Die Beweisentwicklungsumgebung Ω–MKRP*

Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis
Dan Nesmith, Jörn Richts, Jörg Siekmann
Fachbereich Informatik, Universität des Saarlandes
D-66041 Saarbrücken

{huang|kerber|kohlhase|melis|nesmith|richts}@cs.uni-sb.de

siekmann@dfki.uni-sb.de

Zusammenfassung

Die Beweisentwicklungsumgebung Ω–MKRP soll Mathematiker bei einer ihrer Haupttätigkeiten, nämlich dem Beweisen mathematischer Theoreme unterstützen. Diese Unterstützung muß so komfortabel sein, daß die rechnergestützte Suche nach formalen Beweisen leichter und insbesondere weniger aufwendig ist, als ohne das System. Dazu muß die verwendete Objektsprache ausdrucksstark sein, man muß die Möglichkeit haben, abstrakt über Beweispläne zu reden, die gefundenen Beweise müssen in einer am Menschen orientierte Form präsentiert werden und vor allem muß eine effiziente Unterstützung beim Füllen von Beweislücken zur Verfügung stehen. Das im folgenden vorgestellte Ω–MKRP-System ist der Versuch einer Synthese der Ansätze des vollautomatischen, des interaktiven und des planbasierten Beweisens. Dieser Artikel soll eine Übersicht über unsere Arbeit an diesem System geben.

Abstract

The proof development environment Ω–MKRP is intended to lend automated assistance to mathematicians in one of their main activities—the proving of mathematical theorems—in such a way that the system is a help, not a hindrance. Among the requisites for such a system are: an expressive object language; a way to speak abstractly about proof plans; a human-oriented presentation of constructed proofs; extensive assistance in filling proof gaps. The Ω–MKRP system presented in the following is an attempt to meet these requirements by fusing the paradigms of fully automated, interactive, and plan-based theorem proving into a single framework. This article gives a survey of our work with this system.

Schlüsselwörter: automatisches Beweisen, mathematische Assistenzsysteme, Beweisentwicklungsumgebung, Beweisplanung, Analogie, Beweispräsentation
CR Subject Classifications: I.2.3

*Das Ω–MKRP-System ist im Teilprojekt D2 des Sonderforschungsbereich 314 „Künstliche Intelligenz – Wissensbasierte Systeme“ entstanden. Informationen über das Projekt kann man auch über das World Wide Web erhalten, URL: http://jswww.cs.uni-sb.de/pub/www/

1 Einführung

Die klassische Aufgabe eines rechnergestützten Beweissystems ist der Nachweis, ob eine bestimmte Formel, das Theorem, aus einer gegebenen Formelmenge, der Axiomenmenge (der Wissensbasis), logisch folgt. Diese Frage ist im allgemeinen unentscheidbar, jedoch wenn das Theorem gilt, dann kann man es in endlich vielen Schritten beweisen. In den letzten dreißig Jahren wurden mächtige Kalküle wie Resolution und Paramodulation entwickelt und verfeinert, mit denen in vielen für die Praxis relevanten Spezialfällen positive Ergebnisse erzielt werden können.
Das Hauptinteresse im Gebiet des automatischen Beweisens war lange Zeit – und ist heute auch weitgehend noch – von der Motivation geprägt, effiziente *vollautomatische* Systeme zu entwickeln. In diesem Paradigma sind verschiedene automatische Beweiser entstanden, wie das äußerst effiziente OTTER-System vom Argonne National Lab, USA, und unser eigenes MKRP-System, ein leistungsfähiger Resolutionsbeweiser für eine sortierte Logik erster Stufe mit eingebauter Gleichheit. Über einen Zeitraum von fast fünfzehn Jahren haben wir Stärken und Schwächen des MKRP-Systems unter anderem dadurch getestet, daß wir Theoreme eines mathematischen Lehrbuches [4] mit dem System bewiesen haben. Obwohl viele Theoreme formal in MKRP bewiesen werden konnten, ist die abschließende Bewertung doch eher negativ, denn es zeigten sich prinzipielle Schwächen, die allen automatischen Beweisern anhaften und die eine Benutzung eines solchen klassischen Systems als Werkzeug im Sinne eines rechnergestützten mathematischen Assistenten utopisch erscheinen lassen. Diese Schwächen haben dazu geführt, daß unser derzeitiges Forschungsinteresse im wesentlichen nicht mehr im Rahmen des klassischen Beweiserbaus liegt, sondern daß sich die Arbeiten auf ein neues Paradigma konzentrieren und dadurch auch ein neues System, nämlich das Ω–MKRP-System, entsteht. Die negativen Erfahrungen mit klassischen, vollautomatischen Beweisern lassen sich im wesentlichen so zusammenfassen:

1. Die Darstellung mathematischer Probleme in einer Logik erster Stufe ist oft sehr umständlich, selbst wenn es sich dabei um eine Logik mit Gleichheit und Sorten handelt. Dies hat bei uns zur Entwicklung der Objektsprache $\mathcal{POST}$, einer Variante der Logik höherer Stufe, geführt.

2. Ein Beweiser wie MKRP zeichnet sich unter anderem dadurch aus, daß er universell einsetzbar ist, weil er im Gegensatz zu Mathematikern kein spezielles mathematisches Wissen hat. Angemessener ist die Vorgabe dieses Wissens in einer zentralen Datenbank des mathematischen Faktenwissens.

3. Oft sind Sätze zu schwer, als daß sie vollautomatisch bewiesen werden können. Will man ein Beweissystem als Werkzeug benutzen, so muß der Benutzer die Möglichkeit haben, in den Beweisprozeß eingreifen und diesen gegebenenfalls interaktiv steuern zu können.

4. Für Beweiser, die wie das MKRP-System auf einer Normalform arbeiten, stellt sich die Frage, wie gefundene Beweise so präsentiert werden können, daß sie

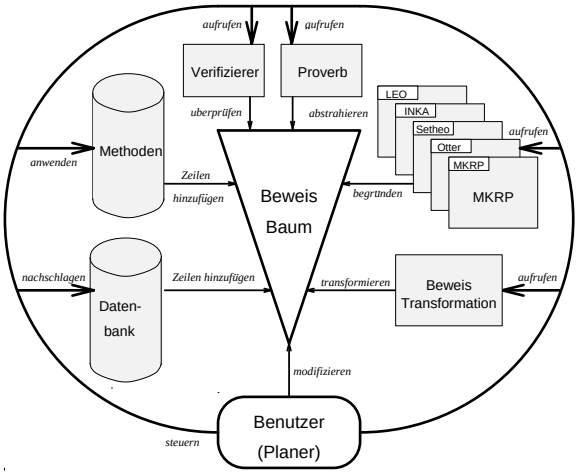

Abbildung 1: Architektur von Ω–MKRP

dem Benutzer verständlich sind. Der Benutzer will keine Klauselbeweise, die oft einige hundert Schritte lang sind, lesen müssen, sondern möglichst gut gewählte und strukturierte Darstellungen, die die wesentlichen Schritte betonen.

Um Mathematikern eine größtmögliche automatische Unterstützung bei der Beweisentwicklung bieten zu können, haben wir das im folgenden beschriebene System Ω–MKRP entworfen und bisher teilweise implementiert. Es ist eine interaktive Beweisentwicklungsumgebung, die Vorteile des maschinenorientierten Beweisens, wie der Resolution, und des eher auf menschlichem Vorgehen aufbauenden planungsbasierten Beweisens, wie es von Bundy [2] und anderen vorgeschlagen wurde, zu verbinden versucht. Dazu sollen in einer Umgebung, in der ein Benutzer seine Beweise interaktiv entwickeln kann, klassische automatische Beweiser erster Stufe wie das MKRP-System integriert werden, aber auch Beweiser für Logiken höherer Stufe wie TPS oder unser eigenes System LEO. Dadurch werden dem Benutzer die Möglichkeiten an die Hand gegeben, die eigenen Ideen auszuführen, ohne daß er wie in einer der heutigen Beweisentwicklungsumgebungen gezwungen ist, den Beweis durch die Angabe von Kalkülregeln oder zusammengesetzten Kalkülregeln (sogenannten Taktiken) detailliert selbst führen zu müssen.
Das Problem wird am Anfang als trivialer Beweisbaum eingeführt, und dieser wird als zentrale Datenstruktur solange modifiziert, bis ein endgültiger Beweis erzeugt ist. Der Benutzer hat dabei mehrere Möglichkeiten einzugreifen, er kann insbesondere

- mathematisches Faktenwissen aus der Ω-Datenbank in den Beweisbaum einbauen (im folgenden gehen wir darauf nicht ein),
- verschiedene automatische Beweiser aufrufen, unter anderem MKRP oder LEO,
- auf vorhandene Methoden aus der Methodenbank zurückgreifen, die dann zum Beispiel einzelne Beweisschritte ausführen (vergleiche Abschnitt 2),

- neue Methoden erzeugen lassen durch die Kombination bestehender Methoden oder durch die Anwendung von Metamethoden,
- einen Analogieplaner verwenden, der Beweispläne durch Analogie zu bereits vorhandenen erstellt (vergleiche Abschnitt 2),
- den erzeugten Beweis von einem Verifizierer überprüfen lassen und
- automatisch erzeugte Beweise in das Format des natürlichen Schließens transformieren oder den Gesamtbeweis in natürliche Sprache übersetzen lassen (vergleiche Abschnitt 3).

Man kann die Aufgabe des Benutzers als einen abstrakten *Planung*sprozeß auffassen, in dem ein Beweis interaktiv mit dem System entwickelt wird. Der Benutzer plant sein Vorgehen auf einer wesentlich abstrakteren Ebene als auf der eines konkreten Beweises, und er kann dabei allgemeine Lemmata oder Beweismethoden verschiedener Granularität benutzen. Diese menschliche Planungsaufgabe soll zunehmend von einer automatischen Planungskomponente unterstützt werden, so daß der Automatisierungsgrad von Ω–MKRP auf die Dauer steigen wird. Der Planer kann den Beweisbaum so modifizieren, daß als Begründungen für Beweiszeilen, die Methoden oder Beweise der angeschlossenen vollautomatischen Beweiser eingetragen werden. Um das Gesamtproblem zu beweisen, müssen diese Schritte abschließend vom Verifizierer überprüft werden, da wir – wie auch Bundy – nicht fordern, daß eine Methode immer erfolgreich, ja nicht einmal korrekt sein muß. Einen Überblick über die Architektur von Ω–MKRP gibt Abbildung 1. Aus Platzgründen gehen wir im folgenden nur auf einige Aspekte von Ω–MKRP ein. Insbesondere wird die Logik höherer Stufe und ihre Mechanisierung nicht behandelt, ebensowenig können wir auf die Frage eingehen, wie man Partialität behandeln soll. Der Leser sei auf [8, 11, 10] verwiesen. Eine ausführlichere Version dieses Beitrags findet man in [7].

2 Beweisplanung

Eine wichtige Komponente zur Unterstützung des Benutzers ist ein rechnergestütztes Verfahren zur *Beweisplanung*. Der wesentliche Unterschied zum traditionellen Vorgehen der *Beweissuche* ist, daß nicht alle möglichen Inferenzschritte aufgelistet werden, um in diesem (riesigen) Suchbaum dann eine Sequenz von Schritten zu finden, die den Satz beweist. Stattdessen wird der Beweis auf einer viel allgemeineren Ebene unter der Verwendung von bereichsspezifischen Methoden *geplant*. Natürlich gibt es auch hier wieder einen Suchraum, aber durch die abstraktere Sichtweise, die dem menschlichen Vorgehen sehr viel näher kommt, werden diese Suchräume auch für schwierige Probleme handhabbar. Wir verfolgen dabei eine Erweiterung von Bundys Ansatz [2], den wir im folgenden beschreiben wollen.
Der Beweisplanungsansatz von Bundy baut auf den taktikbasierten Deduktionssystemen wie NUPRL auf. Während die Taktiken in NUPRL Beweisprozeduren sind,

werden in Bundys Ansatz die Taktiken zu sogenannten *Methoden* erweitert, indem sie um Spezifikationen ergänzt werden. Methoden stellen damit Planungsoperatoren im klassischen Sinne der Planverfahren in der KI dar, mit denen eine rechnergestützte Planung des Beweises möglich ist. Wenn ein vollständiger Plan gefunden ist, wird dieser ausgeführt, das heißt die entsprechenden Taktiken werden auf den partiellen Beweis angewendet und auf diese Weise wird dann ein fertiger Beweis auf Kalkülebene generiert. Da die Spezifikation der Methoden in der Regel eine Abstraktion der Eigenschaften der Taktik ist, muß der gefundene Beweis abschließend auf seine Korrektheit überprüft werden. Falls der Beweis nicht vollständig ist, wird der Planungsvorgang mit den verbliebenen offenen Teilproblemen rekursiv erneut gestartet. Der gesamte Prozeß von der Analyse des Problems bis zum vollständigen Beweis ist also ein verschränkter Prozeß von Beweisplanung, Anwendung von Taktiken und anschließender Verifikation (und eventueller menschlicher Interaktionen).
Unsere Erweiterung des Ansatzes von Bundy besteht darin, zusätzlich Metamethoden einzuführen, die die bestehenden Methoden verändern können. Dies wird dadurch motiviert, daß Mathematiker die von ihnen erlernten Beweisverfahren durchaus abändern, um sie an neue Situationen anzupassen. Um dies auch mit den im Rechner benutzten Methoden tun zu können, haben wir vorgeschlagen, Methoden und Taktiken teilweise deklarativ zu beschreiben [6], da die Änderung einer deklarativen Sprache durch die Metamethoden natürlich einfacher ist als die einer prozeduralen.

2.1 Methoden

Methoden sollen sowohl Beweisprozeduren (die Taktiken) repräsentieren als auch entsprechende Planungsoperatoren (durch deren Spezifikation). Da außerdem Methoden weitgehend deklarativ beschrieben werden sollen, benötigen wir Metavariablen, die für Symbole, Formeln, Beweiszeilen usw. stehen. Zu diesem Zweck bestehen Methoden in unserem Ansatz aus sechs Komponenten, von denen die ersten fünf deklarativ sind, während die letzte prozedural ist:

- Die *Deklaration* ist eine Signatur, die die verwendeten Metavariablen definiert.
- Die *Voraussetzungen* bestehen aus einer Liste von Beweiszeilen, die von der Methode benutzt werden, um die sogenannten Folgerungen zu beweisen.
- Die *Randbedingungen* sind zusätzliche Bedingungen, die gelten müssen, bevor die Methode angewendet werden kann.
- Die *Folgerungen* bestehen aus einer Liste von Beweiszeilen, die von der Methode bewiesen werden.
- Der *Deklarative Inhalt* wird von dem prozeduralen Inhalt interpretiert.
- Der *Prozedurale Inhalt* ist entweder eine Standardprozedur, die den deklarativen Inhalt interpretiert, oder eine spezielle Inferenzprozedur, die für einen ganz bestimmten Problemtyp geschrieben wurde.

Die *Voraussetzungen*, *Randbedingungen* und *Folgerungen* geben an, ob eine Methode in einer bestimmten Beweissituation anwendbar ist. Während diese Spezifikation also alle Informationen enthält, die für den Planer notwendig sind, repräsentieren der *deklarative Inhalt* und der *prozedurale Inhalt* zusammen eine Taktik, die in Systemen wie NUPRL oder in Bundys Ansatz als Prozedur realisiert würden. Die Taktik generiert später den eigentlichen Beweis. Die Aufteilung von deklarativem und prozeduralem Wissen einer Taktik bewegt sich innerhalb einer breiten Spannweite. Die einzige Bedingung, die wir an Methoden stellen, ist, daß der prozedurale Inhalt Beweiszeilen konstruiert, die in den aktuellen Beweiszustand eingefügt werden können. Im einen Extrem ist der prozedurale Inhalt ein einfacher Interpreter, der das Beweisschema im deklarativen Inhalt in den aktuellen Beweiszustand einfügt (indem er die Metavariablen entsprechend bindet); im anderen Extrem (das genau Bundys Ansatz entspricht) ignoriert der prozedurale Inhalt den leeren deklarativen Inhalt und konstruiert neue Beweiszeilen rein prozedural.
Würden wir nur rein prozedurale Taktiken erlauben, so müßten bei einer Modifikation von Methoden die entsprechenden Programme verändert werden. Wird eine Methode durch eine Metamethode modifiziert, so bleibt das prozedurale Wissen unverändert, nur das deklarative wird geändert. Erst durch die Trennung von prozeduralem und deklarativem Wissen in den Methoden werden Metamethoden praktisch realisierbar.

2.2 Ein Beispiel: Die Homomorphie-Methode

Wir wollen nun die `hom1-1`-Methode beispielhaft vorführen (siehe Abbildung 2). Das Beweisverfahren, das durch diese Methode repräsentiert wird, kann informell so beschrieben werden: *Wenn* f *eine* in Zeile l_1 *gegebene Funktion und* P *ein* in l_2 *definiertes Prädikat ist und wir haben das Ziel,* in l_6 $P(f(c))$ *zu zeigen, dann zeige* in l_3 $P(c)$ *und benutze dies, um* $P(f(c))$ *zu zeigen.* Die Idee dieses Verfahrens ist, daß f ein Homomorphismus bezüglich der Eigenschaft P ist.

Methode : `hom1-1`		
Dekl.	$l_1, l_2, l_3, l_4, l_5, l_6$: beweiszeile j_1, j_2, j_3: begründung $\Phi, \Psi, \Psi_1, \Psi_2$: formel	x, y: variable P, f, c: konst
Vor.	l_1, l_2, l_3	
Rand-bed.	$\mathsf{typ}(c) \doteq \mathsf{ergebnistyp}(f)$ & $\Psi_1 := \Psi[x \leftarrow c]$ & $\Psi_2 := \Psi[x \leftarrow f(c)]$	
Folg.	l_6	
Dekl. Inhalt	(l_1) (l_1) $\vdash \forall y.\, \Phi_f$ (l_2) $(l_1, l_2) \vdash \forall x.\, P(x) \Leftrightarrow \Psi$ (l_3) $(l_1, l_2) \vdash P(c)$ (l_4) $(l_1, l_2) \vdash \Psi_1$ (l_5) $(l_1, l_2) \vdash \Psi_2$ (l_6) $(l_1, l_2) \vdash P(f(c))$	(j_1) (j_2) (j_3) (`DefE` l_2, l_3) (offen l_1, l_4) (`DefI` l_2, l_5)
Proz. Inhalt	`schema-interpreter`	

Abbildung 2: Die Methode `hom1-1`.

Die Spezifikation von `hom1-1` fordert in l_1 und l_2 lediglich die Existenz einer Definition für f und P und keine weiteren Eigenschaften. Die Ausführung der Taktik hinterläßt daher eine Lücke im Beweis: Nachdem ein vollständiger Plan gefunden wurde, fügt die Taktik von `hom1-1` die Zeilen l_1 bis l_6 in den Beweis ein; für l_1, l_2 und l_3 sind durch

das Planen Begründungen gefunden worden; l_4 und l_6 erhalten eine Begründung, indem die Untertaktiken `def-e` und `def-i` ausgeführt werden, die die Definition von P aus Zeile l_2 auf die Formeln aus l_3 bzw. l_6 anwenden; die Begründung von Zeile l_5 bleibt jedoch offen. Dies führt in der folgenden Verifikationsphase zu einem rekursiven Aufruf der Beweisplanung. Kann l_5 so bewiesen werden, ist der Beweis vollständig geführt, andernfalls ist die Anwendung von `hom1-1` gescheitert.
Die Methode `hom1-1` läßt sich nur für ein einstelliges Prädikat und eine einstellige Funktion benutzen (daher der Name `hom1-1`). Ein entsprechendes, auf Homomorphie beruhendes Beweisverfahren läßt sich auch auf mehrstellige Prädikate oder Funktionen anwenden. In unserem Ansatz kann aus der Methode `hom1-1` eine passende Methode durch eine Metamethode `add-argument` automatisch erzeugt werden [6]. Die Metamethode erhält als Argumente eine Methode und ein einstelliges Funktions- bzw. Prädikatssymbol g; sie ersetzt dann alle Vorkommen von $g(a)$ durch $g'(a, b)$ und fügt zu jeder Zeile, in der solch ein a ohne g vorkommt, eine zusätzliche Kopie ein, in der a durch b ersetzt ist.
Als Ergebnis der Anwendung von `add-argument` auf `hom1-1` und f erhält man eine Methode `hom1-2`, die unter anderem als Folgerung die Zeile $P(f'(c, d))$ und als Voraussetzung die Zeilen $P(c)$ und $P(d)$ enthält. Diese Methode kann man nun auf ein Beweisziel wie $symmetrisch(\sigma \cup \rho)$ anwenden, und man erhält die Teilziele $symmetrisch(\rho)$ und $symmetrisch(\sigma)$ als Ergebnis. Dies ist ein wichtiger Schritt in dem Beweis, daß die Vereinigung zweier symmetrischer Relationen wieder symmetrisch ist.

2.3 Rechnergestützte Analogiebildung

Eine der wichtigsten Heuristiken, die Menschen bei der Suche nach Beweisen verwenden, ist die Analogie. Beweisen durch Analogie bedeutet, einen Beweis für ein Zieltheorem T_2 zu finden, indem man versucht, den Beweis eines ähnlichen Quelltheorems T_1 zu übertragen.
Um Analogie effizient im Ω–MKRP-System behandeln zu können, haben wir zunächst empirische Untersuchungen der Beweise in [4] durchgeführt, die vom Autor explizit als „analoger Beweis“ angegeben waren.
Diese Untersuchungen haben ergeben, daß bisherige Ansätze, Beweisen durch Analogie zu automatisieren (siehe zum Beispiel [14]), nur bedingt geeignet sind, tatsächlich vorkommende Beweise durch Analogie zu finden, da diese Verfahren vor allem auf der Abbildung von Symbolen des Quelltheorem auf Symbole des Zieltheorem und auf dem Transfer von logischen Kalkülschritten beruhen.
Es zeigt sich, daß auch Abstraktionen und andere *Reformulierungen* für die untersuchten Analogien erforderlich sind, die über Symbolabbildungen hinausgehen. Diese Reformulierungen führen Heuristiken aus, die den Zusammenhang zwischen den Änderungen einer zu beweisenden Formel und der Änderung ihres Beweisplanes beschreiben. Beispielsweise wird eine Reformulierung, die die Argumente einer Funktion vervielfacht, häufig benutzt, um Theoreme von n-dimensionalen Räumen

auf $(n+m)$-dimensionale Räume zu übertragen. Eine solche Reformulierung kann nicht durch schlichte Symbolabbildung erreicht werden, weil das Vervielfachen von Argumenten die Beweisstruktur verändert.
Reformulierungen werden durch die schon erwähnten Metamethoden, die Beweispläne verändern, ausgeführt. Einzelne Reformulierungsprozeduren sind zu Testzwecken implementiert worden. Dadurch wurde es möglich, einige (relativ schwierige) mathematische Theoreme erstmalig per Analogie zu beweisen. Ein interessantes Beispiel ist der Beweis eines zweidimensionalen Heine-Borel-Theorems, der durch Analogie zu einem eindimensionalen Heine-Borel-Theorem geführt werden konnte.
Mit dem Transfer von Beweisplänen umgeht man Probleme, die von Versuchen mit der detaillierten Übertragung logischer Kalkülschritte bekannt sind. Besonders deutlich wird das bei Methoden, die den Aufruf eines automatischen Beweisers beinhalten, der einen kleinen Beweis einer Teilbehauptung ausführt. Die analoge Übertragung erfordert nur, daß der Beweiser den Beweis einer reformulierten Teilbehauptung führen kann, jedoch nicht, daß jeder logische Kalkülschritt übertragen werden muß. Dadurch wird die analoge Übertragung robuster [13].

3 Beweispräsentation

Eine wichtige, aber bisher wenig untersuchte Anforderung an Deduktionssysteme ist die Präsentation gefundener Beweise in für Menschen verständlicher Form. Im Gebiet des automatischen Beweisens sind Übersetzungsverfahren entwickelt worden, die Beweise aus verschiedenen maschinenorientierten Formalismen in Beweise des natürlichen Schließens (ND-Beweise) transformieren [12, 15], aber leider sind die meisten Beweise auf der ND-Ebene nicht wirklich verständlich. Im Gebiet der Generierung natürlicher Sprache wurde untersucht, wie ND-Beweise direkt verbalisiert werden können, wobei sich jedoch herausgestellt hat, daß eine direkte Verbalisierung von ND-Beweisen keinen adäquaten Text produziert, da die durch die jeweiligen ND-Inferenzregeln begründeten Ableitungen (im Vergleich zu Ableitungen, wie man sie in typischen mathematischen Lehrbüchern findet) auf einer viel zu niedrigen Abstraktionsebene liegen.
Wir haben deshalb einen rekonstruktiven Ansatz zu diesem Problem gewählt, in dem die Präsentation eines ND-Beweis vom PROVERB-System in zwei Stufen durchgeführt wird [5]: Das System sucht zuerst nach einer neuen Repräsentation des Beweises, die abstraktere Ableitungsschritte enthält und besser strukturiert ist. In einer zweiten Stufe verbalisiert dann die Präsentationskomponente den so abstrahierten Beweis.

3.1 Ein Modell für informelles mathematisches Beweisen

Das hier vorgestellte Verarbeitungsmodell definiert eine Zwischenrepräsentation, die einen geeigneteren Ausgangspunkt für die Präsentation der Beweise darstellt; insbesondere müssen dabei Ableitungsoperatoren definiert werden, die abstrakter sind als

die üblichen logischen ND-Inferenzregeln.
Sieht man sich die Beweise in mathematischen Lehrbüchern an, so stellt man fest, daß die meisten dort präsentierten Ableitungen durch die Anwendung eines *Faktums* (das heißt einer Definition, eines Axioms oder eines bewiesenen Theorems) und nicht allein durch einen einzelnen logischen Inferenzschritt begründet sind. Zum Beispiel kann man aus den Prämissen $U_1 \subset F_1$ und $a_1 \in U_1$, die Schlußfolgerung $a_1 \in F_1$ mittels einer Anwendung der Teilmengendefinition ableiten. Logisch gesehen erklären wir die Anwendung eines Faktums durch die Existenz eines Teilbeweisbaums. Mit solchen Bäumen kann man zu jedem Faktum eine endliche Menge von *bereichspezifischen* Inferenzregeln erzeugen, die alle Fälle der Anwendung dieses Faktums abdeckt.
Als eine neue Zwischenrepräsentation spielen *Faktenbeweise* eine entscheidene Rolle in unserem Ansatz zur Beweispräsentation. Bevor die Präsentationstechniken der Sprachgenerierung in der zweiten Stufe angewendet werden können, müssen also zuerst aus maschinell erzeugten ND-Beweisen abstraktere Beweise erzeugt werden, deren Ableitungen zum großen Teil aus der Anwendung von Fakten bestehen.
Die praktischen Erfahrungen mit unserer Implementierung zeigen, daß ein breites Spektrum von ND-Beweisen auf diese Weise wesentlich gekürzt werden kann, der Faktor liegt bei drei- bis zehnmal kürzeren Beweisen. Darüberhinaus entsprechen die Ableitungen auf der Faktenebene sehr gut der Intuition von Mathematikern, die mechanisch erzeugten Beweise waren in den von uns untersuchten Beispielen fast immer mit den im Lehrbuch angegebenen vergleichbar.

3.2 Ein Modell für die Beweispräsentation

Zur Sprachgenerierung soll nun ein Modell vorgestellt werden, wie aus einem Faktenbeweis eine Folge von Beweiskommunikationsaktionen (proof communicative acts, PCAs) erzeugt werden kann, die am Ende (nach einer Reihe linguistischer Transformationen) den natürlichsprachlichen Beweis erzeugen. Jede PCA entspricht einem Ableitungsschritt, in dem die Information über die benutzten Prämissen, über das neue Zwischenergebnis und über die Begründung enthalten sind. Im folgenden sehen wir eine PCA, die aus der Regel zur Anwendung der Teilmengendefinition erzeugt wird:

(Derive Reasons: ($a \in S_1$, $S_1 \subset S_2$)
Derived-Formula: $a \in S_2$
Method: Def-Subset)

Daraus erzeugt PROVERB die folgende Verbalisierung:

> "Because a is an element of S_1 and S_1 is a subset of S_2, according to the definition of subset, a is an element of S_2."

Die Präsentation eines Beweises wird teilweise durch *hierarchische Planung* und teilweise durch einen *lokalen Fokus* gesteuert. Im Modus der hierarchischen Planung muß

das System zu jedem Zeitpunkt entscheiden, wie der zu präsentierende Beweis in Teilbeweise aufgespalten werden kann, in welcher Reihenfolge die Teilbeweise präsentiert werden, welche PCAs den Übergang von einem Teilbeweis zum nächsten vermitteln sollen und welche PCAs die primitiven Teilbeweise präsentieren sollen. Die Verbalisierung der PCAs erfolgt schließlich durch das linguistische System TAG-GEN [9] zur syntaktischen Generierung.

4 Ein Beispiel

In diesem Abschnitt soll gezeigt werden, wie eine mathematische Aussage in Ω–MKRP bewiesen werden kann. Als Beispiel haben wir die Aussage gewählt, daß die transitive Hülle der Vereinigung zweier Relationen gleich der transitiven Hülle der Vereinigung ihrer transitiven Hüllen ist:

$$(\rho \cup \sigma)^* = (\rho^* \cup \sigma^*)^*$$

Obwohl wir aus Darstellungsgründen ein recht einfaches Beispiel gewählt haben, läßt es sich mit den heutigen vollautomatischen Beweisern allein nicht ohne weiteres lösen. Als erstes muß sich der Benutzer in der Datenbank die notwendigen Axiome, Definitionen und Hilfssätze auswählen, die für den Beweis notwendig sind bzw. die er benutzen will. In diesem Fall wählen wir die Definitionen der *transitiven Hülle*, der *Vereinigung* und der *Mengengleichheit*. Natürlich könnte diese Auswahl auch semiautomatisch auf der Basis der verwendeten Symbole erfolgen, und in der Tat wird das System diese Auswahl in Zukunft unterstützen. Bei der Definition der transitiven Hülle haben wir die Wahl zwischen zwei äquivalenten Definitionen. Wir wählen die Definition, die die transitive Hülle einer Relation σ als die kleinste transitive Relation beschreibt, die σ enthält. Während des Beweises werden wir später feststellen, daß wir zusätzlich noch die Definition für die *Teilmengenbeziehung* benötigen.
Nun können wir mit dem Beweisen beginnen und wenden als erstes die Definition der Gleichheit von Mengen auf das Beweisziel an. Hierbei handelt es sich um die Anwendung eines Faktums (siehe Abschnitt 3), der am häufigsten verwendeten Schlußweise in Ω–MKRP, für die natürlich eine Taktik vorhanden ist. Als Ergebnis erhalten wir eine Konjunktion, die wir in die beiden Hauptteilziele $(\rho \cup \sigma)^* \subseteq (\rho^* \cup \sigma^*)^*$ und $(\rho^* \cup \sigma^*)^* \subseteq (\rho \cup \sigma)^*$ aufspalten können.
Nach weiteren Anwendungen von Definitionen auf die erste Formel und anschließender Simplifikation erhalten wir die Formel $\rho \cup \sigma \subseteq (\rho^* \cup \sigma^*)^*$. An dieser Stelle ist eine spezielle Taktik anwendbar, die die Homomorphieeigenschaft von $\cup$ für $\subseteq$ ausnutzt (siehe Abschnitt. 2.2). Als Ergebnis erhalten wir die beiden Teilziele $\rho \subseteq (\rho^* \cup \sigma^*)^*$ und $\sigma \subseteq (\rho^* \cup \sigma^*)^*$. Nun haben wir die Problemstellung genügend zerlegt, so daß sie von einem der angeschlossenen automatischen Beweiser in angemessener Zeit gelöst werden kann. Das zweite Hauptteilziel ist etwas schwieriger, insgesamt erhalten wir vier Teilprobleme, mit denen wir nun alternativ die Beweissysteme MKRP oder OTTER aufrufen.

Bei diesen Aufrufen wird als erstes unsere Repräsentation des Beispiels, das ja wegen der obigen Relationendarstellung eine Logik höherer Stufe voraussetzt, von $\mathcal{POST}$ in eine Repräsentation der Logik erster Stufe übersetzt. Dann werden die entstandenen Formeln in Klauselnormalform transformiert und dem gewählten Beweiser in dessen Syntax übergeben. Nachdem ein Beweis gefunden ist, wird das spezielle Ausgabeprotokoll des Beweisers in eine einheitliche Sprache übersetzt, wobei eventuell im Protokoll fehlende Informationen (wie zum Beispiel die angewendeten Substitutionen) wiederberechnet werden müssen. Dieser einheitlich repräsentierte Resolutionsbeweis wird dann in einen Beweis transformiert, der in dem in Ω–MKRP benutzten Format, dem Kalkül des natürlichen Schließens, formuliert ist. Anschließend wird dieser Beweis wieder von der Logik erster Stufe in die ursprüngliche Logik höherer Stufe übersetzt. Dieser Beweis kann nun in den Beweisbaum in Ω–MKRP eingefügt werden. Nachdem wir die vier Teilprobleme in unserem Beispiel mit einem der automatischen Beweiser gelöst haben, erhalten wir einen vollständigen ND-Beweis. Dieser kann nun mit der oben beschriebenen Beweistransformation abstrahiert und in natürliche Sprache transformiert werden (siehe Abschnitt 3).

5 Zusammenfassung

Die Entwicklung, die zu dem beschriebenen Ω–MKRP-System geführt hat, steht natürlich nicht für sich allein, sondern muß im Grenzbereich zwischen automatischen Beweisern, Beweisprüfern und planbasierten Systemen zur Beweiserstellung gesehen werden. Damit wird im Rahmen dieses Systems versucht, verschiedene Paradigmen der Beweiserstellung zu verbinden. Automatische Beweiser bauen meist auf einem maschinenorientierten Kalkül wie dem Resolutionsverfahren auf: man spezifiziert zu Beginn einer Beweissuche ein Problem, setzt gewisse Parameter, und dann versucht das System ohne weitere Benutzerinteraktion einen Beweis zu finden. Die meisten automatischen Beweiser bauen auf einer Variante der Logik erster Stufe auf, es gibt aber zum einen spezielle Systeme zur Behandlung von vollständiger Induktion wie den Boyer-Moore-Beweiser oder das INKA-System. Zum anderen gibt es Systeme, die auf einer Logik höherer Stufe aufbauen, wie das TPS-System. Das letzte System hat sowohl einen automatischen als auch einen interaktiven Modus.
Systeme zur Beweisprüfung wie AUTOMATH [1] wurden auch benutzt, um mathematische Lehrbücher streng formal zu beweisen. Sie nutzen in stärkerem Umfang bereichsspezifisches Wissen aus, aber die eigentliche Beweislast liegt fast ausschließlich beim Benutzer. Fortgeschrittene Systeme sind taktikbasierte Systeme wie NUPRL, Isabelle (Isabelle erlaubt zusätzlich die Spezifikation der Objektlogik) und KIV oder wissensbasierte Systeme wie Muscadet und Ontic. Die Taktiken wurden im OYSTER-CLAM-System [3] zu Methoden erweitert, so daß eine deklarative Spezifikation von Taktiken angegeben werden kann.
Wir sehen die wachsende Aufmerksamkeit, die dem Bereich des taktikbasierten und planbasierten Beweisen gewidmet wird, nicht als Zufall, sondern als ein Indiz dafür

an, daß vollautomatische Beweiser zwar starke Hilfmittel, aber als eigenständige Systeme nur von einem begrenzten Nutzen sind. Sie entfalten ihre Stärke vor allem als integrierte Systeme in einer interaktiven Umgebung.

Literatur

[1] N. G. d. 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.

[2] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk, R. Overbeek, Hrsg., *Proc. of the 9th CADE*, S. 111–120. Springer, LNCS 310, 1988.

[3] A. Bundy, F. van Harmelen, C. Horn, A. Smaill. The OYSTER-CLAM system. In [16], S. 647–648.

[4] P. Deussen. *Halbgruppen und Automaten.* Springer, 1971.

[5] X. Huang. PROVERB: A system explaining machine-found proofs. In A. Ram, K. Eiselt, Hrsg., *Proc. of 16th Annual Conference of the Cognitive Science Society*, S. 427–432. Lawrence Erlbaum Associates, 1994.

[6] X. Huang, M. Kerber, M. Kohlhase, J. Richts. Adapting methods to novel tasks in proof planning. In B. Nebel, L. Dreschler-Fischer, Hrsg., *KI-94: Advances in Artificial Intelligence – Proc. of KI-94.* Springer, LNAI 861, 1994, S. 379–390.

[7] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts, J. Siekmann. Ω-MKRP*, ein mathematisches Assistenzsystem*, SEKI Working Paper SWP-95-01, Universität des Saarlandes, 1995.

[8] M. Kerber. On the translation of higher-order problems into first-order logic. In T. Cohn, Hrsg., *Proc. of ECAI-94*, S. 145–149. John Wiley & Sons, 1994.

[9] A. Kilger, W. Finkler. TAG-based incremental generation. *Computational Linguistics*, 1995. Im Druck.

[10] M. Kerber, M. Kohlhase. A tableau calculus for partial functions. In *Annals of the Kurt-Gödel-Society.* Springer, 1995. Im Druck.

[11] M. Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle.* Dissertation, Universität des Saarlandes, 1994.

[12] C. Lingenfelder. *Transformation and Structuring of Computer Generated Proofs.* Dissertation, Universität Kaiserslautern, 1990.

[13] E. Melis. A model of analogy-driven proof-plan construction. In C. Mellish, Hrsg., *Proc. of the 14th IJCAI*, S. 182–188. Morgan Kaufman, 1995.

[14] S. Owen. *Analogy for Automated Reasoning.* Academic Press, 1990.

[15] F. Pfenning, D. Nesmith. Presenting intuitive deductions via symmetric simplification. In [16], S. 336–350.

[16] M. E. Stickel, Hrsg. *Proc. of the 10th CADE.* Springer, LNAI 449, 1990.