Kognitionswissenschaft manuscript No.
(will be inserted by the editor)

ΩMEGA: Ein mathematisches Assistenzsystem

Jörg Siekmann, Michael Kohlhase, Erica Melis *

Universität des Saarlandes, FB Informatik, D-66041 Saarbrücken

Received: date / Revised version: date

ΩMEGA: A Mathematical Assistant System

Zusammenfassung ΩMEGA is a deduction system for the mathematical practice and for and mathematics education. The underlying vision is that of an automated mathematical assistant that supports the working mathematician in many tasks. The current system consists of a proof planner and an integrated collection of tools for formulating problems, proving subproblems, and proof presentation. In the otherwise more technology-dominated field of automated theorem proving, ΩMEGA is one of the few systems that are cognitively motivated as well.

Zusammenfassung ΩMEGA ist ein Deduktionssystem, das für die mathematische Forschung und Lehre entwickelt wird. Die zugrundeliegende Vision ist die eines automatischen mathematischen Assistezsystems, das den Mathematiker bei allen anfallenden Aufgaben unterstützt. Das System besteht aus einem Beweisplaner und einer Reihe integrierter Unterstützungssysteme, die bei der Problemformulierung, beim Beweisen von Teilproblemen und bei der Präsentation der Beweise helfen können. Die Systemkonzeption stellt im sonst eher technologiedominerten automatischen Beweisens einen der wenigen auch kognitiv motivierten Ansätze dar.

1 Motivation

Deduktionssysteme sollen Menschen bei der Entwicklung mathematischer Beweise unterstützen – aber sie werden bis heute in der mathematischen Ausbildung und Praxis kaum genutzt. Das liegt unserer Meinung nach unter anderem an den verwandten Inferenzverfahren, die kein spezifisch mathematisches Wissen kodieren, und daran, daß diese Systeme nicht genügend benutzerorientiert sind und letzlich daran, daß die Performanz noch recht beschränkt ist.

Sehen wir uns die bisherigen Systeme etwas genauer an: Klassische automatische Deduktionssysteme wie etwa OTTER und SPASS versuchen ein gegebenes mathematisches Theorem aus einer kleinen Menge von Axiomen durch *Suche auf der Kalkülebene* zu beweisen. Das heißt, das Theorem folgt aus diesen Axiomen, wenn es gelingt eine Kette von Inferenzschritten, die den Regeln eines *Kalküls* entsprechen, zu finden, die mit den Axiomen beginnt und mit dem Theorem endet. Die einzelnen Verfahren (wir nennen hier stellvertretend das Resolutionsverfahren und Tableaux-Verfahren (siehe (Bibel & Schmitt, 1998) für einen Überblick) unterscheiden sich in der Repräsentation des Suchraums und in der Auswahl der jeweiligen Inferenzschritte. In jedem dieser logischen Systeme sind die Inferenzschritte jedoch viel kleiner als die Schlußfolgerungsschritte, die Menschen anwenden und die Suchheuristiken sind nicht durch mathematische Inhalte und Vorgehensweisen bestimmt, sondern lokal und syntaktisch.

Trotz der enormen Leistungssteigerung der auf auf diesen Verfahren basierenden Systeme, die dazu führte, daß selbst offene mathematische Probleme automatisch bewiesen werden konnten (McCune, 1997), ist der Einsatz solcher Systeme in der Mathematik begrenzt. Generell stellt sich – nicht zuletzt wegen der kombinatorischen Explosion der Suchräume – die Frage, ob diese automatischen Systeme jemals komplexe Beweise, die auch für Menschen schwierig sind, in mathematisch wohlverstandenen Domänen werden finden können.

In interaktiven, taktikbasierten Systemen wie etwa NuPrl oder Isabelle (siehe (Bibel & Schmitt, 1998) für einen Überblick) gibt der Benutzer den Beweis vollständig vor. Dabei können häufig vorkommende Inferenzfolgen zu sogenannten Taktiken zusammengefaßt werden, so daß der Benutzer des Systems nur noch eine Abfolge von Taktiken angeben muß. Diese Art der rechnergestützten Beweisentwicklung unterliegt natürlich nicht der kombinatorischen Explosion, aber der relativ niedrige Auto-

* Die dem Artikel zugrunde liegende Arbeit wurde von der DFG im SFB 378 gefördert.

matisierungsgrad macht die praktische Arbeit mit diesen Systemen sehr aufwendig.

Obwohl diese beiden prinzipiellen Vorgehensweisen heute die dominanten Paradigmen unseres Forschungsgebietes sind, muß dies keineswegs zwingend oder notwendig so sein – und ist im historischen Rückblick auch keinesfalls immer so gewesen. Schon 1956 auf der Darthmouth Conference, von vielen als Geburtsstunde der Künstlichen Intelligenz (KI) und des automatischen Beweisens angesehen, wurden zwei große Forschungsrichtungen sichtbar: Zum einen der Ansatz, der beispielsweise M. Davis' Programm zugrunde lag und der dem oben beschriebenen Paradigma klassischer automatischer Beweiser am nächsten kommt; er implementierte eine Entscheidungsprozedur für die Presburger Arithmetik, die ein gegebenes Theorem maschinell beweisen kann. Zum Zweiten wurden dort aber auch KI-orientierte Ansätze vorgestellt, die sich an der Vorgehensweise von Mathematikern orientierten, wie in dem damals prominentesten System des „Logical Theorist" von A. Newell und H. Simon und später in den Arbeiten von W. Bledsoe.

Das im folgenden vorgestellte ΩMEGA System steht wieder deutlicher in der ursprünglichen geistigen Tradition der KI-orientierten Deduktionssysteme. Die zugrundeliegende Vision ist ein System, das den Mathematiker oder Studenten bei den anfallenden Aufgaben unterstützt und von Routineaufgaben entlastet. Als Konsequenz soll das ΩMEGA-System weder als vollautomatisches System noch als einfaches taktisches Deduktionssystem arbeiten, sondern die Vorzüge beider Verfahren integrieren. Das ΩMEGA-System benutzt den Ansatz der Beweisplanung, der eher der Vorgehensweise von Mathematikern entspricht. ΩMEGA integriert darin sowohl das interaktive Theorembeweisen als auch die klassische Beweissuche. Dabei soll der Automatisierungsgrad schrittweise erhöht werden, um längerfristig eine wirkliche Systemunterstützung bei der Beweisentwicklung zu erreichen.

Das Beweisplanungsparadigma basiert auf der Beobachtung, daß Mathematiker anhand von bereichsspezifischem Wissen zuerst einen Beweisplan entwickeln, der aus Zwischenzielen und Methoden, die diese Ziele erreichen, besteht. Die Beweisplanung setzt daher Pläne aus sogenannten Methoden zusammen, die zu einem vollständigen Beweis auf Kalkülebene expandiert werden können. Methoden enkapsulieren typische mathematische Vorgehensweisen und sind als frame-artige Datenstrukturen mit Vor- und Nachbedingungen implementiert. Bei dieser Expansion können dann wieder Planungsprozesse auftreten, aber auch Prozesse mit externen Systemen, wie z.B. einem automatischen Beweissystem oder einem System der Computeralgebra. Zu einer weitergehenden kognitiven Motivation der Beweisplanung verweisen wir auf (Melis, 1997).

2 Architektur des Systems

Das ΩMEGA-System (siehe Abbildung 1) besteht aus einer Reihe relativ unabhängiger Komponenten, die gemeinsam einen partiellen Beweis bearbeiten, der durch eine hierarchische Datenstruktur, die *Proof Plan Data Structure* ($\mathcal{PDS}$) repräsentiert ist. Diese zentrale Datenstruktur, auf die alle Komponenten von ΩMEGA, einschließlich des Benutzers, zugreifen, repräsentiert einen (partiellen) Beweis auf verschiedenen Abstraktionsebenen. Die Rechtfertigung eines Beweisschritts auf einer

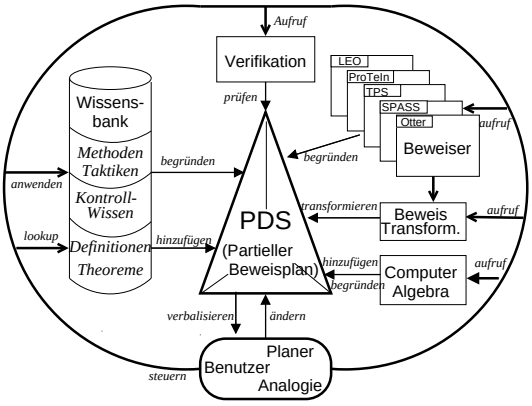

Abb. 1 Architektur des mathematischen Assistenzsystems ΩMEGA

abstrakteren Ebene kann bei Bedarf (rekursiv) zu einer detaillierteren Begründung expandiert werden. Die Expansion kann fortgesetzt werden bis hinab zur Ebene des logischen Kalküls, auf der die Beweise dann automatisch auf Korrektheit überprüft werden können. Im Gegensatz zu anderen Systemen speichert ΩMEGA alle hierarchischen Ebenen in der $\mathcal{PDS}$, und diese hierarchische Struktur kann später wiederverwendet werden, beispielsweise zum Beweisen durch Analogiebildung oder für die Präsentation eines Beweises auf verschiedenen Abstraktionsebenen.

Als mathematisches Assistenzsystem führt ΩMEGA die Suche nach einem Beweis in enger Kooperation mit dem Benutzer. Insbesondere hat dieser die Möglichkeit, einen Beweis ganz manuell durchzuführen – das heißt in diesem Fall arbeitet ΩMEGA wie ein taktisches Beweissystem – oder sich durch einen Beweisplaner oder andere Hilfesysteme (wie zum Beispiel klassische automatische Beweiser oder Computeralgebra-Systeme) bei der Beweissuche unterstützen zu lassen, schließlich kann der Benutzer die Beweissuche auch ganz dem System zu überlassen, d.h. das Beweissystem läuft in verschiedenen Modi, die vom Benutzer gewünscht werden können. Das nötige Wissen kann entweder aus einer integrierten Wissensbank automatisch aktiviert werden oder vom Benutzer zur Verfügung gestellt werden.

3 Beispiel

Um den Prozeß der Beweisentwicklung in ΩMEGA zu zeigen, soll uns ein Beispiel helfen; eine genauere Beschreibung des Beweisplaners und der externen Systeme folgt

danach. Das Beispiel ist eine Optimierungsaufgabe und ist einer Diplom-Klausur der Wirtschaftswissenschaften an der Universität des Saarlandes entnommen.

> **Problem:** *Die Produktionsrate d einer Maschine kann über ein Intervall $I = [1, 7]$ eingestellt werden. Die Kosten des erzeugten Produkts prod hängen ab vom Wasser- und Elektrizitätsverbrauch bei der Produktion einerseits (diese sind durch die folgenden Kostenfunktionen gegeben)*
> - $r_1 = (0.5d^2 + 3)\ \frac{m^3}{prod}$
> - $r_2 = (4d^2 - 24d + 24)\ \frac{kWh}{prod}$
>
> *und andererseits von den Preisen von Wasser und Elektrizität*
> - $p_1 = 2\ \frac{DM}{m^3}$
> - $p_2 = 0.5\ \frac{DM}{kWh}$
>
> *Bestimmen Sie die Produktionsrate d, so daß die totalen Produktionskosten minimal sind.* ■

Der erste Schritt der Problemlösung besteht nun darin, das relevante Problemwissen in der Wissensbank von ΩMEGA zu finden und zu aktivieren. In diesem Fall werden die Theorien `size` für Maßgrößen (in unserem Beispiel Produktionsfaktoren und -Kosten) und `calculus` für elementare reelle Analysis (hier Minimierung von Funktionen) gefunden und bereitgestellt. Die Theorie `economy`, in der unser Beispiel gelöst werden kann, erbt das Wissen aus beiden, und stellt weiterhin die konkreten Einheiten DM, m^3, kWh und *prod* zur Verfügung. Mit diesem Wissen läßt sich die Optimierungsaufgabe durch das folgende Theorem ausdrücken:

> **Theorem:** *Es gibt eine reelle Zahl $d \in I = [1, 7]$, so daß d ein Optimum der Kostenfunktion $r_1 \oplus r_2$ in I ist.*

Dabei ist $\oplus$ der Summenoperator auf Kostenfuntionen aus der Theorie `size`. Das Prozeßwissen über diesen Operator wird dort durch die Methoden `mult-by-price` und `add-by-denom` formalisiert: punktweise Multiplikation einer Kostenfunktion mit einem Preis (dabei werden die Einheiten entsprechend angepaßt) und punktweise Addition von zwei Kostenfunktionen (falls die Einheiten übereinstimmen). Weiterhin gibt es dort die Methode `optimize`, die ein Optimierungsproblem von Kostenfunktionen auf das Finden eines globalen Minimums der zugehörigen reellwertigen Funktion auf dem gegebenen Intervall zurückführt. Beispielsweise überführt die Methode `add-by-denom` eine Summe aus Kostenfunktionen in eine einzelne Kostenfunktion falls die Einheiten übereinstimmen.

Der Planer setzt dann den folgenden Beweisplan für das Theorem zusammen:

```
1  Mult-by-Price
2  Mult-by-Price
3  Add-by-Denom
4  Optimise
5  TotMin-Rolle
```

Dieser Beweisplan entspricht dem Vorgehen eines Studenten in der WiWi-Klausur: zuerst müssen die Einheiten der beiden Kostenfunktionen auf *DM/prod* normiert werden, bevor sie addiert werden können und damit kann das jetzt explizit in der Aufgabe enthaltene Optimierungsproblem gelöst werden. Dies geschieht mit der Methode `totmin-rolle` aus der Theorie `calculus`, die den Satz von Rolle ausnutzt, der bei Polynomen zweiter Ordnung das globale Minimierungsproblem auf ein lokales zurückführt. Dieser Satz wird durch die wohlbekannten Bedingungen an die ersten zwei Ableitungen anwendbar, wobei die Methode zur Berechnung der Ableitungen auf das in ΩMEGA integrierte Computeralgebrasystem μ-CAS zurückgreift.

Da der obige Beweisplan keine offenen Teilziele mehr hat, ist der Plan vollständig. Allerdings ist dieser Beweisplan noch kein nachprüfbarer Beweis auf Kalkülebene. D.h. die Methoden müssen nun rekursiv expandiert werden und gegebenenfalls müssen durch einen klassischen Beweiser die „Lücken“ geschlossen werden. Insbesondere wenn der Benutzer der Korrektheit des Computeralgebrasystems nicht vertraut oder die Berechnung erklärt bekommen möchte, muß der Beweisplan im angegebenen Sinne ausgeführt werden. Das heißt, die Methoden müssen durch Beweissegmente ersetzt werden, die sie repräsentieren. Für den vorliegenden Beweisplan ergibt die Expansion einen Kalkülbeweis mit mehr als 350 Schritten.

4 Beweisplanung

Das Beweisplanungsparadigma wurde technisch erstmals von A. Bundy für induktive Beweise vorgeschlagen. Spezifisch für die Beweisplanung sind Planungsoperatoren, *Methoden* genannt, die typische wiederkehrende Beweisabläufe repräsentieren sowie *globale* und *domänenspezifische Steuerungsmechanismen* für den Planungsprozess (Melis & Bundy, 1996).

Eine Methode enthält eine Spezifikation der Veränderung des Zustands der $\mathcal{PDS}$, durch diese Methodenanwendung. Anhand dieser Spezifikationen setzt der Planer dann einen globalen Beweisplan für das Theorem zusammen. Das eigentliche Planen eines Beweises basiert im wesentlichen auf Standardtechniken der Planverfahren in der KI, die jedoch den speziellen Anforderungen des Theorembeweisens angepaßt wurden. Wie bei Planungsverfahren üblich, beginnt die Beweisplanung mit einem partiellen Plan, der durch das zu beweisende Theorem (offenes Ziel) und die Beweisannahmen gegeben ist. Offene Ziele werden durch Methoden, die in den Plan eingefügt werden, geschlossen. Die eingefügten Methoden erzeugen eventuell erneut Teilziele, die geschlossen werden müssen. Die Planung ist beendet, wenn es keine offenen Ziele mehr gibt. Die zur Verfügung stehenden Methoden werden dem Planer als Domänenwissen theoriespezifisch übergeben.

Voraussetzung dafür, daß ein Beweis überhaupt automatisch geplant werden kann, ist natürlich, daß ge-

eignete Methoden und das Steuerungswissen schon bekannt sind. Die Wissensakquisition, die dafür notwendig ist, gestaltet sich in mathematischen Domänen oftmals noch schwieriger als beispielsweise in der Klötzchenwelt oder anderen Anwendungsgebieten, in denen ein gutes intuitives Vorverständnis der Domäne vorhanden ist. Ein wesentlicher Teil der Arbeit im ΩMEGA-Projekt war daher die Entwicklung von geeigneten Repräsentationsmechanismen für das faktische, das methodische und das Steuerungswissen, das aus bekannten mathematischen Beweisen extrahiert wurde (Cheikhrouhou, 1997; Melis, 1998).

4.1 Techniken zur Beherrschung des Suchraumes

Hierarchisches Planen in ΩMEGA kann entweder durch Expansion von abstrakten Methoden oder durch Verschieben weniger wichtiger Teilziele realisiert werden. Das hierarchische Planen kann jederzeit unterbrochen werden und liefert, je nach verbrauchter Zeit, unterschiedlich detaillierte Beweispläne. Hierarchisches Planen ist eine prinzipielle Möglichkeit, die Suchräume einzuschränken, allerdings ist dies nicht ausreichend: Je mehr Methoden anwendbar sind, desto größer kann der Suchraum beim Planen werden. In ΩMEGA wird die Suche daher weiter begrenzt durch die Auswertung von Kontrollwissen, durch Constraintlösen und durch bestimmte Planungsstrategien, die im folgenden erläutert werden.

Zu den Planungsstrategien in ΩMEGA, die die Suche verringern oder ganz vermeiden gehört die analogiegeleitete Beweisplankonstruktion, die den Beweisplan eines Quellproblems in einen partiellen Plan für ein Zielproblem überführen kann (Melis, in dieser Zeitschrift, 1997). Mit dieser Technik kann man häufig Sätze beweisen, die sonst weit außerhalb der Möglichkeit des automatischen Systems lägen, da sie Methoden *vorschlägt* anstatt nach anwendbaren Methoden zu *suchen*.

Durch die Auswertung von Steuerungswissen, das in ΩMEGA in Form von deklarativen Kontrollregeln kodiert ist, kann die Suche nach geeigneten Methoden oder nächsten Zielen eingeschränkt werden. Die Kontrollregeln können Bedingungen an die $\mathcal{PDS}$ und deren Entstehungsgeschichte enthalten ("Wenn die zuletzt angewandte Methode X war, dann wende danach Methode Y an."), sie können Bedingungen aus dem Constraintzustand enthalten („Ist der jetzige Constraintzustand eingeschränkter als der vor der Anwendung von Methode X, so bevorzuge als nächstes zu bearbeitendes Ziel Y."), und sie können Bedingungen an die vorhandenen Ressourcen enthalten („Sind die Zeitressourcen kleiner als X, dann bearbeite Ziel Y erst am Schluß.").

Schließlich hilft die Einbindung von theorie-spezifischen Constraintlösern in die Beweisplanung, die Suche nach mathematischen Objekten einzuschränken, deren Existenz behauptet wird. Sie sammeln Constraints an diese Objekte, die in der Beweisplanung auftreten, und testen, ob die Menge der Constraints noch erfüllbar ist. Beweisversuche, in denen widersprüchliche Constraints auftreten, wie etwa $X > 3$ und $X < 3$, werden abgebrochen.

4.2 Ressourcenverwaltung in ΩMEGAs Planer

In einem mathematischen Assistenzsystem – wie wir es verstehen – tritt die *Benutzerinteraktion* als zentrale Ressource auf. *Zeit und Speicherplatz* sind sekundäre Ressourcen, wobei die Zeitressource allerdings wieder mit der Benutzerinteraktion gekoppelt ist, da sie (über Wartezeiten) mit den Zeitressourcen des Benutzers interagiert. Schließlich werden in ΩMEGA auch *Information* bzw. *Wissen* als Ressourcen behandelt.

Durch die explizite Repräsentation und Verwaltung von Ressourcen kann der Kontrollfluß in einem so komplexen System gesteuert werden. Hierfür gibt es mehrere Realisierungsmöglichkeiten: Eine ressourcenverwaltende Architektur, ressourcenadaptive Teilprozesse (wie im hierarchischen Planen) und explizites Schlußfolgern über Ressourcen (wie in der Auswertung von Kontrollregeln). Alle drei Möglichkeiten werden von uns untersucht und eingesetzt. Im folgenden wollen wir nun die Verwaltung der *Ressource Benutzerinteraktion* in ΩMEGAs Planer als Beispiel näher betrachten.

Die Unterstützung des Systems durch den Benutzer ist der Vorteil eines interaktiven Systems. Trotzdem muß mit derartiger Interaktion sparsam umgegangen werden, wenn das System für den jeweiligen Benutzer von Interesse sein soll. Damit das wichtigste Subsystem, der Planer durch den letztlich die Beweissuche gesteuert und durchgeführt wird, ressourcenadaptierend[1] arbeiten kann, muß er abschätzen können, wann eine Anfrage an den Benutzer sinnvoll ist. Dazu muß das System Information über die zeitliche Verfügbarkeit des Benutzers haben.

Der Benutzer kann das System in verschiedener Weise nutzen wollen: Er kann beispielsweise einen echten Dialog führen wollen, das bedeutet, er ist jederzeit zu Antworten auf Fragen bereit – im Extremfall könnte jede einzelne Entscheidung über die Anwendung des besten Planoperators vom System erfragt werden. Natürlich sind dann kurze Antwortzeiten vom System gefordert. Im anderen Extrem läßt der Benutzer das System über Nacht (oder etwa über die Mittagspause) eigenständig arbeiten und dieses präsentiert danach Angaben über (Teil-)Erfolge. Daraus ergibt sich eine *ressourcenadaptive* Parametrisierung des Planer mit folgenden Verarbeitungsmodi:

`Automatisch` Hier unterscheiden wir weiter in den *vollautomatischen Modus* und den *Beweisidee-Modus*, in dem der Benutzer eine initiale Beweisskizze vorgibt, und das System dann versucht, diesen partiellen Beweis zu einem vollständigen Beweis automatisch zu expandieren.

[1] Im folgenden benutzen wir die Begriffe ressourcenadaptiert, ressourcenabhängig und ressourcenadaptierend, wie in diesem Heft eingeführt ((siehe ?).

`Interaktiv` Hier unterscheiden wir drei konkrete Modi:

1. *Dialogmodus*: Die Initiative liegt beim Planer, der bei Bedarf Fragen an den Benutzer richten kann.
2. *Unterstützungsmodus*: Hier hat der Benutzer die Kontrolle. Der Planer verwaltet lediglich den Beweisbaum und wird nur auf Anweisung aktiviert.
3. *Vorschlagsmodus*: Die Kontrolle liegt zunächst beim Planer. Der Benutzer verfolgt jedoch über ein Display die Schritte und kann jederzeit intervenieren und steuernd eingreifen.

Um diese Modi adäquat realisieren zu können, muß der Planer abschätzen können, welche Fragen der Benutzer vermutlich beantworten kann, wie hilfreich die erwartete Antwort ist und wie häufig der Benutzer „belästigt" werden darf. Hierfür ist ein *adaptierendes* Vorgehen angemessen, das die Anpassung an die Expertise des Benutzers durch meta-kognitives Schließen erreichen kann.

5 Integration externer Unterstützungssysteme

Wie im Beispiel von Abschnitt 3 deutlich wurde, kann ΩMEGA für bestimmte Teilaufgaben andere automatische Systeme, wie etwa μ-CAS aufrufen. Beispielweise werden dort zum Zweck der Expansion von `Optimise` die Berechnungsschritte der Ableitungen, die in μ-CAS erfolgten, als Beweisplan ausgegeben (so wie ein Student auf Anfrage alle Berechnungsschritte offenlegen kann). Durch die Anbindung leistungsstarker autarker automatischer Komponenten, die für bestimmte Aufgaben spezialisiert sind, kann die Performanz eines mathematischen Assistenzsystems gesteigert werden. Daher integriert ΩMEGA verschiedene spezialisierte Deduktionssysteme, wie zum Beispiel die automatischen Beweissysteme OTTER, SPASS und das Computeralgebra-System μ-CAS (Kerber, Kohlhase & Sorge, 1998) die für die Lögsung bestimmter Teilprobleme aufgerufen werden können. Die Beweise, die von diesen Systemen gefunden wurden, werden in das $\mathcal{PDS}$-Format transformiert, um die Korrektheit des Gesamtbeweises prüfen zu können.

Automatische Beweiser Während der Planer seinen internen Zustand bei beliebiger ressourcenabhängiger Unterbrechung bewahrt und in jedem Fall ein interpretierbares Resultat (eben einen partiellen Beweis) liefert, ist dies bei herkömmlichen Beweisern nicht möglich, und es ist sehr schwierig, sinnvolle Information aus unvollendeten Läufen (einer Klauselmenge von möglicherweise Millionen von Klauseln) zurückzugewinnen. In der Konzeption des automatischen Beweisers LEO (Benzmüller & Kohlhase, 1998) für Logik höherer Stufe, der von ΩMEGA aus aufgerufen werden kann, ist die Speicherung von sinnvollen Informationen aus unvollständigen Läufen dagegen möglich. Dies ist insbesondere wichtig im Vorschlagsmodus (siehe Abschnitt 3), da hier der interne Systemzustand kommunizierbar sein muß, um dem Benutzer ein planvolles Eingreifen zu ermöglichen.

Wissensbasis Da in ΩMEGA sehr viel theorie-spezifisches Wissen (wie z.B. Axiome, Definitionen, Theoreme und Methoden) benutzt wird, ist es notwendig, dieses in einer Hierarchie mathematischer Theorien abzulegen. Ein einfacher Vererbungsmechanismus verbindet Theorien mit den jeweiligen Elterntheorien.

Ressourenverwaltung in Unterstützungssystemen Unter dem Aspekt der Ressourcenbehandlung gibt es folgende interessante Entwicklungen in der Anbindung der externen Systeme:

Ein Vorschlagsmechanismus für Kommandos (Benzmüller & Sorge, 1998) wird in ΩMEGA durch einen agentenbasierten Ansatz realisiert. Dessen Architektur ermöglicht es, diesen Mechanismus als eigenständige verteilte Komponente auszulagern. Verfügbare Ressourcen können dadurch besser ausgenutzt werden weil die Agenten – gesteuert durch den momentanen Beweiszustand – im Hintergrund ständig nach neuen Vorschlägen suchen und diese über die graphische Benutzeroberfläche mit dem Benutzer kommunizieren.

Die in ΩMEGA eingebundenen externen automatischen Beweiser können bisher parallel und mit jeweils vorgegebenen Zeitschranken gestartet werden. Über Kontrollregeln werden wir in Zukunft auch erreichen, daß unter Berücksichtigung des Fortschritts eines Beweisers und abhängig von der Benutzer-ressource dem externen Beweiser auch zusätzliche Zeitressourcen zugeteilt werden können.

Die Speicherung von theorie-spezifischem Wissen in einer hierarchischen Wissensbasis kann für ein ressourcenadaptives Verhalten des Planers benutzt werden, indem zum Beispiel nur „nahe" Theorien nach geeigneten Axiomen durchsucht werden, oder falls mehr Ressourcen verbraucht werden dürfen, auch in der Hierarchie weiter entfernte Theorien. Allgemeiner wird *Information* in ΩMEGA als eine dem System zur Verfügung stehende Ressource begriffen, deren explizite Verwaltung ein wichtiges Steuerungsmittel darstellt. Beispielweise kann es für die Beweisplanung wichtig sein, an den kritischen Stellen die richtigen Lemmata aus der Wissensbank herauszufiltern oder zu spekulieren. Durch den strukturierten Aufbau liefert ΩMEGA's Wissensbank hierfür einen Teil der nötigen Infrastruktur.

ΩMEGA enthält das Beweispräsentationssystem PROVERB, das die $\mathcal{PDS}$ auf verschiedenen Abstraktionsebenen in englischer Sprache verbalisieren kann (Huang & Fiedler, 1997). So können Beweise produziert werden, die für Benutzer leichter verständlich sind. Erste Ergebnisse bei der Berücksichtigung von Information über die Expertise des Benutzers erlauben, die Beweispräsentation benutzeradaptiv zu gestalten (Fehrer & Horacek, 1997).

6 Bewertung und Ausblick

Um die Frage zu beantworten, welche Komponenten eines mathematischen Assistenzsystems in welcher Weise

zu integrieren sind, hat das Projekt ΩMEGA bis jetzt die Grundarchitektur eines kognitiv motivierten, mathematischen Assistenzsystem entwickelt und implementiert. Dabei wurden auf verschiedenen Gebieten erste Erfolge erzielt: einerseits durch die systematische Einbindung externer Komponenten, die helfen, den Beweis zu vervollständigen und andererseits durch die Erweiterung der Beweisplanung.

Unser Beweisplanungsansatz wurde für Klassen von Problemen erfolgreich eingesetzt: beispielsweise liegt das Klausur-Beispiel weit jenseits der Lösungsmöglichkeiten eines klassischen automatischen Beweissystems. Ebenso wurden die bekannten Limes-Theoreme[2] automatisch geplant, einschließlich des bisher nicht automatisch beweisbaren Theorems LIM*. Dieser Erfolg ist u.a. auf das Metareasoning mit domänen-spezifisches Kontrollwissen zurückzuführen.

Darüberhinaus haben wir eine Reihe von Mechanismen entwickelt, die es erlauben, das Verhalten des Systems in Abhängigkeit von den vorhandenen Ressourcen zu steuern. Diese müssen insbesondere noch für die ressourcenabhängige Interaktion der verschiedenen Komponenten erweitert werden. Unsere zukünftige Arbeit konzentriert sich also auf die Steuerung des Verhaltens der Komponenten des Systems und der Interaktion mit dem Benutzer sowie auf eine Verbesserung der bisher prototypisch implementierten Komponenten. Mit diesem Ziel wollen wir weitere Planungsstrategien und verschiedene Formen von mathematischesm Kontrollwissen untersuchen und in die Beweisplanung integrieren.

References

Benzmüller, C. & Kohlhase, M. (1998). LEO, a higher-order theorem prover. *Proceedings of the 15th International Conference on Automated Deduction*, 1998.

Benzmüller, C. & Sorge, V. (1998). An agent-based approach for guiding interactive proofs. *8th International Conference on Artificial Intelligence: Methodology, Systems, Applications*

Bibel, W. & Schmitt, P. (Eds.). (1998). *Automated Deduction – a Basis for Applications.* Kluwer.

Bundy, A. (1988). The use of explicit plans to guide inductive proofs. In E. Lusk & R. Overbeek (Eds.), *Proceedings 9th International Conference on Automated Deduction*, 111-120.

Cheikhrouhou, L. (1997). Planning diagonalization proofs. In *Proceedings of the 18th Annual German Conference on Artificial Intelligence KI'97*, 377–380. Freiburg, Germany.

Fehrer, D. & Horacek, H. (1997). Exploiting the adressee's inferential capabilities in presenting mathematical proofs. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence*, 959-970. Nagoya: Morgan Kaufmann.

Huang, X. & Fiedler, A. (1997). Proof verbalization as an application of NLG. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 965-970. Nagoya: Morgan Kaufmann.

Kerber, M., Kohlhase, M. & Sorge, V. (1998). Integrating computer algebra into proof planning. *Journal of Automated Reasoning.* (Special Issue on the Integration of Computer Algebra and Automated Deduction, in press)

McCune, W. (1997). Solution of the robbins problem. *Journal of Automated Reasoning*, *19*(3), 263–276.

Melis, E. (1997). Beweisen durch Analogie. *Kognitionswissenschaft*, *6*(3), 115-126.

Melis, E. (1998). AI-techniques in proof planning. In *European Conference on Artificail Intelligence, 1998.* Brighton: Kluwer.

Melis, E. & Bundy, A. (1996). Planning and proof planning. In S. Biundo (Ed.), *ECAI-96 Workshop on Cross-Fertilization in Planning*, 37-40. Budapest.

[2] Etwa LIM+, das besagt, daß die Summe der Limites zweier Funktionen gleich dem Limes der Summe der Funktionen ist. LIM* ist ein ähnliches Theorem für die Multiplikation von Funktionen.