Kapitel I:
Geschichte und Anwendungen

(Jörg H. Siekmann)

It is reasonable to expect that the relationship between computation and mathematical logic will be as fruitful in the next century as that between physics and analysis in the past.

J. McCarthy, 1963

1 Einleitung

Die Entwicklung der mechanischen Rechenmaschinen von W. Schickard (1623), B. Pascal (1642) und G.W. Leibniz (1671) im siebzehnten Jahrhundert gilt als ein Meilenstein der Technik. Eine Rekonstruktion der ersten, im Original verlorengegangenen Maschine (von Wilhelm Schickard in Tübingen im Jahr 1623 gebaut) hat in unserem Fachbereich einen Ehrenplatz und wird nicht nur von den Erstsemestern gebührend bewundert. Was gibt diesen Maschinen ihre historische Bedeutung? Weder die zugrundeliegenden mathematischen Prinzipien noch die handwerklichen Fähigkeiten zur Herstellung der Mechanik einer solchen Maschine waren, gemessen am Stand der Kunst im siebzehnten Jahrhundert, etwas Ungewöhnliches – und trotzdem wissen wir über die Geschichte und Entwicklung dieser Maschinen heute besser Bescheid als über viele andere, als intellektuelle Einzelleistung vielleicht beeindruckendere Erfindungen der Zeit. Was macht also die Bedeutung aus?

Diese frühen, herausragenden Zeugnisse aus der langen Geschichte der Mechanisierung kognitiver Prozesse [McC79] hatten zwei wichtige, die Realisierung erst ermöglichende Voraussetzungen. Zum einen waren die Fähigkeiten in der Herstellung mechanischer Geräte und Uhren in Tübingen, Paris, Nürnberg und anderswo so weit fortgeschritten, daß die *technische Realisierung* einer solchen Rechenmaschine kein unüberwindbares Problem mehr bedeutete. Und zum anderen war das Rechnen mit Zahlen – das noch bei den Römern beispielsweise eine von ganz wenigen gutdotierten Spezialisten ausgeübte Kunst blieb – soweit *kalkülisiert*, daß es zu einer auch für den Laien relativ leichten und mechanischen Arbeit wurde.

Die Entwicklung der heutigen Deduktionssysteme hatte zwei analoge Voraussetzungen: Die Realisierung sehr schneller elektronischer Rechner einerseits und die Kalkülisierung der Logik, das heißt die Kalkülisierung einer wichtigen Fähigkeit des menschlichen Denkens, andererseits. Das Zusammenführen dieser beiden historischen Entwicklungen hatte eine fundamentale Bedeutung, deren Folgen wir heute in der Künstlichen Intelligenz (KI), für die das Gebiet der Deduktionssysteme zu einer Grundlagendisziplin geworden ist, aber auch in der traditionellen Informatik beobachten können: Programmieren und Rechnen auf dem Computer beziehen sich nicht nur auf die Logik als Grundlagenwissenschaft – wie von J. McCarthy in den 60er Jahren dieses Jahrhunderts antizipiert – sondern sie *sind* logische Deduktion. In Anlehnung an R.

Kowalskis berühmte Gleichung ausgedrückt:

COMPUTATION = DEDUCTION + CONTROL

Diese Einsicht hat unsere Vorstellung vom Computer und von Berechnungen, die man für diese oder jene Anwendung darauf anstellen mag, tiefgreifend verändert [Kow79, HS85, FGCS84] und wird von vielen – wegen ihrer weitreichenden wissenschaftlichen und technischen Auswirkungen – als eine der großen Entdeckungen dieses ausgehenden Jahrhunderts angesehen.

Während die Entwicklung der Computer von Zuses Z1 aus dem Jahre 1936 bis zur heutigen LISP-Maschine (Lambda-Kalkül) beziehungsweise den Rechnern der 5. Generation (Hornlogik), den meisten Lesern dieses Bandes wahrscheinlich vertraut ist, hat die zweite historische Voraussetzung, nämlich die Entwicklung mechanisierbarer Logikkalküle, noch bei weitem nicht den gebührenden Platz in unserem Bildungssystem gefunden.

2 Frühgeschichte

*... broadly conceived, the history of automated deduction is really the history of the **idea** of mechanizing human thought.*

M. Davis, 1979

Wenn wir von den frühen logikorientierten Arbeiten der griechischen Philosophen, die in Aristoteles' Werk einen ersten Höhepunkt fanden [Rus46], und ebenfalls von der mittelalterlichen Scholastik und deren Beiträgen zur Logik einmal absehen, beginnt die Geschichte der Automatisierung des menschlichen Denkens mit den Beiträgen von R. Descartes und G. W. Leibniz im siebzehnten Jahrhundert. Descartes' Entdeckung, daß die klassische griechische Geometrie allein mit algebraischen Methoden entwickelt werden kann, war eine Einsicht, die nicht nur die Mathematik stark beeinflußt hat, sondern auch für die Entwicklung der Deduktionssysteme, das heißt für den Traum, menschliches logisches Denken auf einer Maschine nachvollziehen zu können, bedeutsam war: Was für Euklid noch das Resultat schöpferischer Intelligenz und mathematischen Könnens der Geometer war, konnte nun weitgehend automatisch durchgeführt werden. Durch die Einführung eines Koordinatensystems und durch das Aufstellen der entsprechenden Gleichungen können die notwendigen geometrischen Konstruktionen auf algebraische Manipulationen zurückgeführt und dann im Prinzip mechanisch ausgeführt werden. Descartes hat sehr wohl diesen Aspekt seiner Arbeit gesehen und in den Vordergrund gestellt: „… it is possible to construct all the problems of ordinary geometry by doing no more than the little [nämlich die vier von ihm vorgeschlagenen Operationen, d. A.] covered in the four figures that I have explained. This is one thing which I believe the ancients did not notice, for otherwise they would not have put so much labor into writing so many books in which the very sequence of the propositions showed that they did not have a sure method of finding all ..." [Des37]. Seine von zeitgenössischen Mathematikern heftig befehdete Vorstellung war, daß damit die Geometrie ihren Reiz für den schöpferischen Mathematiker verloren habe, da sie ja nun mechanisierbar geworden sei.

Was R. Descartes für die Euklidsche Geometrie getan hatte, hoffte G. W. Leibniz auf alles menschliche Denken erweitern zu können und schlug dazu, beeinflußt von der Idee der „Ars Magna“ des Ramonus Lullus aus dem 14. Jahrhundert, ein sehr weitreichendes und langfristiges (es sollte über dreihundert Jahre dauern) Forschungsprojekt vor: Nämlich erstens eine universelle formale Sprache, die „lingua characteristica“, zu entwickeln, in der jegliche Wahrheit formuliert werden könne, und zweitens einen Kalkül, den sogenannten „calculus ratiocinator“, für diese Sprache. So wie man die folgende Hau-Aufgabe aus der Zeit 1800 v. Chr. von einer natürlichen Sprache in eine andere übersetzen kann, z.B. aus der demotischen Schrift:

in die eventuell leichter lesbare Hieroglyphenschrift:

oder in das möglicherweise doch besser verstehbare, mittelalterliche Deutsch:

Form der Berechnung eines Haufens, gerechnet anderthalb mal zusammen mit

vier. Er ist gekommen bis zehn. Der Haufe nun nennt sich?

so kann man sie auch in eine heute üblichere *formale Sprache* übersetzen:

$$1\frac{1}{2} \cdot x + 4 = 10$$

in der man mit rein syntaktischen Operationen eines *Kalküls* rechnen kann, hier z. B.

$$y + a = b \quad \Rightarrow \quad y = b - a \qquad \text{und} \qquad c \cdot z = d \quad \Rightarrow \quad z = \frac{d}{c}$$

In der gleichen Weise wollte Leibniz eine natürlichsprachliche Beschreibung auch über Sachverhalte, die nicht aus der Zahlentheorie kommen, in eine formale Sprache („lingua characteristica“) und einen dazugehörigen Kalkül („calculus ratiocinator“) übersetzbar machen.

Obwohl Leibniz' technische Beiträge zu diesem Forschungsprojekt aus heutiger Sicht vergleichsweise gering angesehen werden, hatte er auf seine Nachwelt sehr einflußreiche Vorstellungen von dem weitreichenden Ziel und der Bedeutung, die diesem Projekt zugrunde lag. Im Vergleich zu den griechischen Mathematikern, die dadurch Unsterblichkeit erlangten, daß sie die mathematischen Gesetzmäßigkeiten geometrischer Körper untersuchten, obwohl diese in der Natur nicht vorkommen und wenig praktische Anwendungen haben, sagte Leibniz, um wieviel bedeutender müsse ein mathematisches Genie eingestuft werden, das die logischen Gesetzmäßigkeiten, die dem menschlichen Denken zugrundeliegen, erforschen könne. Leibniz hatte die Vorstellung, daß ein solcher Kalkül, wenn einmal entwickelt, auch – einer Rechenmaschine vergleichbar – mechanisierbar sei, und daß auf diese Weise dem menschlichen Denken alle langweilige Arbeit erspart werden könne. „Der Verstand wird befreit von der Vorstellung der Dinge selbst, und doch wird die Berechnung korrekt herauskommen“. Die Vorstellung, die Leibniz von einer solchen universellen Sprache und dem damit verbundenen Kalkül hatte, gipfelt in der herrlichen Beschreibung zweier Menschen guten Willens, die, in einen philosophischen Disput verwickelt und auf der Suche nach der Wahrheit, ihre Argumente in die „lingua characteristica“ übersetzen und dann nicht wie zwei Philosophen disputieren, sondern wie zwei Computerexperten sagen: „Calculemus – Rechnen wir es einfach aus“. („Quo facto, quando orientur controversiae, non magis disputatione opus erit inter duos philosophos, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus.“ [Ger90]).

Ein wirklicher Kalkül im Sinne von Leibniz wurde erst zwei Jahrhunderte später von A. de Morgan und G. Boole entwickelt und ist den meisten von uns als *Aussagenlogik* schon seit der Schulzeit vertraut. Diese Boolesche Algebra, zunächst entwickelt, um Mengen zu beschreiben, hat die von Boole ebenfalls gesehene heute übliche aussagenlogische Interpretation und wurde von ihm als eine Fortsetzung des Leibnizschen Programms angesehen. Wie weit die Booleschen Arbeiten die Logik mechanisierbar machen wollten, läßt sich besonders schön an der Maschine zeigen, die der Wirtschaftswissenschaftler und Logiker Stanley Javins 1869 entwickelte, um Boolesche Ausdrücke zu evaluieren – eine Maschine, die einer Registriermaschine in den Büros des vorigen Jahrhunderts nicht unähnlich war.

Die wichtigsten Beiträge zu der Entwicklung der mathematischen Logik, wie sie auch den heutigen Deduktionssystemen zugrundeliegt, sind die Arbeiten von Gottlob Frege, insbesondere sein Buch „Begriffsschrift“, das er als die Erfüllung und Ausarbeitung des Leibnizschen Forschungsprogrammes (allerdings eingeschränkt auf die Mathematik) sah. Auf den Vorwurf E. Schröders [Sch82], er habe die Booleschen Arbeiten nicht genügend berücksichtigt, antwortet Frege: „Ich wollte nicht (wie Boole) eine abstracte Logik in Formeln darstellen, sondern einen Inhalt durch geschriebene Zeichen in genauerer und übersichtlicherer Weise zum Ausdruck bringen, als es durch Worte möglich ist. Ich wollte in der That nicht einen blossen ‚calculus ratiocinator‘, sondern eine ‚lingua characterica‘ [sic] im leibnizischen Sinne schaffen, wobei ich jene schlussfolgernde Rechnung immerhin als einen nothwendigen Bestandteil einer Begriffsschrift anerkenne.“ [Fre82].

Freges „Begriffsschrift“ enthält die erste vollständige Entwicklung desjenigen Anteils der mathematischen Logik, den wir heute als *Prädikatenkalkül erster Stufe* bezeichnen. Die enorme Bedeutung dieser Arbeit wird erst deutlich, wenn man die Verwirrung über die unterschiedlichsten Interpretationen sieht, die seine Zeitgenossen und Vorgänger einer mathematischen Logik gaben; insbesondere die Diskussion darüber, ob die mengentheoretische Interpretation oder die propositionale Interpretation der Booleschen Algebra das Primäre sei. In der „Begriffsschrift“ wird die aussagenlogische Interpretation als fundamental angenommen und der Entwicklung der weiteren Konstrukte zugrundegelegt. Diese weitere Entwicklung betrifft den Gebrauch von Quantoren und den gesamten *funktionalen* Aufbau der Logik, so wie er uns heute vertraut ist und jedem Logikbuch als Prinzip zugrundeliegt: Nämlich der Gebrauch Boolescher logischer Verknüpfungen einerseits und der Quantoren, Relationen und Funktionen andererseits. Als Inferenzregel wird der „modus ponens“ verwendet. Damit war das alte Leibnizsche Programm, menschliches logisches Denken auf rein syntaktisches Vorgehen zu reduzieren, für einen wichtigen Teil realisiert und abgeschlossen.

Die „Begriffsschrift“ ist nicht nur deshalb so relevant, weil hier ein logischer Kalkül entwickelt und vollständig formal aufgebaut wurde, sondern auch wegen der methodischen Vorgehensweise, in der Syntax und intendierte Semantik einer formalen Sprache getrennt und explizit entwickelt wurden. Die „Begriffsschrift“ war damit nicht nur der Vorgänger der heutigen mathematischen Logik, sondern Vorbild aller formalen Sprachen, einschließlich aller Programmiersprachen.

So bedeutend die Arbeiten von Frege waren, so wenig wurden sie von seinen Zeitgenossen anerkannt. Möglicherweise war der Fortschritt gegenüber dem bis dahin Erreichten doch zu groß, um ihre volle Tragweite in jener Zeit zu erkennen. Hinzu kam, daß die syntaktische Genauigkeit und Präzision in einer graphisch orientierten Sprache ausgeführt wurde, die das Buch auch heute noch schwer zugänglich macht.

Es waren die Arbeiten von G. Peano, die unsere heutige Notation geprägt haben. Peano sah dies als die weitere Fortsetzung des Leibnizschen Programms, das er nicht nur auf die Mathematik, sondern auf alle Wissenschaften anwenden wollte: „I think that the propositions of any science can be expressed by these signs of logic alone, provided we add signs representing the objects of that science.“ ([Pea89], zitiert nach [Hei67], S. 86). Obwohl Peano, aufbauend

auf Frege, den Grundformalismus für eine solche universelle Sprache entwickelte und damit die prinzipielle Möglichkeit schuf, die als zu vage empfundene natürliche Sprache zu eliminieren, war es doch so, daß die Deduktionsschritte selbst wieder natürlichsprachlich beschrieben wurden. Dies war sicherlich ein Rückschritt gegenüber Frege, der mit Hilfe des „modus ponens“ einen vollständigen formalen Kalkül entwickelt hatte.

Die Formalisierung des logischen Denkens war zunächst für die Mathematik am wichtigsten und hatte hier den stärksten Einfluß. Allerdings wurde dieser Einfluß durchaus nicht von allen Mathematikern positiv gesehen. Ganz im Gegenteil, es gab scharfe, bisweilen polemische Kritik und offene Gegnerschaften. Einer der schärfsten Kritiker war der große Mathematiker Henri Poincaré, den die Arbeiten von Peano ganz besonders reizten. Der Typ der Kritik läßt sich sehr schön an folgendem Zitat sehen: „It is difficult to admit that the word *if* acquires when written $\Rightarrow$, a virtue it did not possess when written if.“ (zitiert nach [Poi52]).

Poincaré sah ganz deutlich, daß, wenn die Arbeiten von Frege, Peano und anderen erfolgreich wären, sich durchaus die Möglichkeit einer Mechanisierung des logischen Denkens und damit insbesondere der Mathematik ergeben würde. Dies war für ihn jedoch kein Vorteil dieser Vorgehensweise oder gar ein großes Ziel, das es langfristig zu erreichen gälte, sondern ganz im Gegenteil schien es ihm die Hoffnungslosigkeit des gesamten Unterfangens aufzuzeigen, da es doch alles, was schön und kreativ am menschlichen Denken sei, zu zerstören schien. Hier ist eine der besonders erbitterten Äußerungen: „Thus it will be readily understood that in order to demonstrate a theorem, it is not necessary or even useful to know what it means. We might replace geometry by the r*easoning piano* imagined by Stanley Jevons; or if we prefer, we might imagine a machine where we should put in axioms at one end and take out theorems at the other, like that legendary machine in Chicago where pigs go in alive and come out transformed into hams and sausages. It is not more necessary for the mathematician than it is for these machines to know what he is doing.“ (zitiert nach [Poi52]).

Dieses negative Klima, in dem trotzdem solche grundlegenden Arbeiten entstanden, war durch den Zweifel an den Grundlagen der klassischen Mathematik, der ihr zugrundeliegenden Logik und durch die Kritik an dieser, geprägt. Die prominentesten Vertreter unter den Kritikern, L. J. Brouwer und H. Weyl, warfen den klassischen Mathematikern vor, in einer Welt der Objekte zu leben, deren Existenz, weil nicht konstruktiv und explizit hergeleitet, in keiner Weise gesichert sei, und daß damit weite Teile der Mathematik möglicherweise einer klaren Grundlage oder gar der Existenzberechtigung entbehren. Einer der bekanntesten Vertreter der alten Schule war D. Hilbert, der daraufhin Brouwer anklagte, große Bereiche der Mathematik einfach zu verbieten. Nun war Hilbert nicht ein Mann der Polemik, sondern er nahm die Kritik an den mangelnden Fundamenten der Mathematik ernst und entwarf ein Programm, das außerordentlich einflußreich war. Dieses Hilbertsche Programm sollte die Mathematik auf eine klare logische Basis stellen und hatte zum Ziel, die klassische Mathematik zu axiomatisieren und formal in einem logischen Kalkül zu entwickeln. Ein Programm, dessen Möglichkeit zumindestens durch N. Whitehead und B. Russel glaubwürdig geworden war, die die gewaltige Arbeit auf sich genommen hatten, in der dreibändigen ‚Principia Mathematica‘ (1910-1913), zu zeigen, daß man klassische Mathematik sehr wohl in einem solchen Kalkül entwickeln könne. Die Kritik der Intuitionisten und anderer sollte dann durch einen

konstruktiven Konsistenz- und Vollständigkeitsbeweis des gesamten Kalküls abgewiesen werden.

Dieses Programm hat die heutige Beweistheorie, die Modelltheorie und den Begriff der Metamathematik geprägt; es wurde übrigens unabhängig von D. Hilbert und seiner Schule auch von dem amerikanischen Logiker E. Post in Angriff genommen.

Der Grundlagenstreit zwischen den klassischen Mathematikern einerseits und der neuen Schule der Konstruktivisten andererseits hat auch auf unsere heutige praktische Arbeit wieder einen großen Einfluß: Insbesondere durch die neueren Arbeiten von P. Martin-Löf und anderen (siehe zum Beispiel [HS85]), die eine interessante Mechanisierung konstruktivistischer Logik ermöglichen, hat diese Schule einen starken Einfluß auf die Entwicklung von Deduktionssystemen und von logischen Programmiersprachen genommen. Beispielsweise hat das von R. Constable an der Cornell University entwickelte, auf Martin-Löfs Typentheorie aufbauende, NUPRL-System das Ziel, mathematisches Beweisen zu erleichtern, so daß jeder vom Benutzer eingegebene Beweisschritt formal auf seine Korrektheit hin überprüft wird. Eingebettet in komfortable, auf die Mathematik zugeschnittene Textsysteme, werden solche Systeme sicher zu unentbehrlichen Hilfsmitteln in der Mathematik werden und herkömmliche Arbeitsweisen verdrängen – so wie die heutigen Textsysteme auf dem PC die alte Schreibmaschine obsolet machten.

Eine weitere Schlüsselrolle in der Automatisierung deduktiven Denkens spielen die Arbeiten von T. Skolem, in denen er eine systematische Vorgehensweise entwickelte, durch die man die Erfüllbarkeit einer beliebig vorgegebenen Formelmenge nachweisen kann. In zwei Arbeiten, die 1920 und 1928 erschienen, wurden die Elimination von Quantoren (durch Skolemfunktionen) und eine Konstruktion eingeführt, die uns heute unter dem historisch nicht ganz korrekten Namen Herbrand-Universum vertraut ist. Für die so transformierte Formelmenge entwickelte er eine mechanische Beweisprozedur, deren Grundgedanke bis heute Basis aller automatischen Deduktionssysteme geblieben ist.

Ebenfalls im Jahre 1928 erschien das Buch „Grundzüge der theoretischen Logik“ von D. Hilbert und W. Ackermann im neugegründeten Julius-Springer-Verlag. In diesem Buch wurden Gedanken, die auch für die Automatisierung der Logik wichtig sind, erstmals einem größeren, mathematisch interessierten Leserkreis vorgestellt, und eine Axiomatisierung des Prädikatenkalküls mit einer Reihe von offenen Problemen bezüglich dieser Axiomatisierung angegeben. Erstens wurde die Frage nach der Vollständigkeit dieser Axiomatisierung gestellt, d.h. ob man jede gültige prädikatenlogische Aussage aus dieser Axiomatisierung folgern könne. Zweitens wurde das „Entscheidungsproblem“ eingeführt, das Problem also, ob es einen Algorithmus gebe, der für jede beliebige vorgegebene prädikatenlogische Aussage entscheiden könne, ob diese Aussage *wahr* oder *falsch* sei. Falls es so einen Algorithmus gäbe, wäre das Hilbertsche Programm damit erfolgreich abgeschlossen: Denn nun ist es ja möglich, eine beliebige mathematische Aussage in diesem so geschaffenen Kalkül zu formulieren, und mit Hilfe des Entscheidungsalgorithmus’ festzustellen, ob die Aussage stimmt. Die Vollständigkeit des Hilbert/Ackermann-Kalküls wurde 1930 von K. Gödel in seiner Doktorarbeit gezeigt.

Dann aber riß die Kette der negativen Resultate nicht ab und schien mehr und mehr das gesamte

Hilbertsche Programm in Frage zu stellen. Ein Jahr später, 1931, zeigte Gödel in einer epochalen Arbeit, daß man für ein hinreichend mächtiges formales System (das mindestens die Arithmetik enthält) die Konsistenz dieses Systems nicht, wie im Hilbertschen Programm gefordert, innerhalb des Systems beweisen kann. Und zum zweiten gab er eine Konstruktion an, nach der ebenfalls für jedes hinreichend starke formale System ein Satz konstruiert werden kann, der innerhalb dieses Systems weder beweisbar noch widerlegbar ist. Zwar kann man mit mächtigeren Methoden die Korrektheit bzw. die Falschheit eines solchen Satzes wieder nachweisen, aber dann gibt es für dieses mächtigere System wiederum einen Satz, dessen Wahrheit oder Falschheit nicht nachweisbar ist.

Wenig später wurde die Unentscheidbarkeit des von Hilbert so genannten „Entscheidungsproblemes“ unabhängig von Alan Turing und Alonzo Church gezeigt. 1936 führte Turing in seiner berühmten Veröffentlichung, in der er die nach ihm benannten abstrakten Maschinen konstruierte, das Entscheidungsproblem auf das Halteproblem dieser Maschinen zurück und zeigte, daß die Frage, ob eine solche Maschine anhält, unentscheidbar ist. Im selben Jahr zeigte Alonzo Church für seinen Lambda-Kalkül, daß die Gültigkeit eines Lambda-Ausdrucks ebenfalls unentscheidbar ist. Damit schien der Traum, etwas für die gesamte Mathematik zu finden, was dem Rechnen mit Zahlen entspricht, für alle Zeit zerstoben – und damit natürlich insbesondere der Traum mechanischer Deduktionssysteme.

So einschneidend und fundamental diese negativen Ergebnisse waren, so wenig haben sie letztlich die Hoffnung auf eine Formalisierung der Mathematik zerstören können. Warum? Nun, die Arbeiten von Skolem hatten gezeigt, daß für eine vorgegebene *gültige* Aussagenmenge dieser Nachweis auch mechanisch gefunden werden kann. Ganz ähnlich lautete das Hauptresultat der Doktorarbeit von J. Herbrand aus dem Jahre 1930, daß nämlich für einen korrekten mathematischen Satz diese Korrektheit in der Tat auch nachgewiesen werden kann. Die dazu verwendete Methode wurde zwanzig Jahre später für die frühen Deduktionssysteme direkt auf dem Rechner implementiert. Diese *Semientscheidbarkeit* des Prädikatenkalküls ist die formale Rechtfertigung aller Deduktionsysteme, d.h. wenn ein Theorem wahr ist, kann dies auch in endlich vielen Schritten nachgewiesen werden, wenn nicht, nun dann gibt es zwei Möglichkeiten: Entweder ein Nachweis der Inkorrektheit gelingt in speziellen Fällen trotzdem, oder das Programm läuft ohne zu terminieren weiter und man weiß es eben nicht.

3 Erste Deduktionssysteme

Als Anfang der 50er Jahre die ersten leistungsfähigen Computer zur Verfügung standen, lag der Gedanke in der Luft, dieses neue Werkzeug auch zur Automatisierung der Logik einzusetzen, die, wie wir gesehen haben, einen weiten Weg in der Entwicklung mechanisierbarer Kalküle gegangen war. Es erforderte den besonderen Wissenschaftler, der einerseits mit den geistigen Traditionen, die aus der Logik kamen, vertraut war, und der sich andererseits in der modernen Welt der Computer auskannte, um beide Welten miteinander in Verbindung zu bringen und weiterzuführen.

Zwei Computerprogramme zum Beweisen mathematischer Sätze wurden in diesen frühen 50er Jahren entwickelt und sind durch ihre Gegensätzlichkeit bis heute typisch für die beiden grundsätzlichen Vorgehensweisen in unserem Gebiet geblieben. Das erste Programm wurde 1954 von Martin Davis in Amerika am Institute for Advanced Studies auf einer „Johniac" (einem Röhrencomputer, wie sie damals den Wissenschaftlern zu Verfügung standen) entwickelt. Dieses Programm realisierte ein von Presburger entwickeltes Entscheidungsverfahren für die nach ihm benannte Arithmetik positiver Zahlen. Ein großer Augenblick im Lebenszyklus dieses Programms war der Beweis, daß die Summe zweier gerader Zahlen wieder eine gerade Zahl ist – der erste computergenerierte Beweis einer mathematischen Aussage in der Geschichte.

Ein zweites, ganz anders arbeitendes Programm, das eine Reihe von Sätzen aus B. Russels und N. Whiteheads „Principia Mathematica" beweisen konnte, wurde ebenfalls Anfang der 50er Jahre von A. Newell, J. C. Shaw und H. Simon entwickelt. Dieses Programm orientierte sich an der Vorgehensweise von Menschen und versuchte allgemeine heuristische Verfahren und psychologische Vorgehensweisen zu simulieren. Nicht zuletzt deshalb war es einer der herausragenden Beiträge auf der ersten Konferenz zum damals noch nicht existierenden Fach „Künstliche Intelligenz" (KI), der Dartmouth Conference im Jahre 1956, die als die Geburtsstunde der KI gilt. Schon auf dieser Konferenz flammte eine Diskussion auf, die bis heute die Leidenschaften in unserem Gebiet immer neu zu entfachen vermochte: Soll man versuchen, die mathematische Logik so gut es eben geht auf dem Computer zu realisieren, oder soll man, wie es Newell und Simon getan hatten, versuchen, menschliche psychologische Mechanismen zu simulieren? Diese Debatte, die seit über 30 Jahren nicht nur das Gebiet der Deduktionsysteme, sondern die gesamte KI-Forschung bewegte, hat in unserem Gebiet zu einer Einsicht geführt, die M. Minsky bereits 1961 so formulierte: „It seems clear that a program to solve real mathematical problems will have to combine the mathematical sophistication of Wang with the heuristic sophistication of Newell, Shaw and Simon." [Min61]. Die Frage, wie dies genau zu bewerkstelligen sei, ist bis zum heutigen Tage der rote Faden geblieben, der sich durch die Geschichte der Deduktionssysteme zieht. Eine Frage, die bis heute keine abschließende Antwort gefunden hat.

Ein interessanter Gedanke, wie in einem Spezialfall die Kombination der beiden Welten zu bewerkstelligen sei, wurde 1956 von M. Minsky selbst vorgeschlagen, nämlich ein Diagramm in einem Geometriebeweiser als Modell zu verwenden. Diese Idee wurde von H. Gelernter und Mitarbeitern aufgegriffen und in einem einflußreichen Projekt zum automatischen Beweisen geometrischer Sätze realisiert. Dieses Beweisprogramm durchsuchte systematisch alle Lösungswege, diese Suche wurde aber durch Hilfsdiagramme so gesteuert, daß viele der nutzlosen Zweige des Suchraumes beizeiten abgeschnitten werden konnten.

Die relative Ineffizienz der logikorientierten Methoden einerseits und der Erfolg der Programme von Gelernter, Newell und Simon andererseits, verstärkten den Eindruck, daß logikorientierte Methoden wohl doch keine erfolgversprechende Basis seien, ein Eindruck, dem sich der junge Wissenschaftler Hao Wang vehement widersetzte. 1958 entwickelte er das erste logikorientierte Programm zum automatischen Beweisen der Firma IBM, und später in den Jahren 1959-1963 weitere Versionen am Bell-Lab. Dies waren damals mit großem Abstand die erfolgreichsten

Programme, sie konnten über 350 Sätze der „Principia Mathematica“ beweisen (relativ einfache Sätze des reinen Prädikatenkalküls mit Gleichheit) und gaben dem Gebiet einen ersten Aufschwung. Für diese Arbeiten hat Hao Wang den ersten „Milestone Award for Automated Theorem Proving“ auf dem 1983er Jahrestreffen der American Mathematical Society erhalten.

Im Jahr 1954 trafen sich Logiker und Informatiker zu einem „Summer Institute for Symbolic Logic“ an der Cornell University, USA, und diskutierten in einer von den Teilnehmern als außerordentlich lebendig und stimulierend geschilderten Atmosphäre viele der teilweise bis heute ungelösten Forschungsfragen. Auf diesem Treffen trug der Logiker Abraham Robinson in einem kurzen Referat den Gedanken vor, daß die zusätzlichen Punkte, Linien oder Kreise die zur Lösung eines Geometrieproblemes konstruiert werden müssen, als Punkte im sogenannten Herbrand Universum (wie wir es heute nennen) aufgefaßt werden können. Man könne daher auf die geometrischen Konstrukte verzichten und gleich Herbrands Methode verwenden.

Dieser Gedanke war sehr einflußreich und hat das Automatische Beweisen in den frühen Jahren geprägt. Eines der ersten Programme, das diese Idee realisierte, wurde von Martin Davis und Hillary Putnam in den Jahren 1960-1962 implementiert. Es bestand im wesentlichen aus zwei Teilen: Zum einen aus den Programmteilen, die das Herbrand Universum systematisch generierten und entsprechend in die Formeln einsetzten, und zum zweiten aus einem Programmteil, das die so gewonnenen aussagenlogischen Formeln auswertete. In diesem zweiten Programmteil sahen die Autoren damals ihren wesentlichen Beitrag. Es zeigte sich jedoch bald, daß das Problem nicht so sehr in diesem zweiten Teil, also dem Nachweis der Tautologieeigenschaft lag, sondern in dem ersten Teil, dem systematischen Einsetzen aller Herbrand-Instantiierungen.

1960 veröffentlichte D. Prawitz, ein schwedischer Logiker, der ebenfalls sehr früh die Bedeutung der Computer für die Logik (und umgekehrt) erkannt hatte, ein Verfahren, in dem die bis dahin unsystematische Einsetzung der Herbrandterme geschickter gesteuert werden konnte, und zwar durch einen Mechanismus, den wir heute *Unifikation* nennen. Dies wurde sofort von Martin Davis aufgegriffen und in einem Computerprogramm, das auf der sogenannten Linked-Conjunct-Methode basierte, weiterentwickelt und implementiert. In diesem Programm war die uns heute vertraute Klauselnormalform ebenso fester Bestandteil wie ein von D. McIlroy im November 1962 am Bell-Telephone-Laboratories entwickelter Unifikationsalgorithmus. Es war das erste Programm [Dav63, CDHI64], das die offensichtliche Schwäche der Herbrand-Prozeduren überwand und dafür die Klauselnormalform und einen Unifkationsalgorithmus einsetzte.

Unabhängig davon arbeitete eine Gruppe von Wissenschaftlern in der Nähe von Chicago am Argonne National Lab ebenfalls an dem Beweisen mathematischer Sätze durch den Computer. Diese Gruppe wurde von G. Robinson geleitet und hatte so talentierte Mitarbeiter, wie D. Carson, J. A. Robinson und L. Wos – Namen, die heute jeder in unserem Gebiet kennt. Die ersten hier entwickelten Programme basierten auch auf Herbrands Methode und man erkannte ebenfalls sehr rasch die Schwäche dieser Vorgehensweise, nämlich das undifferenzierte Einsetzen aller Herbrand Instantiierungen. In dieser Arbeitsgruppe wurden die Veröffentlichungen von M. Davis und H. Putnam und natürlich die Arbeit von D. Prawitz heftig

diskutiert, und man war auf der Suche nach einem allgemeinen maschinenorientierten logischen Prinzip, das diese Gedanken in einer einzigen Inferenzregel vereinigen würde. Eine solche Regel wurde im Laufe der Jahre 1963-64 gefunden und dann 1965, in dem für unser Gebiet epochemachenden Papier von J.A. Robinson „A machine oriented logic, based on the resolution principle“, im Journal der ACM veröffentlicht [Rob65].

Das Resolutionsprinzip ist heute fester Bestandteil praktisch jeder Logik- und Informatikausbildung geworden und schien mit einem Mal alle Schwächen der früheren Beweisprozeduren zu eliminieren, ohne selbst neue einzuführen. Es hat ganz wesentlich zu der Begeisterung für dieses Gebiet und zu der zentralen Rolle beigetragen, die die Deduktionssysteme für gut ein Jahrzehnt in der nun aufkommenden Künstlichen Intelligenz spielen sollten.

4 Die Zeit nach 1965

Trotz der anfänglichen Euphorie, die das Resolutionsprinzip auslöste, wurde bald deutlich, daß mit einer Ableitungsregel allein keine wirklich leistungsfähigen Beweiser zu bauen seien: Die Vorhersage von Hao Wang, daß in wenig mehr als einer Dekade automatische Beweissysteme menschliche Mathematiker in der Beweissuche ersetzen würden, schien ferner denn je. Wieder gab es die beiden großen Hauptrichtungen mit den entsprechenden Schulen, die die Geschichte des automatischen Beweisens von Anfang an geprägt hatten. Während die einen – das Hauptlager, wenn man die Anzahl der beteiligten Wissenschaftler zählt – weiterhin auf das Resolutionsverfahren und dessen Verfeinerungen setzten, sahen andere die Zukunft dieses Gebietes in der Weiterentwicklung heuristischer Methoden und in dem Versuch, menschliche Mathematiker zu imitieren.

In dem ersten Lager, das *logikorientierte* Methoden auf seine Fahnen geschrieben hatte, und das insbesondere am Argonne National Lab, USA, unter Leitung von Larry Wos und an der University of Edinburgh, Großbritannien, unter der Leitung von Bernhard Meltzer seine Hochburgen fand, wurden Verbesserungen, sogenannte Refinements, für das Resolutionsprinzip entwickelt. Diese Refinements sind Prädikate, die den Suchraum einschränken. Die Standard-Lehrbücher wie z. B. [CL73, Lov78] behandeln in erster Linie diese Forschungsergebnisse, die im wesentlichen von 1965 bis Mitte der 70er Jahre entstanden und zu einigen Dutzend solcher Verfeinerungsstrategien führten.

Ein etwas anderer Forschungsschwerpunkt, jedoch aus demselben Lager der logikorientierten Methoden, hat die Inferenzregel selbst zum Gegenstand seiner Untersuchung gemacht. Es wurde bald deutlich, daß ein einzelner Resolutionsschritt, entgegen der ursprünglichen Motivation, viel zu feinkörnig ist. Inferenzregeln, wie zum Beispiel die Hyperresolution oder die UR-Resolution versuchen daher, Makroschritte einzuführen, indem sie eine bestimmte Sequenz von Schritten auf einmal durchführen. Dieser wichtige Gedanken wird in diesem Buch in den Vordergrund gestellt und war die Motivation für Mark Stickels Theorieresolution, die im zweiten Kapitel behandelt wird.

Bei den graphorientierten Beweisverfahren, wie dem Klauselgraphverfahren [Kow75], geht es darum, die Grunddatenstruktur, bisher eine *Menge* von Klauseln, mit mehr Information

anzureichern, beispielsweise dadurch, daß die potentiellen Resolutionsschritte explizit in einer Graphen-Struktur dargestellt werden. Ebenso wie die Unifikation zwar die zentrale Operation der Resolution ist, jedoch nicht auf diese beschränkt blieb, sondern heute in allen vernünftigen Deduktionskalkülen zu finden ist, so stellten sich auch diese graphorientierten Verfahren als ein fundamentaler Mechanismus heraus, den man heute in fast allen neueren, für das automatische Beweisen entwickelten, Kalkülen in der einen oder anderen Form wiederfinden kann.

Die neuesten Beweiskalküle, die Matrix- oder Konnektionsverfahren, wurden unabhängig von Peter Andrews an der Carnegie-Mellon University und von Wolfgang Bibel an der Universität München entwickelt und gelten derzeit als die aussichtsreichsten Kalküle für eine noch radikalere Mechanisierung der Logik. Während den alten Verfahren noch eine gewisse Vorstellung von einem menschlichen Beweis, der aus einer wohldefinierten Abfolge von Schritten besteht, zugrunde lag, brechen diese Verfahren vollständig mit diesen logischen Traditionen und sind ausschließlich maschinenorientiert. Die Beweissuche vollzieht sich implizit in einer bestimmten Datenstruktur, einer sogenannten Matrix, und der so gefundene Beweis hat mit unserer herkömmlichen Vorstellung davon eigentlich nicht mehr viel gemein – er kann jedoch wieder in einen herkömmlichen Beweis transformiert werden.

Das zweite Lager, das *„Simulate People“* auf seine Fahnen geschrieben hatte, war ebenfalls nicht untätig: Die fast monolithische Hinwendung zu uniformen logikorientierten Beweisverfahren (in der überwältigenden Mehrzahl vom Resolutionstyp) forderte bald scharfe Kritik heraus, inbesondere von Marvin Minsky am Massachusetts Institute of Technology. Anfang der 70er Jahre hatte einer seiner Promotionsstudenten, Carl Hewitt, eine Arbeit über eine Programmiersprache PLANNER geschrieben, die eine totale Attacke auf das Gebiet und eine schwere Grundlagenkrise nicht nur im Bereich der Deduktionssysteme, sondern in der gesamten Künstlichen Intelligenz auslöste. Bis zu diesem Zeitpunkt waren uniforme Verfahren (wie GPS, STRIPS u.a.) der unumstrittene Forschungsgegenstand in der KI. Am Beispiel der Deduktionssysteme, die bis dahin eine relativ unangefochtene und vornehme Stellung innerhalb der KI einnahmen, wurde nun Kritik geäußert und ein Gegenangriff eingeleitet, der sowohl das Gebiet der Deduktionssysteme wie die KI als Ganzes verändern sollte.

Das Argument war, kurz zusammengefaßt, daß ein einziges Verfahren niemals ausreichen könne, wirkliche Intelligenz zu realisieren, sondern daß sehr viel mehr Komponenten eine Rolle spielen müßten. Insbesondere sei eine geistige Haltung verwerflich, die explizite Tips und Wissen in einem Programm als „schummeln“ betrachtet. Ganz im Gegenteil sei die wichtigste Komponente eine explizite *Repräsentation von Wissen* über den zu behandelnden Gegenstandsbereich. Die Programmiersprache PLANNER hatte genau dies zum Ziel, nämlich ein Beweissystem so zu strukturieren, daß lokal verteiltes Wissen an verschiedenen Stellen des Beweisprogramms explizit repräsentiert werden konnte. Über die Frage der Repräsentation kam es dann wieder zu zwei unterschiedlichen Standpunkten. Einerseits wurde eine prozedurale Repräsentation favorisiert, wie sie von PLANNER vorgeschlagen wurde, auf der anderen Seite eine rein deklarativ logikorientierte. Synthese dieses fast eine Dekade andauernden Grundlagenstreites sind die logischen Programmiersprachen mit einer prozeduralen Sicht der Logik und einer logischen Sicht der Programmierung.

Dieser Wissenschaftsstreit war außerordentlich fruchtbar und die tiefgreifenden Kontroversen über die methodischen Grundlagen haben unser Gebiet vollständig verändert. Insbesondere haben sie dem zweiten Lager, dem der Simulation menschlicher Vorgehensweise, einen erneuten Aufwind gegeben. Beweissysteme, wie das in den 60er Jahren von L. M. Norton, entwickelte ADEPT, einem heuristischen Theorembeweiser für die Gruppentheorie, bekamen wieder einen neuen Stellenwert, und die Spezialbeweiser, von A. Nevins an der George Washington University auf der Basis einer „human oriented logic" entwickelt, waren in ihrem jeweiligen Spezialgebiet von den damaligen Resolutionsbeweisern nicht mehr zu schlagen. Besonders einflußreich waren die Arbeiten der Gruppe um Woody Bledsoe an der University of Texas, die von Anfang an die vermeintlichen Vorzüge einer einzelnen uniformen Inferenzregel für die gesamte Mathematik in Frage stellte und statt dessen gebietsspezifische Methoden für Standardbereiche der Mathematik entwickelte. Besonders bekannt wurden die Methoden, die für Grenzwertsätze, für die Mengentheorie und für die Nichtstandard-Analysis entwickelt wurden. Ebenso wurden Spezialtechniken für das Rechnen mit Gleichungen und Ungleichungen, sowie für das Rechnen mit Zahlen entwickelt.

So einflußreich diese Arbeiten für das intellektuelle Klima in der Forschung auch waren, so muß doch anerkannt werden, daß bis heute die stärksten Deduktionssysteme auf der Basis uniformer Beweiskalküle entwickelt wurden, unterstützt durch eine sich immer rascher entwickelnde und verbessernde Hardware. Selbst die relativ schwierigen Sätze, die anfangs ausschließlich den Fähigkeiten von Spezialbeweisern vorbehalten waren, wurden nach und nach auch von uniformen resolutionsbasierten Systemen bewiesen, insbesondere dem der Gruppe am Argonne National Lab. Sie hat bis heute neben der von R. Boyer und J Moore in Austin und neben der Markgraf-Karl-Gruppe in Karlsruhe und Kaiserslautern eine gewisse Dominanz des Gebietes (was die leistungsfähigsten Beweissysteme anbelangt) halten könne. Kurzbeschreibungen fast aller derzeitig existierender Beweissysteme findet man in [Sie86].

Während sich diese Beschreibung der wichtigsten Trends völlig auf die Mechanisierung der Prädikatenlogik erster Stufe konzentriert, gab es Entwicklungen, die sich nicht unter dieser Rubrik einordnen lassen und von denen wegen ihrer Bedeutung wenigstens zwei erwähnt werden sollen. Mathematische Lehrbücher sind nicht im Prädikatenkalkül erster Stufe geschrieben, sondern erfordern schon im sehr elementaren Bereich Darstellungskonstrukte, die über die erste Stufe hinausgehen. Höhere Stufe ist ein Gebot nicht nur der Bequemlichkeit der Formulierung in der Mathematik, und seit den Anfängen gab es Versuche, entsprechende Systeme zu entwickeln. Das erste Beweissystem höherer Stufe entstand in einer Gruppe, die Anfang der 60er Jahre unter der Leitung von J. R. Guard verschiedene Beweiser entwickelte und die erste Firma zur Vermarktung der Logik (mit dem zu erwartenden Geschäftsausgang) gegründet hatte. Diese Arbeiten wurden von G. Huet und von T. Pietrzykowski aufgegriffen und weiterentwickelt. Die wichtigste Gruppe mit dem Arbeitsziel der Mechanisierung von Logiken höherer Stufe ist an der Carnegie-Mellon University seit vielen Jahren unter der Leitung von Peter Andrews tätig. Das dort entwickelte System ist beispielsweise in der Lage, Cantors Satz zu beweisen und hat im Laufe der letzten Jahre ebenfalls einen stetigen Zuwachs an Leistungsfähigkeit gewonnen.

AUTOMATED THEOREM PROVING

Some major contributions to ATP classified by philosophy of design

Logic		Human Simulation
Presburger prover Davis 1954		
		Logic Theory Machine, Newell, Shaw, Simon 1957
Beth tableau prover Gilmore 1959		
		Geometry Theorem Proving Machine, Gelernter *et al.* 1959
Gentzen-Herbrand prover Hao Wang 1959		
D-P procedure Davis, Putnam 1980		
Matching Prawitz 1960		
Resolution Robinson 1964		Semi-Automated Math. Guard, Bennett, Easton 1963-67
Resolution prover Wos *et al.* 1965		
Resolution restrictions, strategies *et al*. 1965-70		
Knuth-Bendix theorem Knuth Bendix 1970		ATP project Bledsoe *et al.* 1970
AURA Overbeek, Wos, Winkler, Lust, Smith 1972-		Human-oriented prover Nevins 1971
	Computational Logic Theorem Prover, Boyer, Moore 1973-	
TPS-higher order logic Andrews 1976		
Decision procedures for quantifier-free theories, Shostak, Oppen *et al.* 1977		
	Markgraf Karl Procedure, Siekmann *et al.* 1980	
LMA – an ATP tool kit. ITP – a people's machine built from LMA tools. Lusk, McCune, Overbeek 1980-		

Eine zweite Richtung, die sich unabhängig entwickelte, betrifft die Automatisierung von Induktionsbeweisen und ist wesentlich durch R. Boyer und J Moore von der University of Edinburgh (später der University of Texas at Austin) geprägt worden. Die vollständige Induktion ist ein wichtiges Beweishilfsmittel für rekursiv definierte Objekte und erfordert spezielle Techniken, die in diesem Band in einem eigenen Kapitel beschrieben werden.

Es soll zum Abschluß nicht unerwähnt bleiben, daß sich dieser Bericht fast ausschließlich auf die westlichen Länder konzentriert und die Arbeiten in den sozialistischen Ländern weitgehend außer Acht läßt. In der Sowjetunion hat es bereits seit Anfang der 60er Jahre eine Gruppe von Logikern und Mathematikern gegeben, die sich für automatische Beweisverfahren interessierten und so, wie der Westen weitgehend durch das Resolutionsverfahren dominiert war, durch Maslovs „Inverse Methode“ dominiert wurde. Eine gute Übersicht geben [MMO83], [Lif86].

Außerdem gibt es in der Volksrepublik China unter der Leitung des Mathematikers Wu eine Gruppe, die effiziente Entscheidungsverfahren für die Geometrie entwickelt und mit erstaunlichen praktischen Erfolgen im Westen implementiert hat. Sie ist in der Automatisierung geometrischer Beweise heute dominant.

Die obige Abbildung aus D. Lovelands Beitrag „Automated Theorem Proving: A Quarter Century Review“ in [BL83] faßt die wichtigsten Stationen in der Entwicklung unseres Gebiets noch einmal zusammen:

5 Teilgebiete

Das Gebiet der Deduktionssysteme präsentiert sich in der Forschung durch eine eigene Konferenz, die International Conference on Automated Deduction (CADE), die alle zwei Jahre abgehalten wird. Praktisch alle größeren KI-Konferenzen enthalten besondere Sektionen, die meistens Deduction Systems, Automated Theorem Proving und neuerdings auch Automated Reasoning heißen. Ferner gibt es zwei internationale Zeitschriften, die das Gebiet der Deduktionssysteme abdecken: das Journal of Symbolic Computation und das Journal of Automated Reasoning. Darüberhinaus enthalten aber auch die meisten Zeitschriften der theoretischen Informatik ebenso wie praktisch alle KI-Zeitschriften eigene Rubriken für logikbasierte Methoden oder Deduktionssysteme.

Aus der Sicht dieser Forschung gliedert sich das Gebiet der Deduktionssysteme in Teilgebiete, die grob nach ihrer Größe (Anzahl der Veröffentlichungen pro Jahr) geordnet wie folgt aussehen:

Logisches Programmieren

Die Einsicht, daß sich der Prädikatenkalkül zu einem Beweisprogramm verhält wie beispielsweise LISP zu EVAL, hat zu einer Explosion an Forschungsinteresse und investierten Mitteln und einer fast unüberschaubar gewordenen Anzahl von Veröffentlichungen geführt, in der die Logik selbst als Programmiersprache aufgefaßt wird. Die klassischen Papiere wurden zwar zunächst auf den CADE-Konferenzen präsentiert und es gibt dort nach wie vor wichtige Beiträge zu diesem Gebiet. Die Mehrzahl der Veröffentlichungen werden inzwischen jedoch auf zwei großen internationalen Tagungen zum logischen Programmieren und einer Vielzahl kleinerer lokaler oder nationaler Konferenzen vorgetragen. Hinzu kommen wieder spezielle Sektionen zum logischen Programmieren auf den KI-Konferenzen und auf den Konferenzen über Rechner der Fünften Generation.

Termersetzungssysteme

Dieses Teilgebiet wurde mit den meisten der inzwischen klassischen Veröffentlichungen zunächst auf den CADE-Konferenzen etabliert. Es hat sich inzwischen jedoch auch verselbständigt, mit einer eigenen internationalen Konferenz (International Conference on Rewriting Techniques and Applications), die alle zwei Jahre abgehalten wird. Das Rechnen mit Gleichungen, insbesondere mit Gleichungen, die zu einem kanonischen Termersetzungssystem gerichtet werden können, ist ein wichtiger Bestandteil der theoretischen Informatik geworden. Die Termersetzungssysteme sind neben der Unifikationstheorie das derzeit aktivste Teilgebiet im Bereich der Deduktionssysteme und stellen rein zahlenmäßig den größten Prozentsatz an Forschungspapieren auf der CADE.

Unifikationstheorie

Der Versuch, gewisse Axiome wie die Assoziatiativität, die Kommutativität oder die Idempotenz direkt in den Inferenzmechanismus eines Beweissystems einzubauen, hat dieses Gebiet zunächst motiviert. Inzwischen hat es sich jedoch zu einem wichtigen Teilgebiet der theoretischen Informatik und der Theorie der Künstlichen Intelligenz entwickelt, dessen zentrale Fragestellung, ob zwei vorgegebene Strukturen gleichgemacht werden können, eine grundlegende Operation aller KI-Systeme betrifft. Die Unifikationstheorie liefert spezielle Algorithmen und das theoretische Rüstzeug für die Untersuchung solcher Probleme. Das Journal of Symbolic Computation bringt 1987 eine Sondernummer zu diesem Thema heraus (Gasteditor: C. Kirchner), es gibt ebenfalls einen internationalen Workshop zu diesem Gebiet, der jährlich abgehalten wird.

Nichtmonotone Logiken

Eine Logik heißt *monoton*, wenn durch die Hinzunahme neuer Annahmen nicht weniger folgt als ohne diese Annahmen: $\Psi \models \mathrm{TH}$ impliziert $\Psi \cup \Xi \models \mathrm{TH}$. Die klassischerweise untersuchten Logiken sind monoton, die meisten Systeme der Künstlichen Intelligenz jedoch nicht. Es stellte sich daher die Frage, ob klassische Logiken überhaupt eine Grundlagendisziplin für die KI sein können, oder nicht doch, wie es die MIT-Schule propagierte, eher hinderlich für den wissenschaftlichen Fortschritt auf dem Gebiet der KI seien. Wenn man dagegen den logikbasierten Ansatz in der KI für fundamental und unverzichtbar hält, muß man sich mit dem Problem nichtmonotoner Logiken und darauf aufbauender nichtmonotoner Systeme auseinandersetzen. Solche Logiken und Systeme („non-monotonic reasoning systems“) sind seit knapp zehn Jahren wichtiger Bestandteil der theoretischen KI geworden, und es gibt einen internationalen Workshop, der regelmäßig abgehalten wird, sowie spezielle Sektionen auf den meisten KI-Konferenzen.

Beweiserbau

Dies ist die klassische Teildisziplin unseres Gebietes, in der Fragen zum Software Engineering von Deduktionssystemen untersucht werden. Im Vordergrund steht dabei das Interesse an der Struktur eines Deduktionssystems und an der Spezifikation von Teilkomponenten. Ganz analog beispielsweise zum Compilerbau, in dem die Standardstrukturen eines Compilers (wie Symboltabelle, Parser, Codeerzeuger usw.) festgelegt werden, bestehen heutige Deduktionssysteme aus festen Bestandteilen, zum Beispiel der Unifikationskomponente, der

Selektionskomponente und speziellen Datenstrukturen zur Repräsentation der Terme und Literale. Leider ist dieses Gebiet in den letzten Jahren wieder etwas in den Hintergrund getreten.

Gleichheitsbehandlung

In dieser Teildisziplin beschäftigt man sich mit Verfahren zur Behandlung der Gleichheit, da diese Relation einerseits sehr schwierig zu mechanisieren ist, andererseits aber so häufig in mathematischen Disziplinen und in der theoretischen Informatik vorkommt, daß sie zu einem eigenen Untersuchungsgegenstand geworden ist.

Logische Methoden wissensbasierter Systeme

Dieser Teilbereich ist wieder von besonderem Interesse in der gegenwärtigen Forschung und erlebt derzeit eine Renaissance mit der entsprechenden Explosion an Veröffentlichungen. Die wichtigsten Probleme sind die Entwicklung von Inferenzmethoden für wissensbasierte Systeme und die Mechanisierung nichtklassischer Logiken (Temporal-, Modal- und dynamische Logik, um nur einige zu nennen), sofern sie für die Künstliche Intelligenz von besonderer Bedeutung sind. Das Journal of Automated Reasoning gibt zu diesem Thema 1987 eine Sondernummer heraus (Gasteditor: M. McRobbie).

Deduktive Programmsynthese

Dies ist eine der älteren Teildisziplinen unseres Gebietes, in dem versucht wird, mit Hilfe deduktiver Verfahren ein Programm automatisch zu synthetisieren. Da ein Beweis auch als Ablauf eines entsprechenden Programmes gesehen werden kann, läßt sich dieses Programm aus dem Beweis synthetisieren. Dieses Gebiet hat zu wichtigen theoretischen Einsichten geführt und auch gewisse praktische Erfolge vorweisen können, der Übergang in die praktische Anwendung ist jedoch bisher nicht erfolgt.

6 Anwendungen

Das Gebiet der Deduktionssysteme hat sich in den letzten dreißig Jahren von einer relativ esoterischen Spezialdisziplin zu einer wichtigen Grundlagendisziplin der KI gewandelt. In den letzten zehn Jahren haben sich darüber hinaus eine Reihe von Anwendungsbereichen für solche Systeme herausgebildet, die sich grob nach ihrer Größe (Umsatz und investierte FE-Mittel) geordnet wie folgt darstellen:

Das Aufkommen *logischer Programmiersprachen* hat nicht nur die Softwareentwicklung stark beeinflußt, sondern auch die Architektur zukünftiger Rechner (der 5. Generation) geprägt: Die Leistungsfähigkeit dieser Soft- und Hardware wird konsequenterweise nicht mehr in MIPS (Million Instructions Per Second), sondern in LIPS (Logical Inferences Per Second) angegeben. Derzeit ist ein starker Wettbewerb der großen Industrienationen zu beobachten, diese Technologie zu beherschen und zu kommerzialisieren.

Ein zweiter relativ großer Anwendungsbereich sind die *deduktiven Datenbanken*. Beeinflußt von den deduktiven Question-Answering-Systemen der späten 60er Jahre und den logischen

Programmiersprachen ist der Grundgedanke einer deduktiven Datenbank, nicht alle Daten explizit wie in einer klassischen Datenband abzuspeichern, um sie dann bei Bedarf abrufen zu können, sondern Wissen auch über Inferenzregeln abzulegen, mit deren Hilfe man Fakten deduzieren kann. Dieses Gebiet hat sich nicht zuletzt wegen des starken kommerziellen Interesses rasch verselbständigt und wächst derzeit mit anderen KI-Gebieten, wie den Expertensystemen, zu einem großen Gebiet der „Wissensbanken“ (Knowledge Bases [Fro86]) zusammen.

Ein dritter Bereich, in dem Deduktionssysteme von Bedeutung sind, beschäftigt sich mit dem Nachweis der *Korrektheit* von Hard- oder Software. Wenn die vor zehn Jahren gehegten Hoffnungen auf eine vollständige *Programmverifikation* auch nicht eingetroffen sind, wurden doch bedeutende Erfolge in der Verifikation von Mikroprogrammen und der Verifikation von Hardwarekomponenten erzielt. Das Hauptproblem in der Programmverifikation liegt derzeit nicht so sehr in der mangelnden Leistungsfähigkeit der Beweissysteme, sondern in den ungenügenden Spezifikationstechniken, die eine breite Anwendung bisher verhinderten. Bei besonders kritischer, spezieller Software werden diese Techniken jedoch ebenso eingesetzt wie für den Entwurf und den Korrektheitsnachweis von Schaltungen, von Ablaufplänen oder von sicherheitsrelevanten Anlageplänen.

Ein weiterer Bereich beschäftigt sich mit der schon geschilderten Entwicklung von *Inferenz-komponenten* für wissensbasierte Systeme, die von allgemeinen Verfahren bis zu speziellen Inferenzkomponenten für spezifische Wissensrepräsentationssprachen wie KL-ONE reichen.

Im dem Anwendungsbereich der *deduktiven Planverfahren* werden Komponenten für Expertensysteme zur Planung, für die Sprachverarbeitung oder für die Robotik entwickelt.

7 Schlußwort

Wenn Leibniz’ beflügelnde Vision einer allgemeinen „lingua characteristica“ auch nicht wahr geworden ist und kein Politiker seine Argumente an eine mit einem mechanischen „calculus ratiocinator“ ausgestattete Behörde zum Nachrechnen und Überprüfen schicken muß – logikorientierte wissensbasierte Systeme sind ein wesentlicher Teil heutiger Technik und werden zunehmend zu einem festen Bestandteil unseres Lebens werden.

Eine erschöpfende Geschichte der Deduktionssysteme wurde bisher leider nicht geschrieben. Dieser Artikel borgt aus vielen Quellen, unter anderem aus [WH83, BL83] und ganz besonders aus M. Davis’ einflußreichem Artikel „The Prehistory and Early History of Automated Deduction“ [Dav83].

Literatur

Die geschilderte historische Entwicklung leistungsfähiger Deduktionssysteme wurde nicht durch Einzelverweise auf die jeweilige Originalliteratur belegt; die angesprochenen Originalartikel findet man in den beiden Sammelbänden [SW83a, SW83b].

Bib83 W. Bibel: *Automated Theorem Proving*, Vieweg, Braunschweig (1983/87)

BK83 K. Berka, L. Kreiser: *Logik-Texte – Kommentierte Auswahl zur Geschichte der modernen Logik*, Akademie-Verlag, Berlin, DDR (1983)

BL83 W. Bledsoe, D. Loveland (eds.): *Contemporary Mathematics, Automated Theorem Proving after 25 Years*, American Mathematical Society, vol. 29, Rhode Island (1983)

BM79 R. Boyer, J S. Moore: *A Computational Logic*, Academic Press, London (1979)

Bun83 A. Bundy: *The Computer Modelling of Mathematical Reasoning*, Academic Press (1983)

CDHI64 T.J. Chinlund, M. Davis, G. Hineman, D. McIlroy: *Theorem Proving by Matching*, Bell Laboratories (1964)

CL73 C.-L. Chang, R.C.-T. Lee: *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York (1973)

Dav63 M. Davis: *Eliminating the Irrelevant from Mechanical Proofs*, Proc. Symp. Appl. Math. XV (1963) 15-30

Dav83 M. Davis: *The Prehistory and Early History of Automated Deduction*, in [SW83a]

Des37 Anhang zu R. Descartes: *Discours de la Methode* (1637); zitiert nach Übersetzung R. Descartes: *Geometry*, von D.E. Smith und M.L. Latham, Dover (1954)

FGCS84 Fifth Generation Computer Systems, Edition ICOT, North Holland (1984)

Fre82 G. Frege: *Über den Zweck der Begriffsschrift* (1882), in: Gottlob Frege, Begriffsschrift, G. Olms Verlagsbuchhandlung, Hildesheim (1964)

Fro86 R.A. Frost: *Introduction to Knowledge Base Systems*, Collins Professional and Technical Books, London (1986)

Ger90 C.I. Gerhardt (Hrsg.): *Die philosophischen Schriften von Gottfried Wilhelm Leibniz*, Band 7, Berlin (1980)

Hei67 J. van Heijenoort: *From Frege to Gödel – A Source Book in Mathematical Logic*, Harvard University Press, Cambridge, Massachusetts (1967)

HS85 C.A.R. Hoare, J.C. Shepherdson: *Mathematical Logic and Programming Languages,* Prentice Hall (1985)

Kow75 R. Kowalski: *A Proof Procedure Using Connection Graphs*, J.ACM 22,4 (1975)

Kow79 R. Kowalski: *Logic for Problem Solving*, North Holland, Amsterdam (1979)

Lif86 V. Lifschitz: *Mechanical Theorem Proving in the USSR: The Leningrad School, Delphic Associates*, Falls Church, Virginia (1986)

Lov78 D. Loveland: *Automated Theorem Proving: A Logical Basis*, Fundamental Studies in Computer Science, Vol. 6, North-Holland, New York (1978)

McC79 P. McCorduck: *Machines who Think*, W.H. Freeman, San Francisco (1979)

Min61 M. Minsky: *Steps toward Artificial Intelligence*, Nachdruck in: *Computer and Thoughts* (ed. E. Feigenbaum und J. Feldman), McGraw Hill (1963)

MMO83 Y. Maslov, G.E. Mints, V.P. Orevkov: *Mechanical Proof Search and the Theory of Logical Deduction in the USSR*, in [SW83a]

MW85 Z. Manna, R. Waldinger: The Logical Basis for Computer Programming, Addison-Wesley, Reading, Massachusetts (1985)

Nil80 N. Nilsson: *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA (1980)

Pea89 G. Peano: *Arithmetices principia, nova methodo exposita*, Turin (1889)

Poi52 H. Poincaré: *Science and Method*, übersetzt von F. Maitland, Dover (1952)

Rob65 J.A. Robinson: *A Machine-Oriented Logic Based on the Resolution Principle*, J.ACM 12 (1965)

Rus46 B. Russell: *History of Western Philosophy,* Unwin University Books (1946/61)

Sch82 E. Schröder: *Recension der Begriffsschrift,* Z. f. Mathematik u. Physik, Bd. XXV (1882)

Sie86 J. Siekmann (ed.): Proc. Eighth International Conference on Automated Deduction, Springer LNCS, vol. 230 (1986)

SW83a J. Siekmann, G. Wrightson (eds.): *Automation of Reasoning; Classical Papers on Computational Logic 1957-1966*, Springer Verlag, Berlin (1983)

SW83b J. Siekmann, G. Wrightson (eds.): *Automation of Reasoning; Classical Papers on Computational Logic 1967-1970*, Springer Verlag, Berlin (1983)

WH83 L. Wos, L. Henschen: *Automated Theorem Proving 1965-1970*, in [SW83b]

WO84 L. Wos, R. Overbeek, E. Lusk, J. Boyle: *Automated Reasoning – Introduction and Applications*, Prentice-Hall, Englewood Cliffs, NJ (1984)