Anfragen“)

$$\begin{array}{ll} (4') & a \lessdot Mensch \\ (5') & \neg Liebt(vater(max), a) \end{array}$$

zu Δ', so erhalten wir eine Verzweigungsrate von 1. Der einzig mögliche Resolutionsschritt ist zwischen $5'(1)$ und $3'(1)$, wobei wir überprüfen müssen, ob a von der Sorte *Mensch* und $vater(max)$ von der Sorte *Mensch* ist. Das ist trivialerweise der Fall und deshalb erhalten wir bereits nach einem Schritt die leere Klausel. Die sortierte Fassung hat somit einen Suchraum von eins.
Eine sortierte Formalisierung verbessert immer das Kalkülverhalten, sofern einstellige Prädikate bei der Lösung involviert sind. Insbesondere lassen sich die Verfahren zur Beantwortung von Fragen der Typen (b)–(d) signifikant verbessern. In unserem Beispiel terminiert die sortierte Version von Δ' bzgl. Anfrage (c).

[6] Die 2 liebt nicht die 4, z.B. weil sie keine Primzahl ist.

Ein weiterer Vorteil der sortierten Formalisierung ist die Möglichkeit einer Syntaxüberprüfung, ähnlich der in höheren typisierten Programmiersprachen. Man könnte für Δ' z.B. festlegen, daß das Prädikat *Liebt* nur auf Termen der Sorte *Mensch* definiert ist. Die Formel $Liebt(2,4)$ wäre dann eine syntaktisch nicht korrekte Formel und könnte a priori zurückgewiesen werden.
Die ersten vollständig entwickelten sortierten Kalküle wurden von Walther [48] und Schmidt-Schauß [42] vorgestellt. Neuere Arbeiten sind z.B. [5, 51, 50].

Logiken höherer Stufe

Logik erster Stufe ist zwar für viele Probleme sehr gut geeignet, aber zur Darstellung mathematischer Sachverhalte ist es oft auch nötig über Prädikate oder Funktionen zu quantifizieren. Beispielsweise lassen sich selbst die natürlichen Zahlen nicht endlich in PL1 ausdrücken, da man das folgende Induktionsaxiom (oder ein vergleichbares) braucht:

$$\forall\ P_{pred} P(0) \land (\forall n_{\mathbb{N}} P(n) \Rightarrow P(n+1)) \Rightarrow \forall\ n_{\mathbb{N}} P(n)$$

Es gibt aber auch speziell für die KI wichtige Formeln höherer Stufe, so ist zum Beispiel die Circumscription-Formel (vgl. [19, S.134]) zweiter Stufe. Diese beiden Formeln sind echt höherer Stufe und lassen sich in ihrer vollen Allgemeinheit nicht durch Formeln erster Stufe ersetzen. Um mögliche Paradoxien wie die von Russell zu vermeiden, ist es üblich allen Variablen und Konstanten von Logiken höherer Stufe einen Typ zuzuordnen, so daß Formeln der Gestalt $P(P)$ ausgeschlossen werden.
Wie bereits erwähnt, gibt es keine vollständigen Kalküle für Logiken höherer Stufe bezüglich der intuitiven Standardsemantik [20]. Allerdings hat Leon Henkin einen abgeschwächten Semantikbegriff eingeführt [21], der die Standardsemantik approximiert und für den es möglich ist, vollständige und korrekte Kalküle anzugeben. Bezüglich dieser Semantik kann man auch Übersetzungen von Logik höherer Stufe in die erster Stufe angeben. Üblicherweise geschieht das durch die Einführung einer Mengenlehre, aber es gibt auch direktere Darstellungen, die für die üblichen Deduktionssysteme erster Stufe besser geeignet sind [25]. Allerdings sind in beiden Fällen für eine große Klasse von Problemen zusätzliche Zusammenfassungsaxiome notwendig.
Betrachten wir als Beispiel die Aussage, daß alle Menschen alle Eigenschaften vom Vater oder von der Mutter erben, das heißt, daß sich jede ihrer Eigenschaften bei Vater oder Mutter wiederfinden läßt. Dies kann man in Logik zweiter Stufe durch die folgende Formel φ ausdrücken:

$$\begin{aligned}\forall\ P_{pred}\ \forall\ x \quad & Mensch(x) \land P(x) \Rightarrow \\ & P(vater(x)) \lor P(mutter(x))\end{aligned}$$

Durch Instantiierung können wir aus φ die folgende Formel ableiten

$$\begin{aligned}\forall\ x \quad & Mensch(x) \land hat\_Schweißfüße(x) \Rightarrow \\ & hat\_Schweißfüße(vater(x)) \lor hat\_Schweißfüße(mutter(x))\end{aligned}$$

Dies ist noch relativ einfach zu handhaben. Allerdings will man nun für die Prädikatsvariable P auch ganze Formeln (mit einer freien Variable) einsetzen können. Zum Beispiel könnte man $P(t)$ durch $(trinkt\_kaffee(t) \Rightarrow schläft\_schlecht(t))$ ersetzen. Als Instanz von φ erhält man dann:

$$\begin{aligned}\forall\, x \quad & Mensch(x) \wedge (trinkt\_kaffee(x) \Rightarrow schläft\_schlecht(x)) \Rightarrow \\ & (trinkt\_kaffee(vater(x)) \Rightarrow schläft\_schlecht(vater(x))) \vee \\ & (trinkt\_kaffee(mutter(x)) \Rightarrow schläft\_schlecht(mutter(x)))\end{aligned}$$

Wenn wir ableiten wollen, daß eine Formelmenge bestehend aus φ, der Aussage, daß Max nach dem Genuß von Kaffee nicht schlafen kann, sowie der Aussage, daß seine Eltern beide nach Kaffee sehr gut schlafen, widersprüchlich ist, so benötigen wir bei einer Übersetzung der Aussagen in Logik erster Stufe das folgende Axiom, das es uns erlaubt die Variable P mit einer ganzen Formel gleichzusetzen:

$\exists Q_{pred}\, \forall x\; Q(x) \iff (trinkt\_kaffee(x) \Rightarrow schläft\_schlecht(x))$

Nun kann P mit Q instantiiert werden. Da die Einführung solcher Axiome – von denen es unendlich viele gibt – im allgemeinen schwer zu automatisieren ist, hat man ein spezielles Beweisverfahren für die Logik höherer Stufe entwickelt: den Lambdakalkül mit Unifikation höherer Stufe [14, 1, 23]. Der wohl am weitesten entwickelte Beweiser für Logik höherer Stufe ist das TPS-System [2]. Will man in diesem Kalkül die Unerfüllbarkeit der obigen Formelmenge zeigen, so liefert der Unifikationsalgorithmus höherer Stufe den Unifikator $\sigma = (P \leftarrow \lambda x(trinkt\_kaffee(x) \Rightarrow schläft\_schlecht(x)))$, kein zusätzliches Axiom ist notwendig. Damit lassen sich viele Probleme effizient lösen. Allerdings wird die Unifikation selbst als Teilschritt im Problemlösungsverfahren unentscheidbar.

Auch im Bereich Logiken höherer Stufe sind wieder Erweiterungen wie um Sorten sinnvoll und werden zur Zeit untersucht [28].

Als eigenes Gebiet im Bereich Beweisen „höherer Stufe“ hat sich das „Induktionsbeweisen“ herausgebildet. Obwohl *theoretisch* durch die allgemeinen Verfahren abgedeckt, sind die Probleme *praktisch* so aufwendig, daß spezielle Deduktionstechniken entwickelt wurden [8]. Ein Hauptproblem beim Beweisen durch vollständige Induktion ist das Finden einer geeigneten Induktionshypothese. Im Bereich des Beweisens durch vollständige Induktion ist man von allen Gebieten wohl am weitesten in der Entwicklung bereichsabhängiger Heuristiken. Diese nutzen die spezielle Beziehung zwischen Induktionsvoraussetzung und Induktionshypothese aus [10, 24].

Beweisprüfer

Während die bislang erwähnten Systeme darauf ausgerichtet sind, Deduktionen selbständig durchzuführen, gibt es eine weitere Klasse von Systemen, die Menschen bei der Entwicklung von Beweisen oder anderen Herleitungen dadurch unterstützen, daß sie die benutzerentwickelten Beweise en détail nachprüfen. Somit kann die Zuverlässigkeit von Beweisen erheblich vergrößert werden, während die kreative Seite des Beweisens weiterhin beim Menschen bleibt. Allerdings hat dieses Verfahren den Nachteil, daß Menschen ihre Sätze üblicherweise auf einer viel abstrakteren, semantischeren Ebene führen als dies in einem streng formalen Verfahren möglich ist. Entsprechende formale Beweise sind um ein Vielfaches länger, so daß auch diese Systeme bislang keinen Durchbruch in die Anwendung geschafft haben. Um die Anwendbarkeit zu vergrößern, wird neueren Beweisprüfern allgemein eine mächtige Sprache zugrunde gelegt, damit der Benutzer sich komfortabel ausdrücken kann. Weiterhin gibt es die Möglichkeit, sogenannte Taktiken zu schreiben, die kleinere

Lücken in eingegebenen Beweisen ausfüllen können. Bekannte Systeme sind Automath [9], Nuprl [15] und Isabelle [37]. Im Projekt Ω-MKRP [44] wird gegenwärtig ein interaktives Beweissystem entwickelt, das Beweisprüfer und automatische Beweiser miteinander verbindet.

Diskussion

Wir wollten mit den Beispielen im vorletzten Abschnitt zeigen, daß die ursprüngliche Sprache der Prädikatenlogik in vielen Fällen zu arm ist, um eine effiziente Mechanisierung zu ermöglichen. Für viele Probleme müssen, obwohl sie sich in Standard-PL1 formalisieren ließen, spezielle Sprachkonstrukte und spezielle Inferenzmechanismen zur Verfügung gestellt werden. Erst dann kann man hoffen, sie automatisch zu lösen. Die Entwicklung neuer Inferenzmechanismen für in der Praxis relevante Theorien ist bei weitem noch nicht zu Ende. Weitere Beispiele für solche Ansätze findet man in den Artikeln zu den Themen nicht-klassische und taxonomische Logiken in diesem Heft.

Wegen der Unentscheidbarkeit mächtiger Logiken werden selbst bei einer guten Repräsentation und bei der Verwendung starker Inferenzmaschinen viele Probleme nicht vollautomatisch gelöst werden können. Deduktionssysteme können aber auch dann wertvolle Unterstützung bei der Lösung bieten. Dazu ist es nötig, die Interaktionsmöglichkeiten von Deduktionssystemen in erheblichem Umfang zu erweitern.

Bei der Frage welches Deduktionsverfahren für ein bestimmtes Anwendungsfeld verwendet werden soll, sind zwar wichtige Teilantworten gefunden, aber die allgemeine Frage ist weitgehend offen. So bleibt beim Einsatz von Deduktionssystemen meist nur eine sehr sorgfältige Untersuchung, inwieweit sie für die speziellen Bedürfnisse geeignet sind.

Danksagung

Für sorgfältiges Korrekturlesen früherer Versionen und für viele hilfreiche Anregungen danken wir Norbert Eisinger, Michael Kohlhase und Renate Schmidt.

Literatur

[1] P. B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, **36**:414–432, 1971.

[2] P. B. Andrews, S. Issar, D. Nesmith, F. Pfenning. The TPS theorem proving system. In *10.CADE*, S.641–642, Springer Verlag, 1990.

[3] L. Bachmair, H. Ganzinger. On restrictions of ordered paramodulation with simplification. In *10.CADE*, S.427–441. Springer Verlag, 1990.

[4] J. Barwise. An introduction to first-order logic. In J. Barwise, Hrsg., *Handbook of Mathematical Logic*, chapter A.1, S.5–46. North-Holland Publishing Company, 1977.

[5] C. Beierle, U. Hedstück, U. Pletat, J. Siekmann. An order sorted predicate logic with closely coupled taxonomic information. LILOG-Report 86, IBM Deutschland, Scientific Center, 1989. erscheint in Artificial Intelligence.

[6] W. Bibel. *Deduktion.* Oldenbourg, 1992.

[7] K. Bläsius, H. Bürckert, Hrsg. *Deduktionssysteme.* Oldenbourg, 1987. Erweiterte und überarbeitete Neuauflage 1992.

[8] R. S. Boyer, J. S. Moore. *A Computational Logic.* Academic Press, 1979.

[9] N. G. de Bruijn. A survey of the project AUTOMATH. In J. Seldin, J. Hindley, Hrsg., *To H.B. Curry - Essays on Combinatory Logic, Lambda Calculus and Formalism*, S.579–606. Academic Press, 1980.

[10] A. Bundy. The use of explicit plans to guide inductive proofs. In *9.CADE.* Springer Verlag, 1988

[11] H. Bürckert. *Constraint resolution.* Springer Verlag, 1991.

[12] R. Caferra, N. Zabel. Extending resolution for model construction. In *JELIA 90, Workshop on Logics in AI*, S.153–170. Springer Verlag, 1990.

[13] C. Chang, R. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Computer Science and Applied Mathematics. Academic Press, 1973.

[14] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, **5**:56–68, 1940.

[15] R. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[16] R. Demolombe, L. Fariñas. An inference rule for hypothesis generation. In *12.IJCAI*, S. 152–157. Morgan Kaufmann, 1991.

[17] H.-D. Ebbinghaus, J. Flum, W. Thomas. *Einführung in die mathematische Logik.* Wissenschaftliche Buchgesellschaft, 2.Auflage, 1986.

[18] E. Eder. *Relative Complexities of First Order Calculi.* Vieweg, 1992.

[19] M. R. Genesereth, N. J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1987.

[20] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, **38**:173–198, 1931.

[21] L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, **15**:81–91, 1950.

[22] J. Hsiang, M. Rusinowitch. On word problems in equational theories. In *14.ICALP*, S.54–71. Springer Verlag, 1987.

[23] G. P. Huet. A mechanization of type theory. In *3.IJCAI*, S.139–146. Morgan Kaufmann, 1973.

[24] D. Hutter. Guiding induction proofs. In *10.CADE*, S.147–161. Springer Verlag, 1990.

[25] M. Kerber. How to prove higher order theorems in first order logic. In *12.IJCAI*, S.137–142. Morgan Kaufman, 1991.

[26] M. Kerber, A. Präcklein. Tactics for the improvement of problem formulation in resolution-based theorem proving. SEKI Report SR-92-09, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1992.

[27] D. Knuth, P. Bendix. Simple word problems in universal algebras. In I. Leech, Hrsg., *Computational Problems in Abstract Algebra*, S.263–297. Pergamon Press, 1970.

[28] M. Kohlhase. Order-sorted type theory I: Unification. SEKI Report SR-91-18, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1991.

[29] R. Kowalski. A proof procedure using connection graphs. *Journal of the ACM*, **22**, 1975.

[30] R. Letz. *First-Order Calculi and Proof Procedures for Automated Deduction.* Dissertation, Technische Hochschule Darmstadt, 1992. in Vorbereitung.

[31] R. Letz, J. Schumann, S. Bayerl, W. Bibel. SETHEO: A high performance theorem prover. *Journal of Automated Reasoning*, **8**(2):183–212, 1992.

[32] P. Lindström. On extensions of elementary logic. *Theoria*, **35**:1–11, 1969.

[33] D. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, 1978.

[34] W. McCune. Otter 2.0. In *10.CADE*, S.663–664. Springer Verlag, 1990.

[35] H. Ohlbach, J. Siekmann. The Markgraf Karl Refutation Procedure. In J. Lassez, G. Plotkin, Hrsg., *Computational Logic: Essays in Honor of Alan Robinson*, S.41–112. MIT Press, 1991.

[36] F. Oppacher, E. Suen. Controlling deduction with proof condensation and heuristics. In *8.CADE*, S.384–393. Springer Verlag, 1986.

[37] L. C. Paulson. Isabelle: The next 700 theorem provers. *Logic and Computer Science*, S. 361–386, 1990.

[38] P. Pritchard, J. Slaney. Computing models of propositional logics. In *10.CADE*, S.685. Springer Verlag, 1990.

[39] M. M. Richter. *Logikkalküle.* Teubner, 1978.

[40] G. Robinson, L. Wos. Paramodulation and tp in first order theories with equality. *Machine Intelligence*, **4**:135–150, 1969.

[41] J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12**(1):23–41, 1965.

[42] M. Schmidt-Schauß. *Computational Aspects of an Order Sorted Logic with Term Declarations.* Springer Verlag, 1989.

[43] J. Siekmann. Unification theory. *Journal of Symbolic Computation*, **7**:207–274, 1989.

[44] J. Siekmann et al. Ω-MKRP. SEKI Report, in Vorbereitung, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1992.

[45] R. Smullyan. *First-Order Logic*. Springer Verlag, 1968.

[46] M. Stickel. Theory resolution. *Journal of Automated Reasoning*, **1**(4):333–355, 1985.

[47] M. Stickel. Rationale and methods for abductive reasoning in natural-language interpretation. In *Proceedings of the International Scientific Symposium Natural Language and Logic*, S.233–252. Springer Verlag, 1990.

[48] C. Walther. *A Many-sorted Calculus based on Resolution and Paramodulation*. Pitman Ltd., 1987.

[49] D. Warren, L. Pereira, F. Pereira. Prolog – the language and its implementation compared with Lisp. *Sigplan Notices*, **12**(8), 1977.

[50] C. Weidenbach. A sorted logic using dynamic sorts. MPI-Report MPI-I-91-218, Max-Planck-Institut für Informatik, Saarbrücken, 1991.

[51] C. Weidenbach, H. Ohlbach. A resolution calculus with dynamic sort structures and partial functions. In *9.ECAI*, S.688–693. Pitman Publishing, 1990.

[52] H. Zhang, D. Kapur. First-order theorem proving using conditional rewrite rules. In *9.CADE*, S.1–20. Springer Verlag, 1988.