Beweisen durch Analogie

Erica Melis*

Universität des Saarlandes, Fachbereich Informatik,

D-66041 Saarbrücken, Germany,

email: melis@cs.uni-sb.de

Zusammenfassung

Der Artikel stellt einen neuen Ansatz für das automatische Beweisen durch Analogie vor und liefert Begründungen für diesen Ansatz. Einige Prinzipien des Beweisens durch Analogie werden aus einer formalen Analyse der in einem Lehrbuch vorkommenden Analogienutzung und aus anekdotischen Evidenzen extrahiert. Diese Prinzipien sind dann in Implementationen der analogie-geleiteten Beweisplankonstruktion in zwei sehr verschiedenen Beweisplanern umgesetzt. Die Erfahrungen mit den Implementationen werden herangezogen, um die Frage zu beantworten, unter welchen Bedingungen automatische Analogienutzung im Theorembeweisen sinnvoll angewendet werden kann.

Theorem Proving by Analogy

abstract

*Die Forschung, über die hier berichtet wird, wurde unterstützt durch die Deutsche Forschungsgemeinschaft, SFB 314 und SFB 378.

This article describes a new approach for automated theorem proving by analogy, the analogy-driven proof plan construction. It provides a cognitive motivation for some features of this approach. The cognitive modelling is based on a formal analysis of analogies in a textbook and on anecdotical evidence. The implementation of automated analogy formation takes into consideration the cognitive modelling. Furthermore, our experience with the implementation of the analogy strategy in two different proof planning systems is used to analyze conditions under which theorem proving by analogy is sensible.

1 Einführung

Die Nutzung von Analogien ist eine Problemlösestrategie, die durch folgendes Prinzip gekennzeichnet werden kann:

Löse ein Problem (Target), indem Du Dich an die Lösung eines ähnlichen Problems (Source) erinnerst und die alte Lösung an das neue Problem anpaßt.

Die Beschäftigung mit Analogien hat eine lange Tradition in der kognitiven Psychologie, siehe z.B. [Sternberg,1977; Gentner,1983; Holyoak & Thagard,1989] und inzwischen auch eine langjährige Geschichte in der Künstlichen Intelligenz (KI) (siehe beispielsweise [Hall,1989] für eine Übersicht bis 1989). In den letzten zehn Jahren hat sich insbesondere das fallbasierte Schließen (CBR) als ein Gebiet der KI mit eigenen Konferenzen und kommerziellen Anwendungen [Althoff et al.,1995] etabliert.

Es hat sich als sinnvoll erwiesen, das oben genannte Prinzip speziell in Domänen, in denen es keine wohldefinierten Regeln für das Problemlösen gibt (ill-defined domains) anzuwenden. Dies ist auch die Standardbegründung für den Einsatz fallbasierter Systeme [Kolodner,1993]. In solchen schlecht definierten Domänen ist fallbasiertes Schließen vor allem für sogenannte klassifizierende Problemlösungen erfolgreich, also z.B. für Diagnose, Vorhersage, Assessment und Klassifikation, weil für diese meist keine Adaptation der übertragenen Lösungen erforderlich ist. Fallbasiertes Problemlösen mit dem Ziel, Pläne, Beweise oder Designs zu synthetisieren, erwies sich als erheblich komplizierter, weil erstens häufig Adaptation notwendig wird, weil die Problemlösungen zweitens oft eine komplizierte Struktur aufweisen und weil drittens häufig schwierig zu begründen ist, warum eine analoge Übertragung gerechtfertigt ist (vgl. auch [Bergmann & Wilke,1996; Börner et al.,1996]).

Im Gegensatz zu typischen schlecht definierten Domänen scheint Mathematik und das Beweisen mathematischer Theoreme wohldefiniert zu sein. Für diesen Bereich gibt es vollständige und formale Beweiskalküle (z.B. Kalkül des Natürlichen Schließens (ND) von Gentzen [Gentzen,1935]) mit formalen Inferenzregeln, und auch Axiome und Theoreme sind formal repräsentiert.

Trotzdem überzeugt ein kurzer Blick in die einschlägige mathematische Literatur (siehe z.B. [Polya,1945; Deussen,1971]), daß die Nutzung von Analogien, wie sie in Abbildung 1 verdeutlicht wird, eine von Menschen häufig benutzte Beweisstrategie ist. Lehrbücher, wie etwa "Halbgruppen und Automaten" [Deussen,1971], überlassen dem Leser das Führen von Beweisen, die analog zu vorangegangenen Beweisen sind: *"Die Übertragung der Sätze 5.6.,5.7., 5.8. auf den vorliegenden Fall bereitet nun keine Schwierigkeiten und sei dem Leser überlassen."* oder auch *"Wir haben ...die Implikation ... gezeigt; die Umkehrung ist analog zu beweisen."*.

HIER ABBILDUNG 1

Wieso kann es auch in anscheinend wohldefinierten Domänen wie dem mathematischen Beweisen sinnvoll sein, Analogien zu nutzen?
Erstens, spannen mathematische Beweisprobleme einen potentiell riesigen Suchraum auf, insbesondere bei langen mathematischen Beweisen oder bei solchen Beweisen, die viele

Axiome und Lemmata verwenden. Die Unbeherrschbarkeit dieses Suchraums stellt das Schlüsselproblem des traditionellen automatischen Beweisens dar, bei dem versucht wird, dieses Problems mit lokalen Heuristiken zu bewältigen. Die Erfolge des klassischen automatischen Beweisens blieben wegen der sehr eingeschränkten Möglichkeiten, die Suche zu steuern begrenzt, obwohl trickreiche lokale Heuristiken (Heuristiken, die Informationen über den aktuellen Beweiszustand benutzen) und zunehmend leistungsfähige Computer eingesetzt werden. Im Gegensatz zu den klassischen Beweistechniken liefert die Analogienutzung, sofern sie möglich ist, eine globale Steuerung der Beweissuche dadurch, daß die Suche nach einem geeigneten Inferenzschritt, die eine Alternative aus allen anwendbaren Inferenzen auswählt, durch das Vorschlagen eines Beweisschrittes ersetzt wird.

Zweitens, sind mathematische Probleme und Kalküle auf der Objektebene (Kalkülebene) zwar wohldefiniert, aber für die (Meta-)Ebene der Steuerung mathematischer Beweise sind nur wenige Regeln bekannt bzw. explizit formuliert [Schoenfeld,1985]. Aufgrund des fehlenden Steuerungswissens können die riesige Suchräume nicht effizient traversiert werden, und dadurch ist es häufig nicht möglich, überhaupt einen Beweis zu finden. Das gilt sowohl für menschliche Beweiser, die keine Beweis-Experten sind, als auch für automatische Beweiser. Die Nutzung von Analogien kann in manchem konkreten Target-Beweis das nicht vorhandene Kontrollwissen ersetzen, indem das in einem ähnlichen Source-Beweisprozeß mühsam (z.B. durch Versuch und Irrtum) akquirierte Steuerungswissen benutzt wird.

In der Künstlichen Intelligenz werden seit etwa 20 Jahren mit gewissem Erfolg Systeme entworfen [Kling,1971; Munyer,1981; Owen,1990; Kolbe & Walther,1994], die das Beweisen durch Analogie simulieren sollen. Beispielsweise konnte Owen einen Beweis von $x \cup x = x$ analog zu einem Beweis für $x \cap x = x$ erzeugen [Owen,1990]. Diese Systeme sind traditionell unabhängig von einem Theorembeweiser, der die Source-Beweise liefert und übertragen die Beweisschritte auf Kalkülebene und zwar auf der Grundlage von Symbolabbildungen, die in neueren Ansätzen [de la Tour & Caferra,1987; Kolbe & Walther,1994; Curien,1995] durch second-order Matching[1] gefunden werden.

1.1 Motivation für einen neuen Ansatz

Die folgende Anekdote legt nahe, daß im Gegensatz zu den bisherigen Ansätzen zum automatischen Beweisen durch Analogie, die analoge Übertragung in einen größeren Problemlösezusammenhang eingebettet sein und die analoge Übertragung nicht auf der Kalkülebene erfolgen sollte:

Anfang der 80er Jahre gelang es dem deutschen Mathematiker Gert Faltings, die 1922 aufgestellte Mordellsche Vermutung zu beweisen, eine enge Verwandte des 'Großen Fermatschen Satzes'. (Mordells Vermutung lautet: Algebraische Kurven vom Geschlecht zwei und mehr haben höchstens endlich viele rationale Punkte.) Die Zeitschrift 'bild der wissenschaft' druckte ein Interview mit Faltings ab, in dem er nachträglich über Strategien berichtete, die ihn zum Beweis der Vermutung führten [Faltings & Deker,1983]. Die folgenden Zitate sind diesem Interview entnommen und charakterisieren Problemlösestrategien, die nicht nur in diesem konkreten Entdeckungszusammenhang eine wichtige Rolle spielten, sondern unter Mathematikern weithin bekannt sind, siehe zum Beispiel [Polya,1945; Polya,1964; Schoenfeld,1985]. Faltings berichtet:

- *"Man hat Erfahrungen, daß bestimmte Schlüsse unter bestimmten Voraussetzungen funktionieren. Als erstes überlegt man sich daher, wie der Weg aussehen könnte. Man überlegt sich also im Groben: Wenn ich das habe, könnte ich das zeigen und*

[1] Second-order Matching erlaubt, Funktionsvariable mit Funktionstermen zu matchen, z.B. $f \mapsto \lambda x \lambda y . x^2 + y$.

dann das nächste. Hinterher muß man die Details einfügen und sieht, ob man es auch wirklich so machen kann."

- *"Ich muß sagen, daß ein wesentlicher Teil des Beweises im Prinzip schon da war, den ich nur entsprechend übertragen habe. Einen ganz neuen Trick habe ich nicht gefunden."*
- *"Es kommt aber auch durchaus vor, daß man mal da sitzt, nicht mehr weiter weiß und dann probiert, wohin der Weg führt."*

Das erste Faltings-Zitat beschreibt das Problemlösen durch eine Technik, die in der KI als *Planen* bezeichnet wird. *"Man hat Erfahrungen, daß bestimmte Schlüsse unter bestimmten Voraussetzungen funktionieren"* beschreibt Steuerungswissen für bestimmte (große) Beweisschritte, sogenannte Planungsoperatoren. *"Man überlegt sich also im Groben: Wenn ich das habe, könnte ich das zeigen und dann das nächste"* beschreibt den eigentlichen Planungsvorgang, und *"Hinterher muß man die Details einfügen und sieht, ob man es auch wirklich so machen kann"* weist auf ein hierarchisches Vorgehen hin, bei dem auf einer höheren Ebene geplant und später auf einer niedrigeren Ebene Details eingefügt werden. Erst die Ausführung des Plans ergibt einen Beweis auf der Objektebene, der nachprüfbar ist.

Mit ganz ähnlicher Diktion rät Polya [Polya,1945] seinen Lesern, beim Lösen mathematischer Probleme folgendermaßen vorzugehen:

1. Verstehe das Problem.
2. Mache einen Plan (an dieser Stelle kommt auch Analogie ins Spiel).
3. Führe den Plan aus.
4. Werte die Problemlösung aus.

Über ähnliche Beobachtungen zum Beweis-Planen im geometrischen Beweisen berichten

Anderson und Koedinger. Ihre Ergebnisse wurden durch Protokollanalyse gefunden und besagen, daß der Prozeß der Beweisfindung aus zwei Phasen besteht: in der ersten Phase wird ein Beweisplan gefunden, und in der zweiten wird der Plan in einen Beweis übersetzt [Anderson,1984; Koedinger & Anderson,1990].

Das zweite Faltings-Zitat stützt die Hypothese, daß (Teile von) Beweisen von Experten tatsächlich analog übertragen werden und daß diese Strategie zu erfolgreichen Problemlösungen beiträgt. Der Gesamtzusammenhang der Zitate legt nahe, daß die Analogienutzung auch ins Beweisplanen eingebunden ist. Äußerungen anderer Mathematiker, obwohl generell spärlich, bestätigen diese Vermutung ebenfalls. Beispielsweise fordert Polya den Leser in seinem Buch "How to Solve It" [Polya,1945] auf, sich bei der Bearbeitung eines Problems folgende Fragen vorzulegen:

- Hast Du das Problem schon einmal gesehen?
- Kennst Du ein ähnliches Problem?
- Hast Du das Problem schon einmal in einer anderen Form gesehen?

Ausgehend von solchen im weitesten Sinne "empirischen" Evidenzen für bestimmte Merkmale von Analogienutzung beim Beweisen, besteht das Hauptanliegen diese Artikels nun darin, einen neuen Ansatz für das automatische Beweisen durch Analogie vorzustellen und Begründungen für diesen Ansatz zu liefern. Zu diesem Zweck wird in Abschnitt 2.1 anhand eines Lehrbuches analysiert, welche Arten von Analogienutzug beim Beweisen tatsächlich vorkommen. Diese Analyse mündet in bestimmten Designprinzipien für den automatischen Analogieprozeß. Darüber hinaus werden Prinzipien der zugrundeliegenden Repräsentation der Source-Beweise aus retrospektiven und pädagogisch orientierten Beschreibungen von Mathematikern extrahiert.

Die Erkenntnis, daß die Nutzung von Analogien sinnvoll auf der Ebene von Beweisplänen anzusiedeln ist, wie sie in Abschnitt 2.2 diskutiert wird, deckt sich mit einschlägigen

Erfahrungen in interaktiven Beweissystemen, daß Beweise auf Kalkülebene zu detailliert sind, um sie sinnvoll analog übertragen zu können[2].

Die Extraktion der genannten Prinzipien führte folgerichtig zur Entwicklung eines neuen Ansatzes, der analogie-geleiteten Beweisplankonstruktion [Melis,1995b], die in Abschnitt 3 vorgestellt wird und die bereits erfolgreich auch auf einige schwierige mathematische Probleme angewendet wurde [Melis,1995c]. Natürlich ist unser Ziel vorläufig nicht, den Beweis der Mordellschen Vermutung auf den Rechner zu bringen. Genuine mathematische Probleme werden aber bearbeitet. Erfahrungen mit der Implementierung der analogie-geleiteten Beweisplankonstruktion in zwei sehr verschiedenen Beweisplanern werden in Abschnitt 3.3 berichtet, um die Frage zu beantworten, unter welchen Bedingungen Analogien beimTheorembeweisen sinnvoll eingesetzt werden können.

2 Prinzipien des analogiebasierten Beweisens

Über Anektdoten hinaus wurden Evidenzen zu der Frage gesammelt, was Mathematiker unter Analogien verstehen und welche Repräsentation von Source-Beweisen in diesem Zusammenhang in Frage kommen. Die umfangreichsten Untersuchungen [Melis,1993] wurden am Lehrbuch "Halbgruppen und Automaten" unternommen. Dabei wurden die in diesem Buch vorkommenden Analogien und die Techniken die benötigt werden, um diese Analogien automatisch zu finden und zu nutzen, analysiert. Eine weitere Analyse von analogen Beweisen, die in einer mathematischen Zeitschrift als solche genannt wurden, gibt zusätzlichen Aufschluß darüber, welche Lösungsbestandteile analog übertragen werden und welche Anforderungen die Source-Repräsentation erfüllen sollte.

[2]Persönliche Kommunikation mit Bob Harper über entsprechende Versuche mit dem System Nuprl

2.1 Reformulierungsmechanismen in der Analogiebildung

"Halbgruppen und Automaten" [Deussen,1971] wird als Lehrbuch für Automatentheorie im Informatik-Vordiplom verwendet. Wir haben die darin vorgeschlagenen Analogiebildungen nachvollzogen und untersucht, welche Mechanismen für sie benötigt werden. Das Lehrbuch "Halbgruppen und Automaten" schien uns geeignet für die Untersuchung analoger Beweise, weil relativ viele analoge Beweise explizit erwähnt werden, sowohl solche mit sehr einfachen Übertragungen als auch komplizierte, die für uns natürlich besonders interessant waren. Das Buch besteht aus drei Kapiteln, zwischen denen analoge Übertragungen von Theoremen und deren Beweisen ausgeführt werden:

- Halbgruppen und Relationen,
- Halbgruppen und Semimoduln,
- Automaten.

So heißt es beispielsweise in Abschnitt 7 über Homomorphismen von Semimoduln (zweites Kapitel) *"Die Übertragung der Sätze 5.6.,5.7., 5.8.* (aus Abschnitt 5 des ersten Kapitels) *auf den vorliegenden Fall bereitet nun keine Schwierigkeiten..."*. Auch innerhalb einzelner Kapitel werden analoge Übertragungen explizit erwähnt.

Eine ganze Reihe der in diesem Buch benutzten Analogien sind nicht mit den herkömmlichen Methoden der Analogiebildung nachvollziehbar, weil diese Methoden auf Symbolabbildungen beruhen. (Unter Symbolabbildung verstehen wir die Abbildungen von Aussagen, die durch die Abbildung von Individuen-, Relations- und Funktions*symbolen* auf Individuen-, Relations- und Funktions*symbole* zustande kommen. Diese sind nicht allgemein genug, um auch allgemeinere Termabbildungen (einschließlich Funktionsterme) zuzulassen.) Insbesondere erwiesen sich für die folgenden drei Typen von Theoremen die bisherigen Systeme für das Beweisen durch Analogie als unbrauchbar: (Detailliertere Angaben entnehme der interessierte Leser bitte dem Bericht [Melis,1993]. Sie würden den Rahmen dieses Artikels sprengen.):

1. Für einige analoge Übertragungen, unter anderem zwischen den Beweisen von Theorem (7)5.7 und von Theorem 5.7[3], muß zunächst eine **abstrahierende Reformulierung** gefunden werden, die das Source-Problem und das Target-Problem jeweils in ein abstrakteres Problem überführt. Diese Reformulierung verändert nicht nur die Probleme sondern auch den Source-Beweis. Die Abstraktionen sind nicht a priori gegeben, sondern müssen für die Analogiebildung konstruiert werden. Beispiel:

   Source-Theorem (7)5.7:[4]
   Seien S, T_1, T_2 F-Semimoduln und $\phi_1 : S \mapsto T_1, \phi_2 : S \mapsto T_2$ zwei Homomorphismen in die F-Semimoduln T_1 bzw. T_2. Dann ist die Abbildung Φ mit $\Phi\phi_1 = \phi_2$ ein Homomorphismus in Semimoduln.

   Target-Theorem 5.7:[5]
   Seien S', H_1, H_2 Halbgruppen und $\phi_1 : S' \mapsto H_1, \phi_2 : S' \mapsto H_2$ zwei Homomorphismen in die Halbgruppen H_1 bzw. H_2. Dann ist die Abbildung Φ mit $\Phi\phi_1 = \phi_2$ ein Homomorphismus in Halbgruppen.

   Für die analoge Übertragung des Beweises von 7.5.7 auf einen für 5.7 muß man die Definition von **Homomorphismen in F-Semimoduln**:

   $$\forall f, s(f \in F \wedge s \in S \rightarrow \phi(f \cdot s) = (f \cdot \phi(s)) \tag{1}$$

   auf die Definition von **Homomorphismen in Halbgruppen**:

   $$\forall f, g(f, g \in F \rightarrow \phi(f \cdot g) = \phi(f) \cdot \phi(g) \tag{2}$$

[3]Die im folgenden benutzte Numerierung folgt der Originalnumerierung in [Deussen,1971]. In Fällen, in denen das Target-Theorem keine eigene Nummer hat, erhält das Target-Theorem die Nummer des Source-Theorems erweitert durch die Nummer des Abschnitts, zu dem das Target gehört.

[4]2.Teil geringfügig vereinfacht

[5]2.Teil, geringfügig vereinfacht

übertragen. Das erfordert zunächst die Abstraktion von (1) zu

$$\forall x(\phi(\mathcal{F}(x)) = \mathcal{F}(\phi(x))), \tag{3}$$

wobei $\mathcal{F}$ eine Variable für eine Funktion ist, bzw. von (2) zu

$$\forall x, y(\phi(\mathcal{F}'(x, y)) = \mathcal{F}'(\phi(x), \phi(y))) \tag{4}$$

für eine Funktionsvariable $\mathcal{F}'$ zu finden. (3) und (4) beschreiben die abstrakte Idee eines Homomorphismus für einstellige bzw. zweistellige Funktionen.

2. Für eine Reihe von Analogien in [Deussen,1971] werden noch **andere Reformulierungen** benötigt, um überhaupt eine Abbildung von Source- und Target-Theoremen und von Axiomen, Definitionen und Lemmata in Source und Target zu finden. Diese Reformulierungen gehen über reine Symbolabbildungen hinaus. Rekapitulieren wir beispielsweise die Übertragung der Definition (1) auf die Definition (2): Nachdem die Abstraktionen zu (3) und (4) erfolgt sind, erkennt man, daß nun noch eine einstellige Funktion $\mathcal{F}$ auf eine zweistellige Funktion $\mathcal{F}'$ abgebildet werden muß. So eine Änderung in Theoremen oder Definitionen zieht auch bestimmte (berechenbare) Änderungen in den jeweiligen Beweisen/Beweisplänen nach sich. So muß im Fall der Reformulierung von (3) zu (4) ein Teilbeweis verdoppelt werden.

3. Sehr häufig übertragen Beweise durch Analogie nur einen **Teilbeweis** des Source-Beweises (wie das auch in dem Interview mit Faltings zum Ausdruck kam), während manche Teile des Target-Beweises völlig neu bewiesen werden müssen. Das gilt unter anderem für die Übertragung eines Teilbeweises von Theorem 4.8 auf einen Teilbeweis von Theorem 5.3.

Diskussion

Ein Lehrbuch gibt normalerweise nicht wieder, wie und daß der Autor die jeweiligen Beweise tatsächlich durch Analogie gefunden hat. Vielmehr hält er Analogie nachträglich für eine gute Lösungsstrategie. Allerdings zählen die vorgeschlagenen Beweise durch Analogie zu den analogen Übertragungen, die ein Leser (bzw. ein automatischer System) fähig sein sollte durchzuführen. Insofern ist das untersuchte Material durchaus aussagekräftig für die Gestaltung eines interaktiven mathematischen Assistenzsystems wie OMEGA [Huang et al.,1994] oder eines automatischen Systems, etwa *CLAM* [Bundy et al.,1991].

Die drei Anforderungen an die Analogiebildung, die aus Deussens Lehrbuch extrahiert wurden, gehen über das hinaus, was in bisherigen automatischen Analogiesystemen mit Symbolabbildungen möglich war. Die Analyse des Lehrbuchs hat gezeigt, daß diese bisherigen Techniken nicht ausreichen, um Analogienutzungen, wie sie tatsächlich von Mathematikern vorgenommen werden, zu simulieren. Die Untersuchungen legen nahe, zusätzlich zu den bisher üblichen Symbolabbildungen auch kompliziertere Reformulierungen (einschließlich abstrahierender Reformulierungen) in Analogieprozeduren zuzulassen.

Solche Reformulierungen begegneten uns nicht nur in den Beispielen aus Deussens Buch, sondern auch in diversen anderen Beispielen für Analogienutzung, z.B. in analogen Übertragungen von Sätzen eindimensionale Räume auf ähnliche Sätze über zweidimensionale Räume [Melis,1995c].

2.2 Strukturierte Beweise und Analogie

In diesem Abschnitt benutzen wir Ergebnisse und Beispiele, die Uri Leron in seinem Artikel "Structuring Mathematical Proofs" in der Zeitschrift *The American Mathematical Monthly* 1983 darstellt und analysieren sie in Hinblick auf das Beweisen durch Analogie.

Lerons Artikel befaßt sich in erster Linie mit der Frage, wie mathematische Beweise präsentiert werden sollten, damit Studenten sie verstehen, nachvollziehen und andere ähnliche Beweise entdecken können. Leron berichtet von Erfahrungen, die er in Mathematik-Lehrveranstaltungen gesammelt hat[6]. Leron vergleicht die Wirkungen von sequentiell dargebotenen Beweisen mit denen von hierarchisch dargebotenen. Eines seiner Beispiele ist der Beweis des folgenden Theorems TRIADIC. Als ausschlaggebende Faktoren für alle drei Ziele sieht er die hierarchische Strukturierung von Beweisrepräsentationen an.

TRIADIC: *Es gibt unendlich viele triadische Primzahlen, d.h. Primzahlen die darstellbar sind als 4k+3, wobei k eine natürliche Zahl ist.*

Ein sequentielle Beweis von TRIADIC könnte folgendermaßen aussehen:
"Betrachte das Produkt zweier monadischer Zahlen[7]:

$$(4k+1)(4l+1) = 4k4l + 4k + 4l + l = 4(4kl + k + l) + 1$$

Dieses ist wieder monadisch. Angenommen, das Theorem sei falsch, d.h. es gäbe nur endlich viele triadische Primzahlen $p_1, p_2, \dots p_n$ $(p_1 = 3)$. Sei $M = 4 \cdot p_2 \cdot \dots \cdot p_n + 3$. Keins der p_i mit $i > 1$ teilt M, weil sonst auch gälte $p_i | 3$[8]. Auch 3 teilt M nicht, weil 3 weder 4 noch die Primzahlen $p_2, \dots p_n$ teilt. Da außerdem alle Primzahlen entweder monadisch oder triadisch sind, sind alle Primfaktoren von M monadisch und somit ist M selbst monadisch. Wegen der Darstellung $M = 4p_2 \dots p_n + 3$ ist M aber triadisch, was zu einem Widerspruch führt, weil keine Zahl gleichzeitig monadisch und triadisch sein kann.
Leron beschreibt die allgemeine Konstruktion von hierarchischen Beweisen wie folgt:
Beginne mit der Beweisidee, d.h. mit den groben Teilzielen.

[6]Lerons Artikel wurde sehr gut aufgenommen, wohl weil er einen sehr wichtigen, viele Professoren bewegenden Punkt angesprochen hatte.

[7]monadische Zahlen sind darstellbar als (4k+1).

[8]$p_i|3$ bedeutet p_i teilt 3.

Die nächst niedrigere Ebene liefert (partielle) Beweise für noch nicht bewiesene Behauptungen der vorangegangenen Ebene, so wie die Konstruktion eines Objekts, dessen Existenz vorher nur behauptet wurde.
Wenn ein Teilbeweis selbst wieder kompliziert ist, verlege seine Details in eine tiefere Ebene.

Ein konkreter hierarchisch strukturierter Beweis von TRIADIC ist der folgende, der auch schematisch in Abbildung 2 gezeigt wird.

HIER ABBILDUNG 2

Level 1: Angenommen, das Theorem sei falsch und $p_1 \dots p_n$ (für $p_1 = 3$) seien alle triadischen Primzahlen. Wir konstruieren (in Level 2) eine Zahl M, die die folgenden zwei Eigenschaften hat:

(a) M und alle ihre Faktoren sind verschieden von $p_1 \dots p_n$.

(b) M hat einen triadischen Primfaktor.

Weil (a) und (b) einander widersprechen, ist die Annahme falsch.

Level 2: Sei $M = 4 \cdot p_2 \cdot \ldots \cdot p_n + 3$. Wir zeigen, daß dieses M die Eigenschaften (a) und (b) hat.

(a) $p_2 \dots p_n$ teilen M nicht, weil 3 nicht durch $p_2 \dots p_n$ teilbar ist; 3 teilt M nicht, weil $4 \cdot p_2 \cdot \ldots \cdot p_n$ nicht durch 3 teilbar ist und $M \neq p_i$ wegen $M > p_i$.

(b) Angenommen, alle Primfaktoren von M wären monadisch. Dann wäre M als Produkt monadischer Zahlen wieder monadisch. Das ergäbe einen Widerspruch zur Konstruktion von M.

Level 3: Beweis des Lemmas:

Das Produkt monadischer Zahlen ist wieder monadisch.

Für Beweise durch Analogie schauen wir uns das von Leron analysierte Source-Theorem TRIADIC und das folgende Target-Theorem MONADIC an:

MONADIC: *Es gibt unendlich viele monadische Primzahlen, d.h. Primzahlen der Form 4k+1.*[9]

Wenn man das Theorem MONADIC per Analogie zum Beweis von TRIADIC beweisen will, ist die sequentielle Darstellung, in der die Beweisidee nur versteckt vorkommt, nicht sehr hilfreich, zumal wenn man der Reihenfolge im sequentiellen Source-Beweis folgt. Es zeigt sich nämlich, daß die Beweise von TRIADIC und MONADIC nur auf der obersten hierarchischen Ebene übertragbar sind, was bei einer sequentiellen Beweispräsentation gar nicht auffällt. Der Beweis von Level 3 kann nicht übertragen werden, weil das Produkt triadischer Zahlen nicht triadisch sein muß.

Dahingegen gibt es andere Source- und Target-Theoreme, bei denen auch die Detailbeweise übertragbar sind. Als Beispiel dafür gibt Leron ein Source-Theorem über Vektorräume mit Hermitischen Operatoren und ein ähnliches Target-Theorem über Vektorräume mit normalen Operatoren an, für die eine Analogie bis zur Ebene n=3 besteht. (Wir belassen es bei dieser Andeutung, weil eine genaue Erklärung zu umfangreich wäre. Die Details finden sich in [Leron,1983].)

Diskussion

Ebenso wie ein Lehrbuch haben Lerons Folgerungen den Charakter einer Empfehlung und nicht der Beschreibung oder Untersuchung eines kognitiven Prozesses. Die in Empfehlungen aggregierte Erfahrung liefert aber eine gewisse Evidenz dafür, daß eine für Analogiebildung geeignete Repräsentation von Source-Beweisen strukturiert sein muß. Dadurch können

[9] Alle Primzahlen sind entweder monadisch oder triadisch.

dann Teile von Beweisen – hier vor allem Beweisideen – zuerst übertragen werden und danach eventuell auch noch Beweisdetails. Ganz ähnliche Erfahrungen haben wir bei der analogen Übertragung von Beweisplänen anderer Theoreme gesammelt. So waren beispielsweise für Heine-Borel Theoreme fast alle Details übertragbar [Melis,1995c], wohingegen für bestimmten Diagonalisierungsbeweisen nur die Idee übertragbar war (z.B. vom Beweis des Cantorschen Satzes auf den Beweis des Halting Problems) [Melis,1994].

Die Forderung, strukturierte Beweisrepräsentationen analog zu übertragen, widerspricht den bisherigen Analogieansätzen, die sequentiell Beweisschritte auf der Kalkülebene übertragen (z.B. Resolution und Paramodulation in [Owen,1990], Gleichheitsinferenzen in [Kolbe & Walther,1994]). Diese Ansätze sind in Anlehnung an die herkömmlichen automatischen Beweissysteme entstanden, die ihre Beweise auf der Ebene der vollständigen und formalen logischen Kalküle, d.h. der kleinsten/detailliertesten Beweisschritte erstellen.

Eine sequentielle Darstellung eines formalen Beweises kann man insofern als eine "oberflächliche" Repräsentation eines Beweises verstehen, als das Wesentliche des Beweises (die Beweisidee) relativ versteckt sein kann. Einen hierarchisch strukturierter Beweis dagegen, kann man als eine "tiefere" Repräsentation ansehen weil er Informationen über besonders relevante Beweisteile (Ideen) und Informationen über "chunks" von zusammengehörigen Schritten enthält. Ein Student, der nur über einen sequentiellen Source-Beweis verfügt, kann nur die oberflächlichen Ähnlichkeiten erkennen und mechanisch die Folge der Beweisschritte übertragen, was zu einem schnellen, vorzeitigen Fehlschlagen der Analogienutzung führen kann.

Zusammenfassung

Fassen wir die wichtigsten Schlußfolgerungen aus dem Vorangegangenen kurz zusammen:

- Für den analogen Transfer von Beweisen ist besonders eine strukturierte

- Häufig werden für analoge Übertragungen Reformulierungen des Source-Problems und Source-Beweises benötigt.

Der im folgenden vorgestellte Ansatz der analogie-geleiteten Beweisplankonstruktion trägt diesen zwei Hauptfolgerungen nun Rechnung, indem er auf der Beweisplanebene und mit Reformulierungen arbeitet.

3 Umsetzung der Designprinzipien: Analogie-geleitete Beweisplankonstruktion

Das Ziel unserer Implementation des Beweisens durch Analogie war es zunächst, Analogie in das System OMEGA, das ein interaktives System mit starken automatischen Komponenten ist, als eine Steuerungsstrategie zu integrieren. Der Hauptzweck der Analogienutzung ist hier, einerseits eine dem Benutzer geläufige Problemlösestrategie zur Anwendung bereitzustellen, und andererseits Benutzerinteraktionen einzusparen. Ein Vergleich der Performanz von OMEGA mit bzw. ohne Analogie kann nur in einer Abwägung zwischen Benutzerinteraktion und automatischer Analogiebildung bestehen, die aber jeweils unterschiedliche Ressourcen verbrauchen. Mehr hierzu im Abschnitt 3.3.

Später wurde dann auch eine Variante der analogie-geleiteten Beweisplankonstruktion in den rein automatischen Beweisplaner *CLAM* integriert mit dem Ziel, mehr Beweise finden zu können. In diesem Falle ist ein Vergleich nur über die Tatsache gegeben, daß einige Theoreme nur mit dem zusätzlichem Aufwand für die Analogiebildung bewiesen werden konnten. Mehr hierzu wiederum im Abschnitt 3.3.

Das automatische Auffinden der Source-Probleme, die analog zu einem gegebenen Target-Problem sind, war zunächst nicht Gegenstand unserer Arbeit, unter anderem, weil dies für beliebige mathematische Probleme und Beweise ein schwieriges Problem ist. In den bisherigen zwei Implementierungen der analogie-geleiteten Beweisplankonstruktion ist das Wiederauffinden von Source-Problemen unterschiedlich gelöst: Die Analogiekomponente in OMEGA erhält das Source-Problem durch den Benutzer. In *CL^AM* kann das Auffinden des Source-Theorems aufgrund beweis-relevanter Abstraktionen erfolgen, worauf wir hier aus Platzgründen allerdings nicht näher eingehen.

Beweispläne

Beweispläne, eingeführt von Bundy [Bundy,1988], bestehen aus Methoden, die auf Sequenten angewendet werden und neue Sequenten erzeugen[10], siehe Abbildung 6. Die Methoden ihrerseits sind Spezifikationen von Taktiken [Gordon et al.,1979] mit Hilfe von Pre- und Postconditions. Die Spezifikationen machen es möglich, mit Methoden zu planen. Taktiken sind Prozeduren, die Folgen von logischen Inferenzschritten erzeugen wenn sie ausgeführt werden. So eine Zusammenfassung mehrere Inferenzschritte bedeutet eine Abstraktion des Beweises. Beweispläne sind eine strukturierte Beweisrepräsentation, weil auch Beweisideen bzw. Teilbeweise als Methoden implementiert werden können. Beispielweise kann man in einer Induktions-Methode festlegen, daß nach der Berechnung des Induktionsaxioms zuerst der Induktionsanfang bewiesen werden soll und dann der Induktionsschritt.

Als Beispiel ist die Methode `Indirekt` für einen Widerspruchsbeweis, wie er beispielsweise im Beweisplan von TRIADIC vorkommt, in Abbildung 3. dargestellt und zwar im Methodenformat von OMEGA.

HIER Abbildung 3

Die Pre- und Postconditions sind die Inputs und Outputs der Methode. Die Constraints sind Bedingungen, vor allem Instantiierungsbedingungen, unter denen die Methode angewendet

[10]Ein Sequent ist ein Paar $\Delta \vdash F$ aus einer Menge von Aussagen Δ und einer Aussage F, das ausdrückt, daß F aus Δ folgt.

werden darf. Das Beweisschema (proof schema) besteht aus Beweiszeilen, die ganz rechts Begründungen für die Formel in der Zeile enthalten. Begründungen sind ND-Regeln (wie etwa $\neg I, \neg E$), LEMMA falls die Zeile zu den Preconditions gehört, oder OPEN falls diese Zeile später bewiesen werden kann. Obwohl in `Indirekt` nicht zu sehen, ist auch der Aufruf eines Beweissystems (z.B. OTTER [McCune,1990]) als Begründung erlaubt, falls dieses System die Formel beweisen kann. Der procedure-Slot enthält entweder einen Standardinterpreter für das Beweisschema oder eine Prozedur, die Resultate der Methode erzeugt.

Ein formaler Beweis entsteht dadurch, daß zunächst ein Beweisplan gefunden wird, der aus instantiierten Methoden besteht. Dann wird der Plan ausgeführt, indem die Prozeduren, die zu den instantiierten Methoden[11] gehören ablaufen. Die einfachste Form des Beweisplanens[12] beginnt mit dem Theorem als *offenes* Ziel g. Dann sucht der Planer nach einer anwendbaren Methode M mit Postcondition g. Diese Methode wird in den Beweisplan eingefügt, und in der Menge offener Ziele wird g durch die Preconditions $g_1 \dots g_n$ von M ersetzt, siehe Abb. 4. Dann heißt g erfüllt. Der Planer setzt die Suche nach Methoden für die neuen offenen Ziele fort, bis kein offenes Teilziel mehr existiert. Planen, das die Strategie des Vorwärtsplanen (mit f(orward)-Methoden) benutzt, sucht in der entgegengesetzten Richtung: Ausgehend von den Beweisannahmen $g_1 \dots g_n$ erzeugt Vorwärtsplanen eine (oder mehrere) neue Annahme g bis eine der Annahmen mit dem Theorem übereinstimmt. Eine einfache Strategie für hierarchisches Planen unterdrückt gewisse Preconditons einer Methode M und verlagert deren Erfüllung in eine tiefere Planungsebene. Dadurch wird in der höheren Planungsebene nur ein partieller Plan erzeugt und Details später geplant.

[11]In instantiierten Methoden sind die Parameter durch Konstanten ersetzt, z.B. ist F in `Indirekt` durch eine konkrete Formel ersetzt.

[12]die die Strategie des Rückwärtsplanen mit b(ackward)-Methoden benutzt

HIER ABBILDUNG 5

Abbildung 5 zeigt als Beispiel den Beweisplan für das Theorem $length(append(a,b)) = length(append(b,a))$[13], der vom Beweisplaner CL^AM erzeugt wurde. das heißt für die Kommutativität von *append* bezüglich *length*. Der Beweisplan enthält instantiierte Induktionsmethoden INDUCTION sowie ELEMENTARY und anderen Methoden, die als Rechtecke dargestellt sind. Zuerst wird INDUCTION über die Variable a angewendet. Dies führt zu zwei neuen Teilzielen, für den Induktionsanfang und den Induktionsschluß. WAVE ist eine Methode, die die Induktionsbehauptung gezielt umformt, so daß FERTILIZE dann die Induktionsvoraussetzung auf das umgeformte Teilziel anwenden kann.

Reformulierungsmechanismen

Reformulierungsprozeduren in der Analogienutzung dienen zunächst dazu, das Source-Problem und das Target-Problem in Übereinstimmung zu bringen. Sie haben zusätzlich den Effekt, den Source-Beweisplan zu verändern, indem z.B. neue Methoden eingefügt, andere Methoden ersetzt oder Methoden verändert werden, also einen Teil dessen zu erledigen, was in den meisten Analogiesystemen erst durch die Adaptation einer Zwischenlösung geschieht. Die Adaptation geschieht also in unseren Systemen in zwei Schritten:

- Reformulierungen sind Prozeduren die **vor** der analogen Übertragung die Änderungen des Source-Beweisplans bewerkstelligen, die aus den Unterschieden des Source- und des Target-Problems vorhersehbar sind.

- Modifikationen, so wie etwa das Neuplanen von Teilen des Target-Beweises, erfolgen **nach** der analogen Übertragung des reformulierten Source-Plans, falls es nötig ist.

[13] *length*, *append* sind Funktionen auf Listen.

Im folgenden beschreiben wir zwei Beispiele für Reformulierungen, die in den Analogiesystemen von CL^AM bzw. OMEGA implementiert sind:

- In der Analogienutzung, die den Beweisplan von $length(append(a,b)) = length(append(b,a))$ aus Abbildung 5 in einen Beweisplan für das Target-Theorem $half(plus(a,b)) = half(plus(b,a))$[14] überträgt, wird die Reformulierung `1to2` angewendet, die u.a. die Methode WAVE(app2) im Source-Plan verdoppelt. Diese Reformulierung wird von der Abbildung von $length$ auf $half$ ausgelöst.

- In OMEGA wird die Reformulierung `Add-Argument` angewendet, wenn eine einstellige auf eine zweistellige Funktion abgebildet werden muß[15], um das Source- auf das Target-Theorem bzw. die Source- und Target-Annahmen aufeinander abzubilden. Gleichzeitig werden auch bestimmte Zeilen in den Beweisschemata von Methoden konjunktiv erweitert bzw. ein Teilplan für ein Unterziel im Beweisplan verdoppelt werden. Wenn beispielsweise die Funktion $interval(x,y)$, die zwei reelle Zahlen x, y auf ein Intervall der reellen Zahlen abbildet, auf die Target-Funktion $rectangular(x,y,s,t)$ abgebildet werden muß, die für vier reelle Zahlen x, y, s, t ein Intervall im R^2 berechnet, dann würde eine Zeile im Beweisschema
  $\emptyset \vdash a \in R \land b \in R \to clsdint(interval(a,b))$
  verändert werden zu
  $\emptyset \vdash a \in R \land b \in R \land c \in R \land d \in R \to clsdint(rectangular(a,b,c,d))$

3.1 Analogieprozedur

Die analogie-geleitete Beweisplankonstruktion, die in [Melis,1995b] eingeführt wurde, dient als Kontrollstrategie für einen Beweisplaner, denn sie **schlägt** Methoden **vor** anstatt

[14] $half, plus$ sind Funktionen auf den ganzen Zahlen
[15] oder allgemeiner eine n-stellige Funktion auf eine (n+m)-stellige

nach anwendbaren Methoden zu **suchen**. Genauer gesagt, arbeitet die Prozedur nach dem Prinzip der *derivationalen* Analogie. Das bedeutet, daß Entscheidungen (z.B. für die Auswahl von Methoden) aus dem Planungsprozeß für des Source-Problems übertragen werden als Entscheidungen im Planungsprozeß für das Target-Problem, sofern die Gründe (Justifications) für diese Entscheidungen im Target wieder gelten. Dieser Test liefert eine Berechtigung für die Übertragung der Entscheidung.

Man stelle sich etwa vor, daß während der Source-Planung eine Fallunterscheidung $(\phi_1 \vee \neg\phi_1)$ eingeführt wurde, weil das Source-Lemma $(\phi_1 \rightarrow g)$ für das gegenwärtige Ziel g bekannt ist. Möchte man diese Entscheidung auf die Target-Planung übertragen, so sollte im Target ein Lemma $(REF(\phi_1) \rightarrow REF(g))$ für eine geeignete Reformulierung REF gelten, damit die Fallunterscheidung $(REF(\phi_1) \vee \neg REF(\phi_1))$ begründet ist.

Die konkreten Justifications, die von der Analogieprozedur getestet werden, hängen selbstverständlich von den Methoden des Planers ab. Da die verschiedenen Planer verschiedene Methoden verwenden, sind auch die in Frage kommenden Justifications abhängig vom jeweiligen Planer. Die Skizze der Analogieprozedur, wie sie in Tabelle 1 gezeigt wird, ist jedoch allgemein genug, um die Analogieprozeduren in OMEGA und CL^AM zu charakterisieren:

HIER TABELLE 1

Tabelle 1 beschreibt die Analogiekomponente allgemein. Eingegeben werden ein Source-Plan, ein Source-Problem und ein Target-Problem. Zuerst wird dann versucht, ein second-order Matching für die Theoreme und danach für die Annahmen zu finden. Aus Erfolg und Mißerfolg dieses Versuchs wird auf die notwendigen Reformulierungen geschlossen. Dann werden die Reformulierungen auf den Source-Plan angewendet. Der reformulierte Plan dient als Grundlage für die sich anschließende Übertragung. Diese analoge Übertragung bearbeitet die Methoden des Source-Planes, in der Reihenfolge, in der sie in den Source-Beweisplan eingeführt worden sind. Sie überträgt eine Methode M, falls deren (reformulierten)

Justifications im Target erfüllt sind. Als Reaktion auf eine nicht erfüllte Justification kann die analoge Übertragung abgebrochen werden, kann versucht werden, die Justification nachträglich zu erfüllen (z.B. kann ein Target-Lemma vorgeschlagen werden), oder eine überflüssige Methode kann weggelassen werden. Die Reaktionen hängen von den jeweils nicht erfüllten Justifications ab.

Es gibt einen Unterschied in der Behandlung von f-Methoden (vom Vorwärts-Planen) und b-Methoden (vom Rückwärts-Planen). Für b-Methoden müssen die Ziele im Source und Target übereinstimmen (nach der Reformulierung). Für f-Methoden M dürfen höchstens m Preconditions von M im Target fehlen (Test in Schritt (*)). Hierbei ist m ein Parameter für das Vertrauen, das man darin setzt, daß die Analogie trotz fehlender Lemmata zum Erfolg führen wird. Diese fehlenden Target-Lemmata werden von der Analogie vorgeschlagen und müssen gesondert bewiesen werden.

Im Ergebnis erhält man einen partiellen Beweisplan, der noch offene Teilziele enthalten kann. Diese müssen durch normales Planen bzw. Beweisen erfüllt werden.

3.2 Erste Erfolge

Mit Hilfe der analogie-geleiteten Beweisplankonstruktion wurde aus dem Source-Beweisplan für das Theorem HB1 (siehe Abbildung 6) der partielle Beweisplan für das Target-Theorem HB2 konstruiert [Melis,1995c]. HB1 und HB2 sind die folgenden Theoreme:

THEOREM: *Heine-Borel-1* (HB1) *Wenn ein geschlossenes Intervall* $[a,b]$ *des* R^1 *von einer Familie G offener Mengen überdeckt wird, dann überdeckt bereits eine endliche Teilfamilie H von G das Intervall.* ■

THEOREM: *Heine-Borel-2* (HB2) *Wenn ein geschlossenes Intervall* $[a,b,c,d]$ *des* R^2 *von einer Familie G offener Mengen überdeckt wird, dann überdeckt bereits eine endliche*

Teilfamilie H von G das Intervall. ■

HIER Abbildung 6

Für die Beweisplankonstruktion für HB2 wurden die Reformulierungen `Symbol-Mapping` und `Add-Argument` benutzt. Als Justifications wurden die Korrektheit von Methoden und die Existenz von Lemmata getestet. Hierbei bedeutet Korrektheit von Methoden in OMEGA, daß die Zeilen des Beweisschemas auf Korrektheit überprüft werden: Liefert die Begründung tatsächlich die Aussage in der Zeile? Mittels der analogen Übertragung des HB1-Plans wurden viele Benutzerinteraktionen vermieden, die vonnöten gewesen wären, um den Beweisplan von HB2 von Hand aufzubauen. Zum Vergleich: die Konstruktion des HB1-Planes erforderte die manuelle Benennung von 40 Teilzielen, von denen 15 mit Hilfe des automatischen Beweisers OTTER bewiesen werden.

Unsere Integration von Analogienutzung in *CL^AM* [Melis,1995a] ergab, daß mit der Analogiekomponente Theoreme beweisbar wurden, die weder vorher von *CL^AM* noch von dem derzeit führenden System für induktive Beweise, Nqthm [Boyer & Moore,1988a], bewiesen werden konnten. Tabelle 2 gibt einige einfache Theoreme an, die durch *CL^AM* mit Analogie (ABALONE) beweisbar wurden. Dort enthält die erste Spalte die Namen der Source-Theoreme (oben) und der Target-Theoreme (unten), und die zweite Spalte enthält die Theoreme selbst.

HIER Tabelle 2

3.3 Wann ist automatische Analogienutzung eine gute Idee?

Unsere Erfahrung mit Analogienutzung in verschiedenen Beweisplanern zeigt, daß man drei mögliche Ziele mit dem Beweisen durch Analogie verfolgen kann: Einsparen von

Benutzerinteraktion, ermöglichen von Lösungen und einsparen von Laufzeit:

1. Für die Analogiekomponente des interaktiven Beweisplaners OMEGA ist eines der erklärten Ziele, Benutzerinteraktionen weitgehend zu vermeiden. Dies ist ein berechtigtes Ziel, unabhängig davon, welche Ressourcen die Analogieprozedur selbst verbraucht, sofern die Benutzerinteraktion eine beschränkende Ressource beim interaktiven Beweisen ist. Diese Situation ist tatsächlich häufig anzutreffen, vor allem bei komplizierten Beweisen[16] oder bei langen Beweisen wie etwa dem von HB1, wie sie z.B. in der Programmverifikation vorkommen, wo jede Verringerung der Benutzerinteraktion zu einer erheblichen Kostenreduzierung führt.

2. Sobald man versucht, ein Analogiesystem in einen automatischen Beweiser zu integrieren, verändert sich der Zweck der Analogienutzung allerdings schlagartig. Ohne näher auf die Einzelheiten der Analogie im automatischen Beweiser *CLAM* einzugehen, wird hier kurz über die gesammelten Erfahrungen und das dort verfolgte Anliegen berichtet: *CLAM* ist ein Beweisplaner, der für induktive Beweise eingesetzt wird. Dank der Zielgerichtetheit induktiver Beweise[17], wegen eines ziemlich einheitlicher Vorgehens bei induktiven Beweisen[18] und dank der cleveren Suchheuristiken des Systems ist *CLAM* mit relativ wenigen Methoden in der Lage, für viele Theoreme einen Beweis ohne viel Suche zu finden. Daher wird die Analogiekomponente ABALONE nur aufgerufen, wenn *CLAM* den Plan für einen Beweis nicht bzw. nicht in angemessener Zeit findet.

[16]1741 Lemmata wurden beispielsweise interaktiv vorgegeben in Shankars Beweis von Gödels erstem Unvollständigkeitstheorem [Boyer & Moore,1988b].

[17]Es wird versucht, die Induktionsannahme so lange umzuformen, bis die Induktionsvoraussetzung darauf anwendbar ist.

[18]Induktive Beweise verlaufen so, daß immer erst ein Induktionsschema gefunden wird und dann der Induktionsanfang und der Induktionsschritt bewiesen werden.

ABALONE konnte Pläne konstruieren, die vorher nicht gefunden werden konnten. Das ist vor allem zurückführbar auf folgende Merkmale der Analogie, die als Kontrollstrategie in CL^AM eingesetzt wurde:

   - Analogie kann nicht gegebene Target-Lemmata nach dem Vorbild von Source-Lemmata vorschlagen.
   - Analogie kann die Default-Steuerung des Planers überschreiben.
   - Analogie kann Methoden vorschlagen, die aus Effizienzgründen dem Planer nicht immer zur Verfügung stehen, sondern vom Benutzer speziell geladen werden müssen.
   - Analogie kann durch den Einsatz von Reformulierungen Beweispläne erzeugen, die sonst nicht gefunden würden.

3. In CL^AM3 war hingegen eine Nutzung von Analogie, mit dem Ziel, Laufzeit einzusparen, erfolgreich, indem sehr zeitaufwendige Teilprozesse beim Planen durch sogenannte interne Analogienutzung ersetzt wurden [Melis & Whittle,1996]. In diesem Zusammenhang kann Analogienutzung im Beweisen tatsächlich (Zeit-)Ressourcen sparen, zumal der zusätzliche Aufwand für die interne Analogienutzung selbst relativ klein ist.

In allen Situationen gilt, daß eine Analogieprozedur nur aufgerufen werden sollte, wenn der automatische Beweisplaner den Plan nicht selbst ausreichend schnell findet. Diese Empfehlung ergibt sich gleichermaßen aus unseren Erfahrungen wie aus den empirischen Ergebnissen von VanLehn und Jones [VanLehn & Jones,1993], die, auf einen kurzen Nenner gebracht, folgendes besagen: Bei Problemlösen in Newtonscher Physik

- benutzen schlechte Lerner Analogien sehr häufig und auch dann, wenn ihnen eigentlich auch effektives Regelwissen zur Verfügung steht;

- greifen gute Lerner seltener und gezielter auf Analogienutzung als Problemlösestrategie zurück. Sie benutzen Analogien, um
  - Steuerungswissen zu lernen,
  - Wissenslücken zu finden und
  - Wissenslücken (z.B. benötigte Lemmata) durch analoges Schließen zu füllen.

Ähnlich zu der Analogienutzung bei diesen guten Lernern kann unsere Analogieprozedur die Steuerung vorgeben sowie fehlende Lemmata entdecken und vorschlagen.

4 Was ist erreicht und was bleibt zu tun?

Die vorgestellte analogie-geleitete Beweisplankonstruktion wurde aufgrund der Erfahrungen mit früheren Analogiesystemen und aufgrund von Annahmen über menschliches Beweisen durch Analogie modelliert und implementiert. Sie überträgt abstrakte Repräsentationen von Beweisen, nämlich Pläne, die zu formalen Beweisen expandiert werden können. Außerdem benutzt diese Analogie Reformulierungen des Source-Plans, zunächst um eine Übereinstimmung von Source- und Target-Problem zu erreichen. Diese Reformulierungen verändern dann auch den Beweisplan.

Die bisherigen Erfolge sind ermutigend. Es bleibt aber noch eine Menge zu tun, um die Analogieprozedur robuster – z.B. für unterschiedliche Formalisierungen ein und desslben Theorems – und effizienter zu gestalten. Unter anderem müssen noch mehr häufig auftretende Reformulierungen empirisch gefunden und implementiert werden. Dies würde den Anwendungsbereich der analogie-geleiteten Beweisplankonstruktion erweitern. Die Weiterentwicklung der existierenden Beweisplaner und ihrer Methoden ist eine Voraussetzung dafür, daß bessere, vor allem hierarchisch strukturierte, Source-Pläne entstehen, deren analoge Übertragung wieder zu hierarchischen Target-Plänen führen. Umgekehrt kann Analogie zum Lernen von Methoden und Kontrollwissen des Planers beitragen.

Die implementierten Systeme können nun kognitionswissenschaftlich umgesetzt und beispielsweise durch Protokollanalyse validiert werden. Eine genauere Kenntnis der kognitiven Grundlagen des Beweisens durch Analogie könnte Änderungen der gegenwärtigen Analogieprozedur als auch eine Erweiterung der Beweisrepräsentation motivieren. In diesem Zusammenhang sehen wir u.a. folgende kognitionswissenschaftliche Forschungsdesiderata: Untersuchungen des unmittelbaren *kognitiven Prozesses* des Beweisens durch Analogie, die die Abhängigkeiten und Reihenfolge von Teilprozeduren in der Analogienutzung, die Rolle und Arten von benutzten Justifications sowie Typ und Häufigkeit von Reformulierungen analysieren.

4.1 Zum Stellenwert der gefundenen Evidenzen

Die bisher zugrundegelegten Evidenzen sind im weitesten Sinne empirische post hoc Beobachtungen. Nicht die kognitiven Prozesse beim Beweisen durch Analogie wurden untersucht, sondern Merkmale, die die Repräsentationen von Beweisen und die analogen Übertragung besitzen müssen, damit analoge Übertragungen von mathematischen Beweisen, wie sie bereits von Menschen vollzogen wurden, überhaupt möglich sind.

Wir sind uns dessen bewußt, daß Lehrbücher normalerweise nicht die Entdeckung von Beweisen reflektieren, sondern in einer didaktisch aufbereiteten Form eine systematische Darstellung eines mathematischen Gebiets beinhalten. Das spielt aber für unsere konkreten Schlußfolgerungen darüber, was als Analogienutzung gelten soll und welche Beweise daher mit rechnergestützte Analogienutzung simulierbar sein sollten, keine große Rolle. Aus der Tatsache, daß für manche Analogien das Source-Problem erst in das Target-Problem reformuliert werden muß, und zwar in einer Weise, die über die übliche Symbolabbildung hinausgeht, folgt die Notwendigkeit eines Mechanismus für Reformulierungen in automatischen

Analogiesystemen, die über die Möglichkeiten anderer Systeme für Analogienutzung beim Theorembeweisen hinausgeht.

Die Folgerungen aus dem Artikel von Leron stützen sich vor allem auf die von ihm angegebenen Beispiele von Beweisen und analogen Beweisen, die tatsächlich in der mathematischen Praxis vorkommen. Die Tatsache, daß in manchen der Analogienbildungen nur die abstrakteren Beweisbestandteile analog übertragbar sind, erfordert eine Repräsentation von Beweisen, die diese abstrakten Bestandteile explizit macht. Da die mit einer solchen Repräsentation verbundene Strukturierung von Beweisen auch entscheidend ist für andere kognitive Leistungen des Menschen, wie Verstehen eines Beweises [Leron,1983] und analogen Transfer von Problemlösungen [Catrambone,1996], liegt es nahe, eine strukturierte Repräsentation von Beweisen für das gesamte Beweissystem zugrunde zu legen.

4.2 Verwandte Ansätze des fallbasierten Schließens

Auf andere Ansätze für das automatische Beweisen durch Analogie sind wir schon in der Einleitung und in den einzelnen Kapiteln eingegangen. Die von uns verwendete Idee der derivationalen Analogie stammt aus [Carbonell,1986] und wurde in mehreren Analogiesystemen, vor allem in Planungsdomänen, instantiiert: [Kambhampati and Hendler,1992; Bhansali & Harandi,1993; Veloso,1994]. Dieses Prinzip wurde bisher jedoch nicht auf das Beweisen durch Analogie angewendet.

Andere fallbasierte Ansätze benutzen Abstraktionen des Source-Problems und der Source-Lösung vor allem, um abstrahierte Source-Problemen wiederzufinden und die abstrahierten Lösungen dann an die Bedingungen des Target zu adaptieren, siehe [Bergmann & Wilke,1996; Börner et al.,1996]. Diese Herangehensweise erleichtert das Auffinden geeigneter Source-Probleme.

Reformulierungen wurden erst kürzlich auch für andere Domänen, z.B. für eine chemischen

Domänen [Lieber & Napoli,1996] und für eine Architekturdomäne [Börner,1994], als eine Technik des fallbasiertes Schließen in komplexen Domänen gefunden.

5 Danksagung

Mein besonderer Dank gilt Jörg Siekmann, der diese Forschungen erst emöglicht hat. Die detaillierten Hinweise der anonymen Reviewer und meines Kollegen Peter Gerjets geholfen, den Artikel für das Publikum dieser Zeitschrift zu überarbeiten.

Literatur

Althoff, K.-D., Auriol, E., Barletta, R., & Manago, M. (1995). *A Review of Industrial Case-Based Reasoning Tools.* AI Intelligence, Oxford.

Anderson, J. (1984). Acquisition of proof skills in geometry. In: Michalski, R., Carbonell, J., & Mitchell, T., (eds.), *Machine Learning, An Artificial Intelligence Approach*, pages 191–219. Springer, Berlin, New York.

Bergmann, R. & Wilke, W. (1996). On the role of abstraction in case-based reasoning. In: Faltings, B. & Smith, I., (eds.), *Fourth European Workshop on Case-Based Reasoning (EWCBR-96)*, Lausanne.

Bhansali, S. & Harandi, M. (1993). Synthesis of UNIX programs using derivational analogy. *Machine Learning*, 10.

Börner, K. (1994). Towards formalization in case-based reasoning for synthesis. In: Aha, D., (ed.), *AAAI-94 Workshop on Case-Based Reasoning*, Seattle.

Börner, K., Coulon, C., Pippig, E., & Tammer, E. (1996). Structural similarity & adaptation. In: Faltings, B. & Smith, I., (eds.), *Fourth European Workshop on Case-Based Reasoning (EWCBR-96)*, Lausanne.

Boyer, R. & Moore, J. (1988a). *A Computational Logic Handbook.* Academic Press, San Diego.

Boyer, R. & Moore, J. (1988b). Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. *Machine Intelligence*, 11:83–124.

Bundy, A. (1988). The use of explicit plans to guide inductive proofs. In: Lusk, E. & Overbeek, R., (eds.), *Proc. 9th International Conference on Automated Deduction*

(CADE), volume 310 of *Lecture Notes in Computer Science*, pages 111–120, Argonne. Springer.

Bundy, A., van Harmelen, F., Hesketh, J., & Smaill, A. (1991). Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324.

Carbonell, J. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In: Michalsky, R., Carbonell, J., & Mitchell, T., (eds.), *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. Morgan Kaufmann Publ., Los Altos.

Catrambone, R. (1996). Generalizing solution procedures learned from examples. *Journal of Experimental Psychology, Learning, Memory, and Cognition*, 22(4):1020–1031.

Curien, R. (1995). *Outils pour la Preuve par Analogie.* PhD thesis, Universite Henri Poincare - Nancy.

de la Tour, T. B. & Caferra, R. (1987). Proof analogy in interactive theorem proving: A method to express and use it via second order pattern matching. In: *Proceedings of the AAAI-87*, pages 95–99.

Deussen, P. (1971). *Halbgruppen und Automaten*, volume 99 of *Heidelberger Taschenbücher*. Springer.

Faltings, G. & Deker, U. (1983). Interview: Die Neugier, etwas ganz genau wissen zu wollen. *bild der wissenschaft*, (10):169–182.

Gentner, D. (1983). Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

Gentzen, G. (1935). Untersuchungen über das logische Schließen I. *Math. Zeitschrift*, 39.

Gordon, M., Milner, R., & Wadsworth, C. (1979). *Edinburgh LCF: A Mechanized Logic of Computation.* Lecture Notes in Computer Science 78. Springer, Berlin.

Hall, R. (1989). Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39(1):39–120.

Holyoak, K. & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355.

Huang, X., Kerber, M., Kohlhase, M., Melis, E., Nesmith, D., Richts, J., and Siekmann, J. (1994). Omega-MKRP: A Proof Development Environment. In: *Proceedings of the 12th International Conference on Automated Deduction (CADE)*, Nancy.

Kambhampati, S. & Hendler, J. (1992). A validation based theory of plan modification and reuse. *Artificial Intelligence*, 55:193–253.

Kling, R. (1971). A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178.

Koedinger, K. & Anderson, J. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, 14:511–550.

Kolbe, T. & Walther, C. (1994). Reusing proofs. In: *Proceedings of 11th European Conference on Artificial Intelligence (ECAI-94)*, Amsterdam.

Kolodner, J. (1993). *Case-Based Reasoning.* Morgan Kaufmann Publishers.

Leron, U. (1983). Structuring mathematical proofs. *The American Mathematical Monthly*, 90:174–185.

Lieber, J. & Napoli, A. (1996). Using classification in case-based reasoning. In: Wahlster, W., (ed.), *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 132–136, Budapest. John Wiley.

McCune, W. (1990). Otter 2.0 users guide. Technical Report ANL-90/9, Argonne National Laboratory, Maths and CS Division, Argonne, Illinois.

Melis, E. (1993). Analogies between proofs – a case study. SEKI-Report SR-93-12, University of Saarbrücken, Saarbrücken. available from http://jswww.cs.uni-sb.de/pub/seki/index.html.

Melis, E. (1994). Representing and reformulating diagonalization methods. Technical Report CMU-CS-94-174, Carnegie Mellon University, Dept.of Computer Science, Pittsburgh, Pennsylvania, U.S.A.

Melis, E. (1995a). Analogy in CLAM. Technical Report DAI Research Paper No 766, University of Edinburgh, AI Dept, Dept. of Artificial Intelligence, Edinburgh. available from http://jswww.cs.uni-sb.de/˜melis/.

Melis, E. (1995b). A model of analogy-driven proof-plan construction. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 182–189, Montreal.

Melis, E. (1995c). Theorem proving by analogy – a compelling example. In: C.Pinto-Ferreira & Mamede, N., (eds.), *Progress in Artificial Intelligence. 7th Portuguese Conference on Artificial Intelligence, EPIA'95*, Lecture Notes in Artificial Intelligence, 990, pages 261–272, Madeira. Springer.

Melis, E. & Whittle, J. (1996). Internal analogy in inductive theorem proving. In: M.A.McRobbie & Slaney, J., (eds.), *Proceedings of the 13th Conference on Automated Deduction (CADE-96)*, Lecture Notes in Artificial Intelligence, 1104, pages 92–105, Berlin, New York. Springer.

Munyer, J. (1981). *Analogy as a Means of Discovery in Problem Solving and Learning.* PhD thesis, University of California, Santa Cruz.

Owen, S. (1990). *Analogy for Automated Reasoning.* Academic Press.

Polya, G. (1945). *How to Solve it.* Princeton University Press, Princeton.

Polya, G. (1964). *Mathematical Discovery.* Wiley & Sons Inc., New York.

Schoenfeld, A. (1985). *Mathematical Problem Solving.* Academic Press, New York.

Sternberg, R. (1977). Component processes in analogical reasoning. *Psych. Review*, 84(4):353–378.

VanLehn, K. & Jones, R. (1993). Better learners use analogical problem solving sparingly. In: *Proceedings of the Tenth International Conference on Machine Learning*, pages 338–345, Amherst, MA. Morgan Kaufmann.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning.* Springer, Berlin, New York.

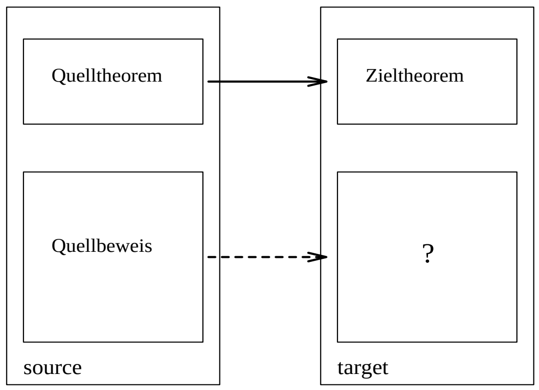

Abbildung 1: Prinzip der Analogienutzung im Theorembeweisen

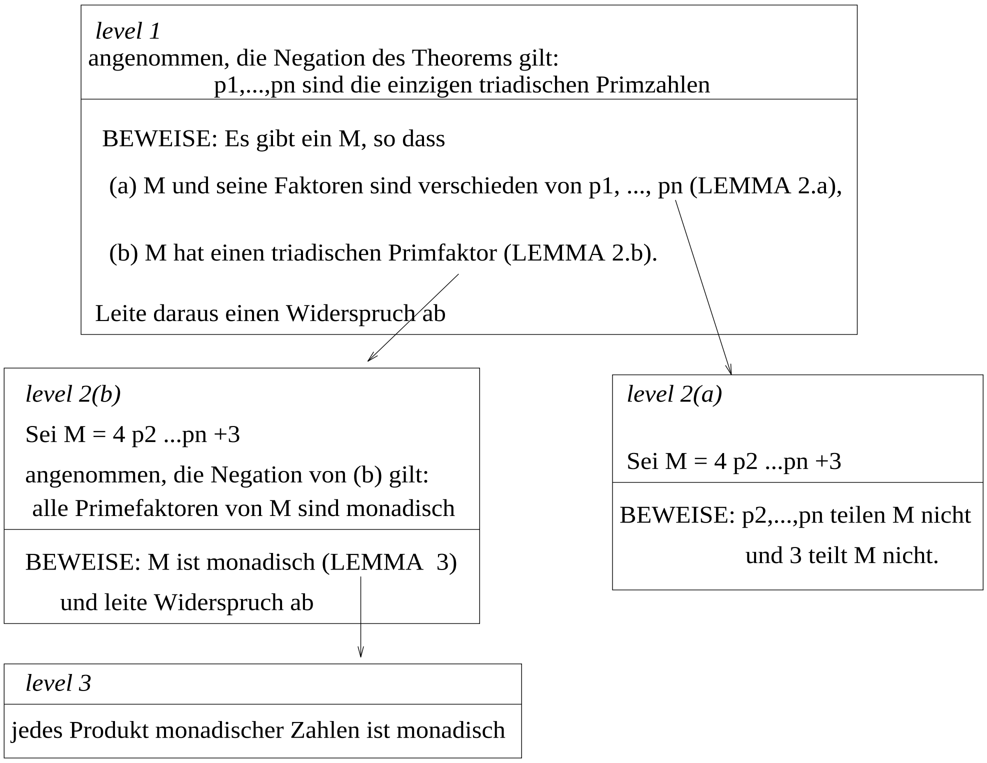

Abbildung 2: Plan des Problems TRIADIC

method: `Indirekt`	
parameter	F: formula, Δ: set of formulas
preconditions	$(\Delta \cup \{\neg F\} \vdash \bot)$
postcondition	$(\Delta \vdash F)$
constraints	
proof schema	1. $\neg F \vdash \neg F$ (HYP) 2. $\Delta, \neg F \vdash \bot$ (LEMMA) 3. $\Delta \vdash \neg\neg F$ ($\neg$I;2) 4. $\Delta \vdash F$ ($\neg$E;3)
procedure	standard schema-interpreter

Abbildung 3: Die Methode `Indirekt` in OMEGA

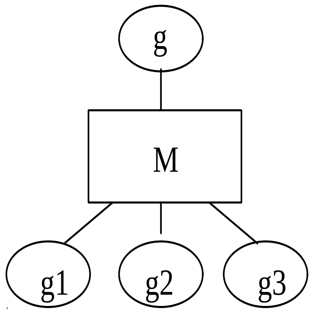

Abbildung 4: Zusammenhang von Methode und Zielen

Abbildung 5: Beweisplan von $length(append(a, b)) = length(append(b, a))$

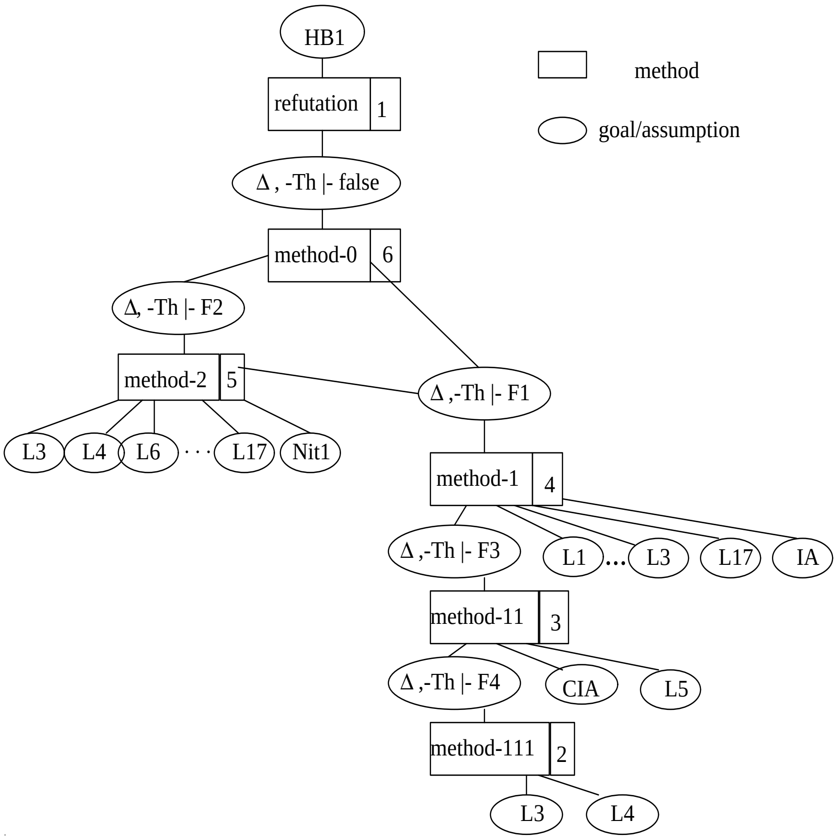

Abbildung 6: Beweisplan von HB1

input: Source-Plan, Source-Problem, Target-Problem

output: Target-Plan

Versuche second-order Matching. Reformuliere Source-Plan

while Source-Plan noch nicht erschöpft **do**

> Nächste Methode des Source-Plan ist M.
>
> **if** M ist f-Methode, **then**
>
> (*) **if** $|missing(\mathrm{M})| > m$ **then** beende Transfer.
>
> **else**
>
> Teste die Justifications von M.
>
> **if** Justifications gelten, **then** übertrage M zum Target
>
> **else** wähle geeigente Aktion.

Tabelle 1: Kurzfassung der analogie-geleiteten Beweisplankonstruktion

lenapp	$len(a <> b) = len(b <> a)$
halfplus2	$half(a + b) = half(b + a)$
zerotimes1	$x = 0 \rightarrow times(x, y) = 0$
zerotimes3	$times(x, y) = 0 \rightarrow times(times(x, y), y) = 0$
sumrev	$sum(rev(x)) = sum(x)$
evensumrev	$even(sum(rev(x))) = even(sum(x))$
assuni	$x \cup (y \cup z) = (x \cup y) \cup z$
assinter	$x \cap (y \cap z) = (x \cap y) \cap z)$
assp2	$(y + x) + z = (x + y) + z$
lenapp	$len(a <> b) = len(b <> a)$
plussum	$sum(x) + sum(y) = sum(x <> y)$
apprev	$rev(x) <> rev(y) = rev(y <> x)$
cnc_plus	$x = y \rightarrow x + z = y + z$
cnc_half	$x = y \rightarrow half(x) = half(y)$
plussum	$sum(x) + sum(y) = sum(x <> y)$
assapp	$x <> (y <> z) = (x <> y) <> z$

Tabelle 2: Einige Beispiele, die mit ABALONE bewiesen wurden