

Einbindung eines  
Computeralgebrasystems  
in eine  
logische Beweisumgebung

Diplomarbeit  
von  
Volker Sorge

Angefertigt nach einem Thema von Prof. Dr. W. Decker  
am Fachbereich Mathematik der Universität des Saarlandes



Hiermit versichere ich an Eides statt, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Saarbrücken, 14. November 1996

## Danksagungen

An dieser Stelle möchte ich mich zunächst bei Professor Wolfram Decker bedanken, der es mir ermöglichte, diese Diplomarbeit in einem Grenzgebiet der Mathematik und der Informatik zu schreiben. Ebenso bei Professor Jörg Siekmann, für die Möglichkeit, dies in seiner Deduktionssysteme Gruppe zu tun. Besonderer Dank gilt natürlich meinen beiden Betreuern Manfred Kerber und Michael Kohlhase, die nicht nur mich, während zahlreicher Diskussionen, sondern auch die Ausgeburten meines Schreibstils ertragen mußten.

Meine tiefe Verbundenheit gilt dem im Herbst 1995 leider verstorbenen Professor Woody Bledsoe, durch den ich zum ersten Mal mit dem Gebiet des automatischen Beweisens in Berührung kam.

Für viele Diskussionen und das Korrekturlesen der Arbeit bedanke ich mich bei Christoph Benzmüller und Arthur Sehn und speziell für Korrekturlesen der Einleitung bei Christoph Benzmüller, Lassaad Cheikhrouhou und Karsten Konrad. Darüber hinaus danke ich für die Überprüfung und Korrektur meiner Kommasetzung meinem Vater, dem einzigen mir bekannten Menschen, der alle 52 Kommaeregeln der deutschen Sprache beherrscht.

Ich möchte mich auch bei meinen Eltern bedanken, die mir mein Studium ermöglichten und für persönliche Zuwendung während des letzten dreiviertel Jahres der Arbeit bei meiner Freundin Catherine Vincent. Schließlich danke ich allen Mitarbeitern und (ehemaligen) Studenten der AG Deduktionssysteme für deren Kooperation bei meinen zahlreichen Problemen und insbesondere den Personen, die stets bereit waren, ihre eigene Arbeit liegenzulassen, um mit mir eine Pause einzulegen.

## Anmerkung des Autors

Aus Gründen der besseren Lesbarkeit hat sich der Autor bei der Abfassung dieser Arbeit auf die männliche Form beschränkt. Es sind damit gleichermaßen Männer wie Frauen angesprochen.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
1.1	Deduktionssysteme . . . . .	6
1.2	Computeralgebrasysteme . . . . .	7
1.3	Synthese der Systeme . . . . .	8
1.4	Aufbau dieser Arbeit . . . . .	9
<b>2</b>	<b>Motivation</b>	<b>10</b>
2.1	Motivierende Beispiele . . . . .	10
2.2	Verwandte Arbeiten . . . . .	14
2.3	CAS in Beweisplanung . . . . .	16
2.4	Fragestellungen dieser Arbeit . . . . .	18
<b>3</b>	<b>Logische Grundlagen</b>	<b>21</b>
3.1	<i>HOL</i> - Eine Logik höherer Stufe . . . . .	21
3.1.1	Syntax von <i>HOL</i> . . . . .	22
3.1.2	Semantik von <i>HOL</i> . . . . .	24
3.2	Ein Kalkül des natürlichen Schließens . . . . .	26
3.3	Taktische Theorembeweiser . . . . .	29
3.3.1	Der linearisierte ND-Kalkül . . . . .	29
3.3.2	Taktiken . . . . .	31
<b>4</b>	<b>SAPPER</b>	<b>35</b>
4.1	$\Omega$ -MKRP . . . . .	36
4.1.1	Zahlen in $\Omega$ -MKRP . . . . .	38

4.1.2	Polynome in $\Omega$ -MKRP . . . . .	39
4.2	$\mu\mathcal{CAS}$ . . . . .	42
4.2.1	Polynome in $\mu\mathcal{CAS}$ . . . . .	42
4.2.2	Algorithmen in $\mu\mathcal{CAS}$ . . . . .	44
4.3	Schnittstelle . . . . .	47
4.3.1	Übersetzer . . . . .	49
4.3.2	Taktikgenerator . . . . .	52
4.4	Arbeitsweise des Systems . . . . .	54
4.5	Eine notwendige zukünftige Erweiterung . . . . .	57
<b>5</b>	<b>Erweiterung von Computeralgebraalgorithmen</b>	<b>59</b>
5.1	Iterativer Algorithmus . . . . .	62
5.2	Rekursiver Algorithmus . . . . .	67
5.3	Devide-and-Conquer-Algorithmus . . . . .	68
<b>6</b>	<b>Ein Beispiel</b>	<b>73</b>
6.1	Problemstellung . . . . .	73
6.2	Problemkodierung in $\Omega$ -MKRP . . . . .	74
6.2.1	Eine Theorie der Einheiten . . . . .	75
6.2.2	Analytische Theoreme . . . . .	76
6.2.3	Voraussetzungen für die Polynombehandlung . . . . .	77
6.3	Problemlösung mit SAPPER . . . . .	78
<b>7</b>	<b>Zusammenfassung und Ausblick</b>	<b>82</b>
<b>A</b>	<b>Beweis des Optimierungsproblems</b>	<b>85</b>
A.1	Beweis auf der Faktenebene . . . . .	85
A.2	ND-Beweis . . . . .	90
<b>B</b>	<b>Verzeichnis der komplexen Taktiken von SAPPER</b>	<b>102</b>
B.1	Taktiken für die Polynomaddition . . . . .	102
B.2	Taktiken für die Polynommultiplikation . . . . .	104
B.3	Taktiken für die Polynomdifferenzierung . . . . .	106

---

B.4 Taktiken für das Sortieren von Polynomen . . . . .	107
<b>Literaturverzeichnis</b>	<b>108</b>
<b>Abbildungsverzeichnis</b>	<b>116</b>
<b>Symbolverzeichnis</b>	<b>118</b>
<b>Index</b>	<b>121</b>



# Kapitel 1

## Einleitung

Als Grundlagenwissenschaft für viele andere Wissenschaften blickt die Mathematik auf eine über 3000 jährige Geschichte zurück. Dabei durchlief sie einen langen Entwicklungsprozeß, in dessen Verlauf sich nicht nur ihr Umfang erweiterte, sondern in dem auch viele Begriffe präzisiert wurden. Auch der in dieser Arbeit eine wichtige Rolle spielende Begriff des formalen Beweises existierte nicht immer in seiner jetzigen Form.

Bei den ersten Versuchen, Phänomene der Natur mathematisch zu beschreiben, wurde die Korrektheit von Aussagen zunächst nur anschaulich und informell erbracht. Im Griechenland der Antike genügte es, wenn Philosophen wie PYTHAGORAS ihre geometrischen Betrachtungen mit Zeichnungen im Sand verdeutlichten. Beweise für manche mathematischen Tatsachen wurden sogar explizit verhindert: So mußte HIPPOSOS VON METAPONT vor den Schülern des PYTHAGORAS fliehen und starb auf der Flucht bei einem Schiffsunglück, als er die Existenz der irrationalen Zahlen zeigte und damit das Dogma der Zahlenharmonie bedrohte [Rob63].

Diesen philosophischen Streit zwischen Glauben und überprüfbarer Korrektheit wollte LEIBNIZ im 17. Jahrhundert dadurch beheben, daß er eine universelle Sprache schaffen wollte, die *lingua characteristica universalis*, mit der es möglich sein sollte, alle Probleme auszudrücken und durch einfaches Rechnen zu lösen [Lei86]. Dieser Versuch der radikalen Formalisierung und Abstraktion von Problemen wurde besonders in der Mathematik der folgenden Jahrhunderte unternommen, als man über immer abstraktere Begriffe zu reden begann. Konnte zum Beispiel der Begriff der irrationalen Zahlen noch anhand der Kreiszahl  $\pi$  oder des goldenen Schnittes veranschaulicht werden, blieben die von GAUSS eingeführten imaginären Zahlen „immer etwas Unwirkliches“ [Ger70].

Die Notwendigkeit, mathematische Theorien in einem korrekten und einheitlichen Formalismus zu beschreiben, führte zu einer raschen Entwicklung der Logik durch BOOLE [Boo54] und FREGE [Fre79] Mitte des letzten Jahrhunderts. Damit ergab sich die Möglichkeit zum abstrakten Lösen von Problemen, was zum Beginn der mathematischen

Moderne führte und zur Loslösung von den Naturwissenschaften, in denen die Mathematik bisher ihre Anwendung fand. Diese Loslösung gipfelte 1904 in HILBERTS Rede zum Thema „Axiom von der Lösbarkeit eines jeden Problems“ [Hil04]:

*Da ist das Problem, suche die Lösung. Du kannst sie durch reines Denken finden; denn in der Mathematik gibt es kein Ignorabimus!*

Mit dieser Rede erteilte HILBERT der Gegenmoderne eine Absage und etablierte in der Mathematik abstrakte, nichtkonstruktive Argumente als Mittel des Beweises. Damit läßt sich die Mathematik in zwei miteinander verknüpfte Gebiete unterteilen: ein konstruktives und damit anwendbares und ein abstraktes, reines.

Nach der Erfindung von Computern und dem Versuch, sie für mathematische Zwecke einzusetzen, wurde diese Teilung in reine und anwendbare Mathematik auch bei der Entwicklung mathematischer Software übernommen. So entstanden zwei Arten von Systemen: Computeralgebrasysteme, mit denen versucht wird, den anwendbaren und berechenbaren Teil der Mathematik zu implementieren, und Deduktionssysteme, die zum Führen von Beweisen auch in abstrakter Weise gedacht sind.

## 1.1 Deduktionssysteme

Die Idee, menschliches Denken zu automatisieren, die bereits Leibniz und Descartes im 17. Jahrhundert inspirierte, bildet auch eine der wichtigen Motivationen für das Gebiet der künstlichen Intelligenz. Zu den ersten Ansätzen gehörte hier der Versuch, menschliche Schlußweisen auf Rechnern zu simulieren [NSS57]. Dabei wollten NEWELL, SHAW und SIMON mit ihrem „Logic Theorist“ ein System zur Lösung allgemeiner Problemstellungen schaffen, indem sie menschliche Denkvorgänge nachvollzogen. Im Gegensatz zu diesem Ansatz, der psychologische Mechanismen in den Vordergrund stellte, entwickelten sich automatische Theorembeweiser, die versuchten mathematische Logik in maschinenorientierten Verfahren zu realisieren. Auf diesem Gebiet wurden im Verlauf der letzten Jahrzehnte verschiedene Konzepte entwickelt und in Systemen realisiert.

Mit rein automatischen Beweisern wird versucht, ohne Einwirkung eines Benutzers Theoreme zu zeigen. Dabei bedienen sich Systeme wie OTTER [McC90] oder MKRP [Prä92] logischer Kalküle, die besonders für computergeführte Beweise geeignet sind, wie der Resolutionskalkül [Rob65]. Die Stärke dieser Systeme besteht darin, große Suchräume bearbeiten zu können. Allerdings spannen viele mathematische Probleme so große Suchräume auf [SSY94], daß Lösungen nicht durch algorithmische Suche gefunden werden können. Darüber hinaus sind die von diesen Systemen erzeugten Beweise in einem für Menschen nur schwer lesbaren Format und damit auch nur schwer nachvollziehbar. Versuche, auto-

matische Beweiser auf intuitiveren Kalkülen wie dem des natürlichen Schließens [Gen35] aufzubauen, führen allerdings zu noch größeren Suchräumen [Ble83].

Um die Unwägbarkeiten der Beweissuche bei vollautomatischen Systemen zu vermeiden und gleichzeitig vom Wissen eines menschlichen Benutzers zu profitieren, wurden interaktive Beweisentwicklungsumgebungen gebaut, in denen Beweise geführt und auf ihre Korrektheit überprüft werden können. Zur Vereinfachung der Beweisführung im jeweiligen Kalkül werden zum Beispiel in IMPS [FGT90], HOL [GM93] oder *Isabelle* [Pau94] sogenannte *Taktiken* benutzt. Dies sind Prozeduren, die eine ganze Sequenz von einzelnen Inferenzschritten ausführen. Damit ist es möglich, Beweise auf einer abstrakteren Ebene als der Kalkülebene zu führen und auch zu präsentieren.

Mit einem System wie CIAM [BvHHS90] wird nun versucht, auch diese Art der Beweissuche zu automatisieren. Dafür werden die Taktiken zu *Methoden* erweitert, indem ihnen Vor- und Nachbedingungen als Spezifikationen beigelegt werden. Der Beweis für ein Theorem wird dann mit diesen Methoden abstrakt *geplant* und erst danach konkret berechnet. Dies verkleinert gegenüber automatischen Beweisern, die direkt auf Kalkülebene Beweise suchen, die zu bewältigenden Suchräume. Außerdem wird weniger darauf geachtet, daß der den Methoden zugrundeliegende Kalkül theoretisch vollständig ist, sondern es wird vielmehr versucht, konkretes Wissen zur Problemlösung in mathematischen Teildisziplinen zu erfassen.

Um nun die Vorteile all dieser Konzepte zu vereinen, werden heute in Systemen wie  $\Omega$ -MKRP [HKK<sup>+</sup>95] oder ILF [DGH<sup>+</sup>94] verschiedene Beweiser als autonome Komponenten in einer Umgebung integriert. Damit wird versucht, einen sogenannten computergestützten intelligenten mathematischen Assistenten zu schaffen, der einem Mathematiker größtmögliche Unterstützung bei seiner täglichen Arbeit, besonders dem Beweisen gibt. Diese Systeme ermöglichen es, nicht nur Beweise zu erstellen und zu überprüfen, sondern auch diese mit Erklärungen zu versehen und auf verschiedene Arten ausgeben zu lassen.

## 1.2 Computeralgebrasysteme

War die ursprüngliche Aufgabe von Computern die Durchführung großer numerischer Berechnungen, änderte sich ihr Einsatzbereich durch die Entwicklung höherer Programmiersprachen. Mit diesen war es möglich auch symbolische Ausdrücke zu manipulieren, wodurch Algorithmen zum algebraischen Rechnen entwickelt werden konnten. Aus Sammlungen solcher Algorithmen und Datenstrukturen entstanden die ersten Computeralgebrasysteme.

Auch hier entwickelte sich eine große Anzahl verschiedener Systeme, von sehr allgemein gehaltenen und in vielen mathematischen Bereichen einsetzbaren [Hea87, Wol91, CGG<sup>+</sup>92] bis hin zu sehr spezialisierten, die sich auf mathematische Teilgebiete, wie Differentialglei-

chungen, Zahlen- oder Gruppentheorie, beschränken [Boc89, BC94, Gro94]. Moderne Computeralgebrasysteme, wie z.B. *Maple* [CGG<sup>+</sup>92] oder *Mathematica* [Wol91], stellen nicht nur algebraische Algorithmen zur Verfügung, sondern darüber hinaus auch graphische Benutzeroberflächen zu deren Bedienung, eine Programmiersprache zur Erweiterung des Systems und verschiedene Möglichkeiten zur Ausgabe von Graphiken, formatierten Formeln etc. Bei der Entwicklung dieser Systeme wird besonders auf effiziente Verwaltung der mathematischen Objekte und Rechengeschwindigkeit der Algorithmen Wert gelegt. Dagegen werden unterschiedliche Eigenschaften dieser Objekte nicht berücksichtigt.

Um dem Abhilfe zu schaffen, wurden in manchen Systemen [Dav92, Fuc96] elaborierte Typsysteme eingeführt, mit denen es möglich ist, mathematischen Objekten eine Axiomatisierung zugrunde zu legen. In *MuPAD* [Fuc96] kann zum Beispiel spezifiziert werden, ob Polynome einem kommutativen Polynomring angehören oder nicht. Dies hat Auswirkungen auf das Verhalten der auf diesen Objekten ausgeführten Algorithmen. So ergeben sich für die Berechnung der ersten binomischen Formel

$$(x + y)^2 = x^2 + 2xy + y^2$$

mit Kommutativitätsaxiom und ohne dieses Axiom

$$(x + y)^2 = x^2 + xy + yx + y^2.$$

Damit ist es zwar möglich, das Rechenverhalten mathematischer Objekte zu spezifizieren, aber eine explizite Repräsentation von Theoremen und vor allem deren Beweisen kann damit nicht erreicht werden. Ein weiterer Nachteil all dieser Systeme ist es, daß sie zu erfolgten Berechnungen keine Erklärungen liefern, was es dem Benutzer erschwert, diese nachzuvollziehen oder die Korrektheit der Ergebnisse zu überprüfen.

### 1.3 Synthese der Systeme

Heutige Deduktions- und Computeralgebrasysteme sind geeignet, einem Mathematiker, wenigstens teilweise, die alltägliche Arbeit zu erleichtern. Allerdings ist diese Arbeit meist nicht so einfach voneinander zu trennen, wie es in beiden Systemarten getan wird. Oft ist es notwendig, innerhalb von Beweisen Berechnungen anzustellen oder bei symbolischen Rechnungen gewisse Vor- und Nebenbedingungen zu prüfen. Zwar können teilweise ausschließlich durch den Einsatz von Computeralgebrasystemen Beweise geführt werden, wie zum Beispiel bei geometrischen Theoremen [Wan91, Ueb94], oder umgekehrt mit Deduktionssystemen allein Rechnungen ausgeführt werden [New92]. Ersteres ist nicht in jedem Fall möglich und letzteres nur selten in vertretbarer Zeit berechenbar. Aus diesem Grund wird neuerdings nach Wegen gesucht, die beiden Systemansätze zu kombinieren, was zu einer „Erweiterung des ‚Intelligenzniveaus‘ entsprechender Programme“ [Buc96] führen kann.

## 1.4 Aufbau dieser Arbeit

In dieser Arbeit wird ein Versuch einer Kombination von einem Deduktions- und einem Computeralgebrasystem vorgestellt, wobei letzterem mehr als nur die klassische Rolle des Lieferanten von Ergebnissen zufallen soll. Statt dessen werden aus Läufen von Computeralgebraalgorithmen zusätzlich Informationen gewonnen, die eine Erklärung der erfolgten Berechnung und eine korrekte Einbindung dieser in den Kalkülbeweis des Deduktionssystems ermöglichen. Dieser Ansatz wird in Kapitel 2 motiviert und mit der bisherigen Literatur zu diesem Thema verglichen. Daraus werden anschließend die konkreten Fragestellungen dieser Diplomarbeit entwickelt.

Zur Kommunikation benutzen die beiden Systeme sogenannte Taktiken, die auf GENTZENS Kalkül des natürlichen Schließens [Gen35] aufgebaut sind. Dieser wiederum formalisiert Beweisprozeduren für eine Logik höherer Stufe (Kapitel 3).

Zur Überprüfung der Brauchbarkeit des Ansatzes wird mit SAPPER (Kapitel 4) ein System entwickelt, in dem die Protokollinformationen der Computeralgebrakomponente  $\mu\text{CAS}$  in die Beweise der Deduktionskomponente  $\Omega\text{-MKRP}$  eingebunden werden können.

Um entsprechende Informationen aus Berechnungen einzelner Computeralgebraalgorithmen gewinnen zu können, müssen diese um einen zusätzlichen Ausgabemodus erweitert werden. Wie diese Erweiterung aussehen kann und welche Algorithmenschemata dazu geeignet sind, wird in Kapitel 5 anhand mehrerer Algorithmen zur Polynommultiplikation untersucht.

In Kapitel 6 wird abschließend die Arbeitsweise der kombinierten Systeme demonstriert. Hierfür wird ein Optimierungsproblem aus der Ökonomie in  $\Omega\text{-MKRP}$  spezifiziert und mittels SAPPER gelöst.

# Kapitel 2

## Motivation

In diesem Kapitel wird der Ansatz der vorliegenden Arbeit näher erläutert, indem zunächst anhand einiger einfacher Beispiele gezeigt wird, wie und warum Systeme zum numerischen und algebraischen Rechnen im Gebiet des automatischen Beweisens eingesetzt werden müssen, falls man schwierige Beweise führen will, die diese Rechnungen beinhalten. Danach wird ein kurzer Überblick über die bisherige Forschung und Literatur zu diesem Thema gegeben, und schließlich, nachdem vorausschickend einige wichtige Begriffe kurz erläutert worden sind, werden die genauen Fragestellungen für die folgenden Kapitel erarbeitet.

### 2.1 Motivierende Beispiele

Mathematische Beweise bestehen im allgemeinen aus drei verschiedenen Komponenten: numerisches Rechnen, algebraisches Rechnen und logisches Schließen. Die beiden ersten können im Prinzip vollständig durch die letztgenannte Komponente ersetzt werden, wodurch Beweise entstehen, die, formuliert in einem geeigneten Kalkül, maschinell überprüfbar sind. Allerdings ist das logische Schließen zur Simulation beider Arten des Rechnens extrem aufwendig und unhandlich. Im folgenden wollen wir diesen Aspekt anhand einiger Beispiele für numerisches und algebraisches Rechnen darlegen.

Wir betrachten als erstes eine triviale Additionsaufgabe, die jeder Schüler im ersten Schuljahr lösen lernt, nämlich die Berechnung von  $3 + 2 = 5$ . Diese erfolgt in Prädikatenlogik erster Stufe (siehe hierzu z.B. [CL73, Fit90]), wobei wir zur Repräsentation der natürlichen Zahlen als einzige Konstante 0 benutzen und alle Zahlen ungleich 0 mit Hilfe der Nachfolgerfunktion  $s : \mathbb{N} \rightarrow \mathbb{N} : s(n) = n + 1$  erzeugen. Damit entspricht unsere Aufgabe in dieser Darstellung der Gleichung  $(s(s(s(0))) + s(s(0))) = s(s(s(s(s(0))))$ . Für den vollständigen Beweis im Kalkül des natürlichen Schließens – vorgestellt in Kapitel 3.2

– benötigen wir weiterhin zwei Axiome, die die Addition definieren. Abbildung 2.1 enthält den in  $\Omega$ -MKRP (vergleiche Kapitel 4.1) geführten Beweis, wobei die mit A0 und A1 bezeichneten Zeilen die angesprochenen Axiome zum Inhalt haben.

A0.	A0	$\vdash \forall N. (0 + N) = N$	(Hyp)
L10.	A0	$\vdash (0 + s(s(0))) = s(s(0))$	( $\forall E$ A0)
A1.	A1	$\vdash \forall N. \forall M. (s(N) + M) = s((N + M))$	(Hyp)
L7.	A1	$\vdash \forall M. (s(0) + M) = s((0 + M))$	( $\forall E$ A1)
L8.	A1	$\vdash (s(0) + s(s(0))) = s((0 + s(s(0))))$	( $\forall E$ L7)
L4.	A1	$\vdash \forall M. (s(s(0)) + M) = s((s(0) + M))$	( $\forall E$ A1)
L5.	A1	$\vdash (s(s(0)) + s(s(0))) = s((s(0) + s(s(0))))$	( $\forall E$ L4)
L1.	A1	$\vdash \forall M. (s(s(s(0))) + M) = s((s(s(0)) + M))$	( $\forall E$ A1)
L2.	A1	$\vdash (s(s(s(0))) + s(s(0))) = s((s(s(0)) + s(s(0))))$	( $\forall E$ L1)
L12.		$\vdash s(s(s(s(s(0)))))) = s(s(s(s(s(0))))))$	(=I)
L9.	A0	$\vdash s(s(s((0 + s(s(0)))))) = s(s(s(s(s(0))))))$	(=Subst L10,L12)
L6.	A0, A1	$\vdash s(s((s(0) + s(s(0)))))) = s(s(s(s(s(0))))))$	(=Subst L8,L9)
L3.	A1, A0	$\vdash (s(s(s(0)) + s(s(0)))) = s(s(s(s(s(0))))))$	(=Subst L5,L6)
THM.	A0, A1	$\vdash (s(s(s(0))) + s(s(0))) = s(s(s(s(s(0))))))$	(=Subst L2,L3)

Abbildung 2.1: Beweis für  $3 + 2 = 5$  im Kalkül des natürlichen Schließens

Wie man in Abbildung 2.1 sieht, ist ein Beweis für eine simple Aussage wie  $3 + 2 = 5$  bereits relativ lang und unübersichtlich. Streng logische Beweise derartig trivialer Behauptungen sind zwar wichtig, um die absolute Korrektheit eines Beweises zu gewährleisten, führen aber zu sehr zeitaufwendigen Berechnungen und umfangreichen Passagen in einem Beweis. So wird man beim interaktiven Führen eines Beweises zu einem Theorem nicht mit jedem Detail einer Addition von natürlichen Zahlen konfrontiert werden wollen, sondern im Gegenteil – um in vertretbarer Zeit einen Beweis zu finden – möglichst numerische Berechnungen an geeignete Systeme auslagern, die diese effizient durchführen können. Dies verkürzt den ganzen Beweis und erhöhte damit dessen Lesbarkeit. Obendrein wäre sicher mancher Benutzer bereit, die Korrektheit der Berechnung auf dieser Ebene zu glauben. Glaubte er sie jedoch nicht, könnte dann der eigentliche Logikbeweis für diese ohne Interaktion des Benutzers erstellt und auf Korrektheit geprüft werden.

Nun beschränkt sich Rechnen in der Mathematik aber nicht nur auf numerische Daten, sondern beinhaltet auch die Manipulation symbolischer Formeln. Auch solches algebraisches Rechnen kann in einem Beweis auf Kalkülebene durch schrittweises logisches Schließen ausgedrückt werden.

Wir betrachten hierzu den ND-Beweis für die erste binomische Formel

$$(x + y) * (x + y) = x^2 + 2xy + y^2 \quad (2.1)$$

in Churchschem  $\lambda$ -Kalkül [Chu40] (vergleiche Kapitel 3.1). In Abbildung 2.2 ist die stark verkürzte Version des eigentlichen ND-Beweises, geführt in  $\Omega$ -MKRP dargestellt: Dabei sind

Addition, Multiplikation und Polynommultiplikation jeweils Funktionen in zwei Variablen, und als Axiome benötigen wir die Assoziativität der Addition (A+), Distributivität von rechts und von links (DR, DL), sowie zur Multiplikation von Polynomen in zwei Variablen (P\*) und zur Addition und Multiplikation von Monomen in je zwei Variablen (M+, M\*).

A+.	A+	$\vdash \forall A. \forall B. \forall C. ((A + B) + C) = (A + (B + C))$	(Hyp)
DR.	DR	$\vdash \forall A. \forall B. \forall C. ((A + B) * C) = ((A * C) + (B * C))$	(Hyp)
DL.	DL	$\vdash \forall A. \forall B. \forall C. (A * (B + C)) = ((A * B) + (A * C))$	(Hyp)
P*.	P*	$\vdash \forall P. \forall Q. (P \otimes Q) = \lambda U. \lambda V. (P.UV) * Q.UV)$	(Hyp)
M+.	M+	$\vdash \forall U. \forall V. \forall M. \forall N. \forall A. \forall B.$ $((A * (U^M * V^N)) + (B * (U^M * V^N))) =$ $((A + B) * (U^M * V^N))$	(Hyp)
M*.	M*	$\vdash \forall U. \forall V. \forall K. \forall L. \forall M. \forall N. \forall A. \forall B.$ $((A * (U^K * V^L)) * (B * (U^M * V^N))) =$ $((A * B) * (U^{(K+M)} * V^{(L+N)}))$	(Hyp)
L73.		$\vdash \lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2)) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=I)
L72.		$\vdash \lambda U. \lambda V. (U^2 + ((2 * (U^1 * V^1)) + V^2)) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(AB L73)
L64.	M+	$\vdash \lambda U. \lambda V. (U^2 + (((U^1 * V^1) + (U^1 * V^1)) + V^2)) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L71,L72)
L60.	M+, A+	$\vdash \lambda U. \lambda V. (U^2 + (((U^1 * V^1) + ((U^1 * V^1) + V^2))) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L63,L64)
L50.	A+, M+, M*	$\vdash \lambda U. \lambda V. (U^2 + (((U^1 * V^1) + ((U^1 * V^1) + (V^1 * V^1)))) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L59,L60)
L36.	A+, M+, M*, DL	$\vdash \lambda U. \lambda V. (U^2 + (((U^1 * V^1) + (V^1 * (U^1 + V^1)))) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L49,L50)
L22.	A+, M+, M*, DL	$\vdash \lambda U. \lambda V. ((U^2 + (U^1 * V^1)) + (V^1 * (U^1 + V^1))) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L35,L36)
L12.	DL, M*, M+, A+	$\vdash \lambda U. \lambda V. (((U^1 * U^1) + (U^1 * V^1)) + (V^1 * (U^1 + V^1))) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L21,L22)
L8.	A+, M+, M*, DL	$\vdash \lambda U. \lambda V. ((U^1 * (U^1 + V^1)) + (V^1 * (U^1 + V^1))) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L11,L12)
L4.	DL, M*, M+, A+, DR	$\vdash \lambda U. \lambda V. ((U^1 + V^1) * (U^1 + V^1)) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L7,L8)
THM.	DR, A+, M+, M*, DL, P*	$\vdash (\lambda X. \lambda Y. (X^1 + Y^1) \otimes \lambda X. \lambda Y. (X^1 + Y^1)) =$ $\lambda X. \lambda Y. (X^2 + ((2 * (X^1 * Y^1)) + Y^2))$	(=Subst L3,L4)

Abbildung 2.2: Der verkürzte ND-Beweis für Gleichung (2.1)

Die ausführliche Version umfaßt, wie an der Zeilennumerierung in Abbildung 2.2 schon zu erkennen ist, 73 Zeilen – zuzüglich der sechs Hypothesen und des eigentlichen Theorems – von denen allerdings die meisten Allquantorinstantiierungen der Hypothesen sind. Andere als die angeführten Zeilen tragen nur wenig Interessantes zum Beweis bei. Für die 73 Schritte werden numerische Vereinfachungen wie in Abbildung 2.1 bereits als ele-

mentar angesehen. Expandierte man diese zusätzlich, erhielte man weitere, ebenfalls wenig interessante Zeilen.

Man könnte sich nun vorstellen, diesen Beweis interaktiv zu erstellen, also einen Benutzer alle 73 Beweisschritte erzeugen zu lassen. Ein System jedoch, das anwenderfreundlich und damit auch verwendbar sein soll, sollte triviale Passagen eines Beweises zumindest halbautomatisch füllen können. Überlegt man sich die Vorgehensweise eines automatischen Beweisers, der versucht, den Beweis des Theorems THM durch Rückwärtsschließen zu erreichen, erkennt man, daß bei naheliegenderem Verfahren – Anwenden der Hypothesen an allen geeigneten Stellen der zu beweisenden Zeile – große Suchräume erzeugt werden.

Betrachten wir konkret die erste Anwendung des Assoziativgesetzes der Addition im Beweis aus Abbildung 2.2 unter den Annahmen,

- daß der Beweiser bereits weiß, daß das Assoziativgesetz anzuwenden ist,
- und auch, daß dies auf der linken Seite der Gleichheit in Zeile L60 zu tun ist,

dann existieren immer noch zwei Möglichkeiten bezüglich der Anwendung des Gesetzes auf die Formel in Zeile L60 (Um die Anschaulichkeit zu erhöhen, sind Multiplikationen eckig geklammert.):

$$\lambda U \cdot \lambda V \cdot (U^2 + ([U^1 * V^1] + ([U^1 * V^1] + V^2))) = \lambda X \cdot \lambda Y \cdot (X^2 + ([2 * [X^1 * Y^1]] + Y^2))$$

läßt sich entweder umformen zu:

$$\lambda U \cdot \lambda V \cdot (U^2 + (([U^1 * V^1] + [U^1 * V^1]) + V^2)) = \dots,$$

also Assoziativität, angewandt auf die drei letzten Summanden, was der Formel in Zeile L64 des Beweises entspricht, oder zu

$$\lambda U \cdot \lambda V \cdot ((U^2 + [U^1 * V^1]) + ([U^1 * V^1] + V^2)) = \dots,$$

wobei hier die Assoziativität auf die ersten drei Summanden angewandt wurde. Beide Möglichkeiten können zwar durchaus zu einem Beweis führen, haben aber in jedem Fall eine Vergrößerung des Suchraumes zur Folge. Fällt dies bei angeführtem Beispiel noch nicht sehr ins Gewicht, erhöhen sich die möglichen Speicher- und Zeitkosten des Beweises bei mehr als zwei möglichen Anwendungen des Assoziativgesetzes in einer Zeile doch beträchtlich. So gibt es bei analoger Vorgehensweise für den Beweis der binomischen Formel in nächsthöherer Potenz an vergleichbarer Stelle bereits drei verschiedene Anwendungsmöglichkeiten des Assoziativgesetzes der Addition. Man betrachte hierfür die Formel:

$$\lambda U.\lambda V.(U^3 + ([3 * [U^2 * V^1]] + ([U^1 * V^2] + ([2 * [U^1 * V^2]] + V^3)))) = \\ \lambda X.\lambda Y.(X^3 + ([3 * [X^2 * Y^1]] + ([3 * [X^1 * Y^2]] + Y^3)))$$

Sieht man jetzt die Beweisschritte aus Abbildung 2.2 nicht separat, sondern jeden Schritt, wie jenen von Zeile L60 zu L64, vor dem Hintergrund des vollständigen Beweises, erkennt man, daß dem Ganzen ein simpler algebraischer Algorithmus zugrunde liegt. Diesem Schema müßte, um effizient zu sein, auch der Beweiser folgen.

## 2.2 Verwandte Arbeiten

Der wahrscheinlich erste Versuch einer Verknüpfung von automatischem Beweisen und symbolischem Rechnen war jener von SUPPES und TAKAHASHI [ST89], die einen interaktiven Beweiser schufen, indem sie den Resolutionsbeweiser *Verify* mit dem *Reduce* System [Hea87] verbanden. Das System ermöglichte den Einsatz von Reduce zum Beweisen von Stetigkeitseigenschaften, war aber sehr limitiert und wies einige essentielle Fehler auf (so konnte mathematisch Unsinniges wie  $\frac{1}{0} = \frac{1}{0}$  bewiesen werden).

Ein etwas anders gelagertes Projekt wurde von CLARKE und ZHAO an der Carnegie Mellon University durchgeführt. Sie integrierten in die Umgebung von *Mathematica* [Wol91] *Analytica* einen in der Sprache von Mathematica geschriebenen Beweiser für elementare Analysis [CZ92c]. *Analytica* ist ein klassischer automatischer Beweiser ohne Benutzerinteraktion, der mit einem Sequenzen-Kalkül arbeitet. Er kann beim Beweisen das Wissen ausnutzen, das explizit von Mathematicas Algorithmen zur Verfügung gestellt wird. Mit den in *Analytica* erstellten Beweisen kann zusätzlich die Korrektheit gewisser Schritte des Algebrasystems garantiert werden, was Fehler wie die Division mit Ausdrücken, die 0 sein könnten, vermeidet. Dabei erweitert *Analytica* durch Regeln Mathematicas Möglichkeiten der Behandlung von Ungleichungen und Summen. Das System kann einige bemerkenswerte, nicht triviale Sätze beweisen, so unter anderem Weierstraß' Beispiel für die Existenz einer stetigen, nirgends differenzierbaren Funktion [CZ92a, CZ92b, CZ94].

In [HT93a] beschreiben HARRISON und THÉRY ein Interface zwischen *HOL* [GM93], einem Beweiser für Logik höherer Stufe, und einem Computeralgebrasystem (CAS), mit dessen Hilfe in *HOL* Sätze für reelle Zahlen bewiesen werden können. Hierbei wird das CAS ausschließlich als Hilfe, eine Art Orakel, zum Führen des eigentlichen Beweises in *HOL* genutzt. Das bedeutet, das CAS 'sucht' einen möglichen Beweis, indem es bestimmte Berechnungen ausführt und somit den Beweiser Ziele vorgibt. Der eigentliche Beweis wird dann ausschließlich von *HOL* erstellt, das heißt *HOL* versucht, das durch das CAS vorgegebene Ziel mit den ihm bekannten Methoden zu erreichen und den erhaltenen Beweis zu überprüfen. Als konkretes Beispiel wird in [HT93b] das Zusammenarbeiten von *HOL*

mit dem Computeralgebrasystem *Maple* [CGG<sup>+</sup>92] beschrieben. Hierbei wird sich eines kleinen Tricks bedient, um Termersetzungen von Maple innerhalb von HOL benutzen zu können. Um nicht die Konventionen von HOL zu verletzen, daß jedes Theorem aus Axiomen abgeleitet werden muß, wird in HOL eine Konstante **Maple** eingeführt, die äquivalent zum logischen Falsch ist. Damit wird es möglich aus dieser Konstanten beliebige Formeln herzuleiten, also auch Termersetzungen von Maple selbst. Jedes Theorem, in dessen Beweis ein Ergebnis von Maple benutzt wird, erhält die Konstante **Maple** als Annahme und gilt somit als korrekt bewiesen unter der Voraussetzung, daß das Resultat von Maple korrekt ist.

Eines der jüngsten Projekte ist das von BALLARIN, HOMANN und CALMET [BHC95], die den Beweiser ISABELLE [Pau94] mit *Maple* [CGG<sup>+</sup>92] so verbunden haben, daß Termvereinfachungen in Isabellebeweisen mit Maple durchgeführt werden konnten, ohne daß an letzterem Veränderungen vorgenommen werden mußten. Dabei wird der Termersetzungsmechanismus in Isabelle um Regeln erweitert, so daß Vereinfachungen mit Hilfe ausgewählter Maple-Operationen möglich sind. Zusätzlich werden Spezifikationen angegeben, mit denen Maple-Syntax und Isabelle-Syntax ineinander übergeführt werden können.

Weiterhin seien noch kurz erwähnt, UEBERBERG [Ueb94], der interaktives Beweisen und Computeralgebra für Anwendungen in der Geometrie endlicher Räume verbindet, und BÜNDGEN [Bün94], der Termersetzungssysteme und Computeralgebrasysteme kombiniert.

Neben Untersuchungen, je zwei Systeme miteinander zu verbinden, gibt es auch Ansätze für Umgebungen, in denen mehrere verschiedene Computeralgebrasysteme oder Theorembeweiser gleichzeitig benutzt werden können. Die Motivation hierfür ist es, einen allgemeinen Rahmen zu schaffen, um bestehende Systeme ohne oder nur mit geringen Veränderungen zu integrieren. Dies soll es dem Benutzer ermöglichen, verschiedene Systeme über eine einzige Schnittstelle zu bedienen.

So wird derzeit im *OpenMath* Projekt [AvLS95] versucht, einen allgemeinen Standard für die Eingabesyntax von CAS zu schaffen. Dies soll es nicht nur Benutzern ermöglichen, problemlos zwischen Systemen zu wechseln, sondern auch den Datenaustausch zwischen den Systemen untereinander und mit externen Schnittstellen zu vereinfachen. Eine solche systemunabhängige Benutzerschnittstelle ist CAS/ $\pi$  [Kaj92]. Mit ihr ist es einem erfahrenen Benutzer möglich, verschiedene CAS zu verbinden und über eine einheitliche graphische Oberfläche anzusprechen.

Ansätze für die Architektur eines Systems, an das problemlos beliebige Theorembeweiser angeschlossen werden können, werden von GIUNCHIGLIA, PECCHIARI und TALCOTT in [GPT94] skizziert. Dafür stellt diese Umgebung sowohl ein logisches Kalkül als auch Komponenten zur Kontrolle und zur Interaktion der einzelnen Beweiser zur Verfügung.

Allgemeine Möglichkeiten der Integration von Computeralgebrasystemen und automa-

tischen Beweisern werden von HOMANN und CALMET in [HC94] beschrieben. Diese Ansätze sind in [HC95] weiterentwickelt zu einer sogenannten *offenen mechanisierten mathematischen Umgebung*, in der es möglich sein soll, verschiedenartige Beweissysteme und CAS zu verbinden und mit einer gemeinsamen Oberfläche auszustatten.

In all den Projekten, in denen jeweils ein CAS und ein Beweiser kombiniert werden, werden mit den Computeralgebrasystemen Termersetzungen oder -vereinfachungen durchgeführt, von denen entweder die Korrektheit angenommen wird oder die von den Beweisern nachvollzogen werden. Für die Ausbeutung dieses Wissens eignen sich natürlich durchaus bereits bestehende Systeme, da an diesen keine oder nur wenig Modifikationen durchgeführt werden müssen.

## 2.3 CAS in Beweisplanung

Der Ansatz in dieser Arbeit unterscheidet sich von den obengenannten dahingehend, daß das Wissen der Computeralgebrasysteme nicht nur in Form reiner Ergebnisse verwendet werden soll. Statt dessen soll das implizit in den Algorithmen des Systems vorhandene Wissen genutzt werden. Bisherige Computeralgebrasysteme liefern im allgemeinen auf ein gegebenes Problem zwar eventuell eine Lösung, geben dem Benutzer aber keine oder nur sehr wenig Information darüber, wie diese Lösung errechnet wurde oder welche Schritte im einzelnen ausgeführt wurden. Dies führt zwar einerseits zu effizienteren Algorithmen, macht es aber auf der anderen Seite schwierig, die Handlungen des Systems nachzuvollziehen oder auch die Korrektheit der Berechnung zu überprüfen.

Überlegen wir uns den Ansatz eines computergestützten mathematischen Assistenzsystems, eines Systems, das möglichst alle Aspekte mathematischen Arbeitens integriert. In ihm sollte man nicht nur Berechnungen durchführen und Beweise erstellen, die auf unterster Ebene verifizierbar sind, sondern auch diese so transformieren können, daß Ausgaben in einer Form, wie sie in mathematischen Lehrbüchern usus ist, möglich sind. Dabei sollten verschiedene Komponenten Hand in Hand arbeiten und auch Computeralgebrasysteme einen Beitrag leisten, der über das bloße Durchführen algebraischer Rechnungen, ihrem klassischen Einsatzbereich, hinausgeht.

Um nun von einem Computeralgebraalgorithmus Informationen zu erhalten, die so weit verwertbar sind, daß einerseits aus ihnen ein Beweis auf Kalkülebene ermittelt werden kann und andererseits eingängige Beweiserklärungen gewonnen werden können, muß dieser weitaus mehr Informationen über sein Vorgehen preisgeben. Aus diesen Anforderungen ergeben sich zwangsläufig einige Fragen bezüglich der erforderlichen Eigenschaften solcher Algorithmen und des Systems, welches sie integriert. Bevor wir nun zu den Problemstellungen kommen, von denen einige im Verlauf dieser Arbeit näher erläutert und untersucht

werden sollen, wollen wir an dieser Stelle kurz den Begriff der *Taktik* und das Konzept von *Beweisplänen*, insbesondere *hierarchischer Beweispläne* vorstellen und dabei gleichzeitig erläutern, welche Bedeutung diesen Begriffen im Rahmen eines mathematischen Assistenzsystems wie  $\Omega$ -MKRP zukommt.

### Taktiken

Wie wir bereits an den einfachen ND-Beweisen obiger Beispiele gesehen haben, sind Beweise auf Kalkülebene sehr umfangreich und daher interaktiv nur sehr umständlich zu führen. Aus diesem Grunde ging man (zuerst im *LCF-System* [GMW79]) dazu über, aufeinanderfolgende Anwendungen von Kalkülregeln zu größeren Einheiten, sogenannten Taktiken, zusammenzufassen.

Taktiken beschreiben gewisse Regeln, wie die Ausführung des Assoziativgesetzes, die als Oberbegriff für mehrere einzelne Kalkülschritte stehen, aber auch die Anwendungen komplexerer mathematischer Techniken, wie z.B. Induktion oder Diagonalisierung. Damit ist eine Taktik eine Prozedur, die, angewandt auf eine oder mehrere Zeilen in einem ND-Beweis, eine Sequenz von Inferenzregeln erzeugt und dem Beweis hinzufügt.

Die Zusammenfassung mehrerer Taktiken zu einer Einheit oder die Kombination von Taktiken nach bestimmten Bedingungen erreicht man mit sogenannten *Taktikalen*. (Vergleiche hierzu das Begriffspaar Funktion und Funktional in der Mathematik!) Taktikale sind also Operationen zur Verknüpfung von Taktiken; so ist z.B. ein Taktikal denkbar, das entscheidet, ob in einer gegebenen Situation die Taktik A oder die Taktik B angewandt wird (Für eine ausführlichere und formale Erläuterung der Begriffe Taktik und Taktikal siehe Kapitel 3.3.2).

### Beweispläne

Eine weitere Komponente innerhalb eines mathematischen Assistenzsystems ist ein rechnergestütztes Verfahren zur *Beweisplanung*. Der Unterschied zur *Beweissuche*, der traditionellen Vorgehensweise automatischer Beweiser, ist, daß nicht mehr alle möglichen Inferenzschritte aufgezählt werden, um eine Sequenz von Schritten zu finden, die den Satz beweisen. Statt dessen wird auf einer weitaus abstrakteren Ebene mit Hilfe sogenannter *Methoden geplant*, was eine starke Verkleinerung des Suchraumes zur Folge hat. (Zum Konzept der Beweisplanung siehe auch [Her89] und allgemein zu *Planverfahren* im Bereich der künstlichen Intelligenz [Ver92]).

Methoden sind prozedurale Taktiken, versehen mit einer Spezifikation [Bun88], oder lassen sich in einen prozeduralen und einen deklarativen Gehalt aufteilen. Derartige flexible Methoden, vorgestellt in [HKRS94b, HKRS94a], werden in  $\Omega$ -MKRP benutzt und sind durch

die deklarative Sprache *DECLAME* in [Seh95] formalisiert. Der eigentliche *Beweisplan* besteht dann aus einer Sequenz dieser Methoden. In  $\Omega$ -MKRP ist dieses Konzept dahingehend erweitert, daß der Plan selbst nicht nur aus Methoden bestehen kann, die in dieser Form auch zum Planen benutzt werden können, sondern allgemein aus Operationen, die als Oberbegriffe für mehrere einzelne Beweisschritte stehen, wie zum Beispiel die im vorhergehenden Abschnitt erläuterten (unspezifizierten) Taktiken.

### Hierarchische Beweispläne

Neben der Erstellung selbst ist ein weiteres Problem beim Planen die sinnvolle und komprimierte Plandarstellung durch Zusammenfassung mehrerer Einheiten zu abstrakteren Operatoren. Dies ist vor allem im Bereich des automatischen Beweisens wichtig, da ein vollständiger Beweis auf Kalkülebene zwar wichtig ist, um dessen Korrektheit verifizieren zu können, aber für die Beweispräsentation, also der endgültigen Ausgabe, die der Benutzer von seinem System erhalten möchte, zu lang und unansehnlich ist. Hierfür sind Beweisteile zu sinnvollen Einheiten zusammenzufassen.

In  $\Omega$ -MKRP dient das Konzept der Beweispläne den unterschiedlichen Ansprüchen an ein mathematisches Assistenzsystem. Dafür sind diese Pläne hierarchisch repräsentiert. Während auf unterster Ebene ein Kalkülbeweis existiert, der vollständig auf seine Korrektheit geprüft werden kann, dienen die anderen Ebenen der vereinfachten und weniger detaillierten Beweisdarstellung und -erklärung. Dies bedeutet, daß die einzelnen Beweisschritte von Ebene zu Ebene größer werden bzw. Oberbegriffe für immer mehr einzelne Inferenzschritte sind. Schließlich soll der Beweis einen Grad der Abstraktion erreichen, der dem einer Darstellung in einem Lehrbuch nahekommt. Eine solche Hierarchieebene ist in  $\Omega$ -MKRP die *Faktenebene* [Hua94c], die genutzt wird, um mittels des integrierten *PROVERB*-Systems [Hua94b] Beweise des ND-Kalküls über eine Makroplanebene [Hua94a] – die Faktenebene – schließlich in natürliche Sprache zu übersetzen [Fie96].

## 2.4 Fragestellungen dieser Arbeit

Das Ziel dieser Arbeit ist die interaktive Beweisentwicklungsumgebung  $\Omega$ -MKRP mit einer Computeralgebrakomponente so zu verknüpfen, daß wir letzteres auf zwei unterschiedliche Arten nutzen können. Zunächst wollen wir während der Suche nach einem Beweis symbolische Rechnungen mittels des CAS lösen und der Korrektheit des Systems vertrauen. Um aber nun die Berechnung auch prüfen zu können, soll es möglich sein, einen entsprechenden ND-Beweis unter Zuhilfenahme des CAS zu erstellen. Hierfür wollen wir das CAS als eine Art externen Beweisplaner einsetzen, der entsprechend zu den von ihm durchgeführten Be-

rechnungen einen Beweisplan liefert, welcher bei Bedarf in einen vollständigen ND-Beweis expandiert werden kann. Unser Ziel ist es also, das implizit in den Computeralgebraalgorithmen vorhandene Wissen nutzbar zu machen, um daraus direkt Beweise zu gewinnen.

Zu diesem Zweck soll unser CAS mit zwei unterschiedlichen Modi ausgestattet sein. Ein Modus, in dem ausschließlich das Ergebnis einer jeweiligen Berechnung zurückgegeben wird, und ein weiterer, um zusätzlich zum Ergebnis noch Informationen zu erhalten, wie diese Berechnung erfolgte. Für die letztere Ausgabeart müssen die benutzten Algorithmen natürlich entsprechend erweitert und verändert werden, was zur Folge hat, daß wir Algorithmen aus bereits existierenden Systemen wie *Maple* [CGG<sup>+</sup>92], *Mathematica* [Wol91] oder *Reduce* [Hea87] nicht direkt verwenden können. Dies führt zu einer der Problemstellungen dieser Diplomarbeit, nämlich inwieweit wir die Algorithmen des Computeralgebrasystems verändern müssen bzw. welche Algorithmen wir überhaupt nutzbar machen können.

Diese Frage muß nun von zwei verschiedenen Standpunkten aus betrachtet werden: zum einen im Hinblick darauf, welche Ausgabe genau von einem Algorithmus erwartet wird bzw. wie sie genutzt werden soll, und zum anderen vor dem Hintergrund der Konzeption und Architektur des ganzen Systems, das  $\Omega$ -MKRP und eine Computeralgebrakomponente verbinden soll.

Da  $\Omega$ -MKRP zur Repräsentation eines Beweises eine hierarchische Struktur benutzt, sollte die Ausgabe des CAS und der davon erstellte Beweisplan ebenfalls in diesen Rahmen eingepaßt werden können. Hierfür wollen wir uns eines Taktikmechanismus bedienen. Denn während beim klassischen Planen mathematischer Beweise – auch mit Kontrollwissen – große Suchräume erschlossen werden müssen, kann beim Beweisplanen mit einem CAS davon ausgegangen werden, daß bereits, implizit im jeweiligen Algorithmus, bekannt ist, was gemacht wird. Damit muß der Plan nicht mehr gesucht, sondern nur noch erstellt werden, wodurch eine Anwendung simpler prozeduraler Taktiken möglich wird. Um später besser präsentiert werden zu können, soll der gewonnene Plan bereits durch geeignete Wahl von Taktiken eine hierarchische Struktur erhalten, in der auch die Möglichkeit einer Beweisstransformation auf Faktenebene enthalten sein soll.

Bezüglich der Architektur des zu erstellenden Systems ergibt sich zunächst das Problem, in welchem Umfang Logik- und Computeralgebrakomponente zusammenarbeiten sollen. Dies wiederum führt auf zwei weitere konkrete Fragen: Erstens, wie die Schnittstelle zwischen logischer und computeralgebraischer Einheit auszusehen hat, und zweitens, welche Ressourcen beide Einheiten gemeinsam nutzen müssen und wie sie dies tun. Auch hierfür ist es wichtig, genau zu spezifizieren, wie wir Algorithmen zum symbolischen Rechnen in  $\Omega$ -MKRP integrieren wollen.

In unserem System sollen die einzelnen Algorithmen des CAS so frei wie möglich von logischen Ausgaben gehalten werden und damit unabhängig vom benutzten Logiksystem.

Das bedeutet, daß die gelieferten Informationen auf ein Minimum reduziert und die benutzten Algorithmen nur geringfügig erweitert werden, womit das CAS selbst immer noch ein eigenständiges System bliebe. Zusätzlich wollen wir die Möglichkeit bereitstellen, neben den Algorithmen zur Beweisplanung auch noch besonders effektive Algorithmen zur Beweissuche verwenden zu können.

Konträr hierzu wäre es auch möglich, Computeralgebraalgorithmen ausschließlich zur Beweisplanung einzusetzen und in ihnen eine ausführliche Logikausgabe zu integrieren. Damit könnte zwar sehr viel Information aus dem jeweiligen Algorithmus gewonnen werden, aber dieser wäre dann zum einen überladen mit zusätzlichen logischen Berechnungen und zum anderen speziell auf die jeweilige Logikeinheit ausgerichtet. Ersteres machte ihn aber ineffizient zur Beweissuche, und letzteres beschränkte seine Einsatzmöglichkeiten außerhalb eines konkreten Logikkalküls. Algorithmen solcher Art wären dann im Prinzip nichts anderes als extrem spezialisierte Taktiken für automatische Beweiser.

Zusammenfassend ergeben sich nun als Fragestellungen dieser Diplomarbeit: Wie erweitern wir die Algorithmen um die Taktikausgabe, und was für Taktiken nutzen wir, um hierarchische Beweispläne zu erhalten? Welche Algorithmen können wir überhaupt verwenden, um Beweise zu extrahieren, und wie muß ein System aussehen, das  $\Omega$ -MKRP auf die gewünschte Weise mit einer Einheit zum symbolischen Rechnen kombiniert?

# Kapitel 3

## Logische Grundlagen

Um mathematische Beweise mit Hilfe von Computern erstellen zu können, müssen sowohl die eigentlich informelle Sprache der Mathematik in eine stark formalisierte übersetzt werden, als auch die Vorgehensweise eines Mathematikers beim Beweisen selbst simuliert werden. Zu diesem Zweck werden als Grundlagen für ersteres eine präzise Logik und für letzteres ein implementierbarer Kalkül bezüglich der Logik definiert. Für beides, Logik und Kalkül, gibt es unterschiedliche Ansätze (vergleiche [EO87, Fit90, Koh92, Sie92]), die dem Ziel einer Modellierung der Mathematik für Computer gerecht werden.

Dieses Kapitel ist der Definition dieser logischen Grundlagen gewidmet, vor deren Hintergrund  $\Omega$ -MKRP und SAPPER arbeiten. In den beiden ersten Abschnitten stellen wir zunächst eine Logik höherer Stufe, die auf einer Variante des getypten Churchschen  $\lambda$ -Kalküls beruht, und den Kalkül des natürlichen Schließens vor. Abschließend erläutern wir, wie Logik und Kalkül in taktikbasierten Theorembeweisern, speziell in  $\Omega$ -MKRP eingesetzt werden, und definieren Taktiken formal, die zur Kommunikation zwischen einer Computeralgebrakomponente und  $\Omega$ -MKRP dienen.

### 3.1 *HOL* - Eine Logik höherer Stufe

Die Sprache in mathematischen Lehrbüchern besteht zumeist aus einer Mischung von formalen und informellen Elementen. Dies ermöglicht zwar einerseits eine leichter lesbare Form der Präsentation, kann aber andererseits zu Ungenauigkeiten und damit zu Fehlern führen. Ein Ansatz, um Beweise mittels automatischer Systeme auf Computern zu führen, ist es, die mathematische Umgangssprache exakt zu formalisieren. Zu diesem Zweck benutzt die Beweisentwicklungsumgebung  $\Omega$ -MKRP die Sprache *POST* [Nes94], die auf dem Churchschen  $\lambda$ -Kalkül aufbaut [Chu40]. *POST* ist eigentlich als sortierte Logik höherer Stufe konzipiert [Koh94], aber Sorten sind im gegenwärtigen Stadium weder ausreichend

integriert noch notwendig für diese Arbeit. Daher wird hier lediglich die einfachere Darstellung von Syntax und Semantik einer unsortierten Logik höherer Stufe gegeben. Diese ist im wesentlichen an [And86] angelehnt.

### 3.1.1 Syntax von $\mathcal{HOL}$

**Definition 3.1 (Typen):** Sei  $\mathcal{T}_B$  eine nichtleere endliche Menge von Symbolen. Dann sei die Menge  $\mathcal{T}$  der *Typen* induktiv definiert als die kleinste Menge, die mit  $\mathcal{T}_B$  auch alle Typen der Form  $\alpha \rightarrow \beta$  mit  $\alpha, \beta \in \mathcal{T}$  enthält.

Die Elemente von  $\mathcal{T}_B$  heißen *Basistypen* und Typen der Form  $\alpha \rightarrow \beta$  *Funktionstypen*.

**Bemerkung 3.2:** Im folgenden benutzen wir immer eine fixe Menge von Basistypen  $\mathcal{T}_B$  und Typen  $\mathcal{T}$ . Dabei gelte stets  $\{o, \iota\} \subset \mathcal{T}_B$ , wobei  $o$  den Typ der *Wahrheitswerte* und  $\iota$  den Typ der *Individuen* bezeichnet.  $\mathcal{T}_B$  kann aber durch weitere Spezialtypen, wie z.B. in Kapitel 4.1.1 für Zahlen, erweitert werden. Als syntaktische Variablen für Elemente aus  $\mathcal{T}$  schreiben wir kleine griechische Buchstaben.

Typen der Form  $\alpha_1 \rightarrow \alpha_2 \rightarrow \dots \rightarrow \alpha_n \rightarrow \beta$  sind rechtsgeklammert, daß heißt  $T$  entspricht  $(\alpha_1 \rightarrow (\alpha_2 \rightarrow \dots \rightarrow (\alpha_n \rightarrow \beta) \dots))$ .

**Definition 3.3 (getypte Mengen):** Eine Menge von Mengen von Symbolen  $\mathcal{M}, =, \tau := \{\mathcal{M}_\alpha | \alpha \in \mathcal{T}\}$  heißt eine *getypte Familie von Mengen* über  $\mathcal{T}$ . Wir nennen  $\mathcal{M}$  *disjunkt*, wenn gilt:  $\mathcal{M}_\alpha \cap \mathcal{M}_\beta = \emptyset$ , für  $\alpha \neq \beta$  und  $\alpha, \beta \in \mathcal{T}$ .

Die Abbildung  $\tau: \mathcal{M} \rightarrow \mathcal{T}$  heißt eine *Typfunktion*, wenn für jedes  $\alpha \in \mathcal{T}$  und jedes  $f \in \mathcal{M}_\alpha$  gilt:  $\tau(f) = \alpha$ . Umgekehrt indiziert eine Typfunktion  $\tau: \mathcal{M} \rightarrow \mathcal{T}$  eine disjunkte getypte Familie  $\mathcal{M}_\tau = \{\mathcal{M}_\alpha\}$  für  $\mathcal{M}_\alpha = \tau^{-1}(\alpha)$ .

Seien  $\mathcal{D}, \mathcal{E}$  getypte Familien von Mengen über der gleichen Typmenge  $\mathcal{T}$ , dann heißt eine Familie  $\mathcal{I} := \{\mathcal{I}_\alpha: \mathcal{E}_\alpha \rightarrow \mathcal{D}_\alpha | \alpha \in \mathcal{T}\}$  von Funktionen *getypte Funktion*  $\mathcal{I}: \mathcal{E} \rightarrow \mathcal{D}$ .

**Definition 3.4 (Signatur):** Sei  $\Sigma$  eine disjunkte getypte Familie von Mengen über  $\mathcal{T}$ , dann bezeichnen wir  $\Sigma$  als *Signatur* bezüglich  $\mathcal{T}$  und die Elemente der  $\Sigma_\alpha$  als *Konstanten*. Die Signatur  $\Sigma$  umfaßt insbesondere die *logischen Konstanten*  $\{\neg, \vee, \Pi^\alpha, =^\alpha\} \subset \Sigma$  mit den Typen:  $\neg: o \rightarrow o$ ,  $\vee: o \rightarrow o \rightarrow o$ ,  $\Pi^\alpha: (\alpha \rightarrow o) \rightarrow o$ ,  $=^\alpha: (\alpha \rightarrow \alpha) \rightarrow o$ .

**Bemerkung 3.5:** In Definition 3.4 machen wir die Konstanten  $\Pi^\alpha, =^\alpha$  explizit von ihrem Argumenttyp  $\alpha$  abhängig. Das bedeutet, daß für jeden Typ  $\alpha$  aus  $\mathcal{T}$  auch genau ein Quantor  $\Pi^\alpha$  bzw. ein Gleichheitsprädikat  $=^\alpha$  existiert.

Die vorangegangenen Definitionen erlauben es uns, die Signatur als Vereinigung getypter Mengen von Konstantensymbolen zu sehen. Durch die Disjunktheit dieser Mengen ist es stets möglich mit Hilfe der Typfunktion  $\tau$  den exakten Typ einer Konstante zu ermitteln.

**Definition 3.6 (wohlgeformte Formeln):** Sei  $\Sigma$  eine Signatur bezüglich  $\mathcal{T}$  und  $\mathcal{V}$  eine Familie von getypten Mengen über  $\mathcal{T}$ , deren Elemente abzählbar unendlich sind.  $\mathcal{V}$  heißt *Menge von getypten Variablen*. Für jeden Typ  $\alpha \in \mathcal{T}$  wird die Menge  $\mathbf{wff}_\alpha(\Sigma)$  der wohlgeformten Formeln induktiv definiert mit

- (i)  $\Sigma_\alpha \subseteq \mathbf{wff}_\alpha(\Sigma)$
- (ii)  $\mathcal{V}_\alpha \subseteq \mathbf{wff}_\alpha(\Sigma)$
- (iii) Wenn  $\mathbf{A}_{\alpha \rightarrow \beta} \in \mathbf{wff}_{\alpha \rightarrow \beta}(\Sigma)$  und  $\mathbf{B}_\alpha \in \mathbf{wff}_\alpha(\Sigma)$ , dann ist auch  $\mathbf{A}\mathbf{B} \in \mathbf{wff}_\beta(\Sigma)$
- (iv) Wenn  $\mathbf{A}_\alpha \in \mathbf{wff}_\alpha(\Sigma)$  und  $X \in \mathcal{V}_\beta$ , so ist  $\lambda X.\mathbf{A} \in \mathbf{wff}_{\beta \rightarrow \alpha}(\Sigma)$

Formeln der Form  $\mathbf{A}\mathbf{B}$  heißen *Applikationen* und Formeln der Form  $\lambda X.\mathbf{A}$   *$\lambda$ -Abstraktionen*. Die Menge aller wohlgeformten Formeln über der Signatur  $\Sigma$  ist  $\mathbf{wff}(\Sigma) = \bigcup_{\alpha \in \mathcal{T}} \mathbf{wff}_\alpha(\Sigma)$ .

**Notation 3.7:** Im weiteren Verlauf dieser Arbeit benutzen wir, wann immer dies eindeutig möglich ist, für Applikation *Infix-Notation* anstelle in *Präfix-Notation*. So schreiben wir zum Beispiel  $\mathbf{A} \vee \mathbf{B}$  anstatt  $\vee \mathbf{A}\mathbf{B}$ .

**Definition 3.8 (Freie Variablen):** Sei  $\mathbf{A}, \mathbf{B} \in \mathbf{wff}(\Sigma)$  und  $Z \in \mathcal{V}_\mathcal{T}$ . Ein Vorkommen einer Variable  $Z$  heißt genau dann *gebunden* in  $\mathbf{A}$ , wenn es in einer Teilformel der Form  $\lambda Z.\mathbf{B}$  innerhalb  $\mathbf{A}$  vorkommt. Ist ein Vorkommen von  $Z$  in  $\mathbf{A}$  nicht gebunden, heißt es *frei* in  $\mathbf{A}$ . Die Menge aller Variablen mit freiem Vorkommen in  $\mathbf{A}$  ist die Menge der *Variablen!freien* von  $\mathbf{A}$ ,  $\mathbf{FV}(\mathbf{A})$ .

**Definition 3.9 ( $\lambda$ -Reduktionen):** Seien  $\mathbf{A} \in \mathbf{wff}_\alpha(\Sigma)$ ,  $\mathbf{B} \in \mathbf{wff}_\beta(\Sigma)$  und  $X, Y \in \mathcal{V}_\beta$ . Für die Formel  $\mathbf{A}$  gelten folgende Regeln der  *$\lambda$ -Reduktion*:

- (i)  $\lambda X.\mathbf{A} \rightarrow_\alpha \lambda Y.[Y/X]\mathbf{A}$ , falls  $Y \notin \mathbf{A}$  ( *$\alpha$ -Reduktion*)
- (ii)  $(\lambda X.\mathbf{A})\mathbf{B} \rightarrow_\beta [\mathbf{B}/X]\mathbf{A}$  ( *$\beta$ -Reduktion*)
- (iii)  $\lambda X.\mathbf{A}X \rightarrow_\eta \mathbf{A}$ , falls  $X \notin \mathbf{FV}(\mathbf{A})$  ( *$\eta$ -Reduktion*)

Dabei bedeutet die Schreibweise  $[\mathbf{B}/X]\mathbf{A}$ , daß alle freien Vorkommen der Variable  $X$  in  $\mathbf{A}$  durch den Term  $\mathbf{B}$  ersetzt werden. Damit entspricht die  $\alpha$ -Reduktion einer Umbenennung der  $\lambda$ -gebundenen Variable  $Y$  in  $\mathbf{A}$ .

### 3.1.2 Semantik von $HOL$

Im folgenden wollen wir für die Logik  $HOL$  eine Semantik angeben, die auf dem Typsystem  $\mathcal{T}$  aufbaut, das als Basistypen mindestens den Typ der Wahrheitswerte und den der Individuen umfaßt.

**Definition 3.10 (Interpretation von Konstanten):** Sei  $\mathcal{D}_o = \{\top, \perp\}$  die Menge der semantischen Wahrheitswerte, in der  $\top$  dem Wert *wahr* und  $\perp$  dem Wert *falsch* entspricht, sei jedes  $\mathcal{D}_{\alpha \rightarrow \beta}$  Teilmenge von  $\mathcal{F}(\mathcal{D}_\alpha, \mathcal{D}_\beta)$ , der Menge der totalen Funktionen von  $\mathcal{D}_\alpha$  nach  $\mathcal{D}_\beta$  und sei  $\Sigma$  eine Signatur bezüglich  $\mathcal{T}$ , dann heißt die getypte Funktion  $\mathcal{I} : \Sigma \rightarrow \mathcal{D}$  *Interpretation der Konstanten* mit Träger  $\mathcal{D}$ .

Seien  $\{\neg, \vee, \Pi^\alpha, =^\alpha\} \subset \Sigma$  die in Definition 3.4 eingeführten logischen Konstanten, dann schränken wir  $\mathcal{I}$  wie folgt ein:

- (i)  $\mathcal{I}(\neg)(d) = \top$  genau dann, wenn  $d = \perp$ ,  $d \in \mathcal{D}_o$
- (ii)  $\mathcal{I}(\vee)(d, e) = \top$  gdw.  $d = \top$  oder  $e = \top$ ,  $d, e \in \mathcal{D}_o$
- (iii)  $\mathcal{I}(=^\alpha)(f, g) = \top$  gdw.  $f = g$ ,  $f, g \in \mathcal{D}_\alpha$
- (iv)  $\mathcal{I}(\Pi^\alpha)(d) = \top$  gdw.  $da = \top$  für alle  $a \in \mathcal{D}_\alpha$  mit  $d \in \mathcal{D}_{\alpha \rightarrow o}$

**Definition 3.11 (Variablenbelegung):** Sei  $\mathcal{D}$  der Träger aus Definition 3.10 und  $\mathcal{V}$  die Menge der Variablen über  $\mathcal{T}$ , dann ist eine getypte Funktion  $\varphi : \mathcal{V} \rightarrow \mathcal{D}$  eine *Variablenbelegung*.

**Definition 3.12 (Denotat):** Seien  $\Sigma, \mathcal{V}$  wie bisher und  $\mathbf{wff}(\Sigma)$  wie in Definition 3.6 und seien  $\mathcal{I} : \Sigma \rightarrow \mathcal{D}$  und  $\varphi : \mathcal{V} \rightarrow \mathcal{D}$  die zugehörige Interpretation bzw. Variablenbelegung, dann wird das *Denotat*  $\mathcal{I}_\varphi : \mathbf{wff}(\Sigma) \rightarrow \mathcal{D}$  induktiv definiert durch:

- (i)  $\mathcal{I}_\varphi(X) = \varphi(X)$ , falls  $X \in \mathcal{V}$
- (ii)  $\mathcal{I}_\varphi(c) = \mathcal{I}(c)$ , falls  $c \in \Sigma$
- (iii)  $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
- (iv)  $\mathcal{I}_\varphi(\lambda X_{\alpha \bullet} \mathbf{A}_\beta)$  ist diejenige Funktion in  $\mathcal{D}_{\alpha\beta}$ , so daß für alle  $z \in \mathcal{D}_\alpha$  gilt:  
 $(\mathcal{I}_\varphi(\lambda X_{\alpha \bullet} \mathbf{A}_\beta))z := \mathcal{I}_{\varphi, [z/X]}(\mathbf{A})$ . Existiert diese Funktion nicht, so ist  $\mathcal{I}_\varphi(\lambda X_{\alpha \bullet} \mathbf{A}_\beta)$  nicht definiert.

Nach Definition 3.12 (iv) ist es möglich, daß einem  $\lambda$ -Ausdruck kein Wert zugewiesen werden kann. In den von uns behandelten semantischen Bereichen – den sogenannten Henkinmodellen [Hen50] – möchten wir dies aber ausschließen und fordern, daß jede Formel aus  $\mathbf{wff}(\Sigma)$  denotiert werden kann.

**Definition 3.13 (Henkinmodell):** Sei  $\mathcal{I}_\varphi : \mathbf{wff}(\Sigma) \rightarrow \mathcal{D}$  ein Denotat, so daß  $\mathcal{I}_\varphi$  definiert ist für jede Formel  $\mathbf{A} \in \mathbf{wff}(\Sigma)$ , dann bezeichnen wir das Paar  $\mathbf{M} = \langle \mathcal{D}, \mathcal{I} \rangle$  als *Henkinmodell*.

Nachdem nun sichergestellt ist, daß wir einer Formel aus  $\mathbf{wff}(\Sigma)$  immer ein Denotat zuweisen können, ist es nun möglich zu definieren, wann eine Aussage innerhalb eines Henkinmodells wahr ist.

**Definition 3.14:** Sei  $\mathbf{M} = \langle \mathcal{D}, \mathcal{I} \rangle$  ein Henkinmodell und  $\mathbf{F} \in \mathbf{wff}_o(\Sigma)$ , dann gilt:

- (i)  $\mathbf{F}$  heißt *gültig* in  $\mathbf{M}$ , wenn für jede Variablenbelegung  $\varphi$  gilt:  $\mathcal{I}_\varphi(\mathbf{F}) = \top$ .
- (ii)  $\mathbf{F}$  heißt *allgemeingültig* oder *Tautologie*, wenn  $\mathbf{F}$  in jedem Henkinmodell  $\langle \mathcal{D}, \mathcal{I} \rangle$  gültig ist.
- (iii) Sei  $\Gamma$  eine Menge von Sätzen, dann ist  $\Gamma$  in  $\mathbf{M}$  genau dann *gültig*, wenn alle  $\mathbf{F} \in \Gamma$  in  $\mathbf{M}$  gültig sind.
- (iv) Ein Satz  $\mathbf{F}$  *folgt* aus einer Satzmenge  $\Gamma$ , wenn  $\mathbf{F}$  in jedem Henkinmodell  $\langle \mathcal{D}, \mathcal{I} \rangle$  gültig ist, in dem auch  $\Gamma$  gültig ist.

**Notation 3.15:** Zur Vereinfachung der Schreibweise aus Definition 3.14 führen wir das *Folgerungssymbol*  $\models$  ein und schreiben  $\Gamma \models \mathbf{F}$  für  $\mathbf{F}$  folgt aus  $\Gamma$ , und  $\models \mathbf{F}$ , wenn  $\mathbf{F}$  eine Tautologie ist.

In Definition 3.13 definierten wir die sogenannten Henkinmodelle oder *verallgemeinerten Modelle*, für die gilt:

$$\mathcal{D}_{\alpha \rightarrow \beta} \subseteq \mathcal{F}(\mathcal{D}_\alpha \mathcal{D}_\beta). \quad (3.1)$$

Damit ist es uns im nächsten Abschnitt nicht nur möglich, einen korrekten Kalkül für die Logik  $\mathcal{HOL}$  anzugeben, sondern auch einen vollständigen. Vollständig bedeutet in diesem Zusammenhang, daß alle in Henkinmodellen allgemeingültigen Formeln mit diesem Kalkül auch bewiesen werden können [Hen50]. Aufgrund der Teilmengenbeziehung in (3.1) entspricht die Menge der in Henkinmodellen allgemeingültigen Aussagen nicht der Menge aller möglichen allgemeingültigen Formeln. Verzichtet man auf diese Teilmengenbeziehung, erhält man die sogenannten *Standardmodelle*, für die gilt

$$\mathcal{D}_{\alpha \rightarrow \beta} = \mathcal{F}(\mathcal{D}_\alpha, \mathcal{D}_\beta). \quad (3.2)$$

Von diesen aber hat GÖDEL 1931 mit seinem *Unvollständigkeitssatz* gezeigt, daß auf ihnen keine gleichzeitig korrekten und vollständigen Kalküle definiert werden können [Göd31].

### 3.2 Ein Kalkül des natürlichen Schließens

1935 stellte GENTZEN [Gen35] seinen Kalkül des natürlichen Schließens – kurz ND-Kalkül, von Natural Deduction Calculus aus dem Englischen – vor. Mit ihm wird die Vorgehensweise eines Mathematikers beim Beweisen simuliert, indem aus gegebenen *Hypothesen* mittels fester *Schlußregeln*, sogenannter *Inferenzregeln*, ein *Theorem* abgeleitet wird. (Für eine Einführung in den ND-Kalkül siehe auch [BE93].)

Bevor wir den eigentlichen Kalkül definieren, wollen wir einige abkürzende Schreibweisen für die logischen Konstanten einführen, die sowohl das Verständnis geführter Beweise erhöhen als auch das Arbeiten im Kalkül selbst vereinfachen. In Definition 3.4 wurden

- $\neg$  als *Negationsjunktork*
- $\vee$  als *Disjunktionsjunktork*
- $\Pi^\alpha$  als *Quantork*
- $=^\alpha$  als *Gleichheitsprädikat*

als Elemente der Signatur  $\Sigma$  gefordert. Diese genügen zwar, um eine Logik zu definieren, aber die folgenden Abkürzungen werden die Darstellung unserer Formeln vereinfachen:

- $\forall$  als *Allquantork*, so daß  $\forall X_{\alpha} \mathbf{A}_o := \Pi^\alpha(\lambda X_{\alpha} \mathbf{A})$
- $\exists$  als *Existenzquantork*, so daß  $\exists X_{\alpha} \mathbf{A}_o := \neg(\forall X_{\alpha} \neg \mathbf{A})$
- $\wedge$  als *Konjunktionsjunktork*, so daß  $\mathbf{A}_o \wedge \mathbf{B}_o := \neg(\neg \mathbf{A} \vee \neg \mathbf{B})$
- $\Rightarrow$  als *Implikationsjunktork*, so daß  $\mathbf{A}_o \Rightarrow \mathbf{B}_o := \neg \mathbf{A} \vee \mathbf{B}$
- $\Leftrightarrow$  als *Äquivalenzjunktork*, so daß  $\mathbf{A}_o \Leftrightarrow \mathbf{B}_o := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A})$

In diesem und in den restlichen Kapiteln gehen wir nun davon aus, daß uns diese Abkürzungen und auch die logischen Konstanten der Signatur zur Verfügung stehen, die wir beide gleichermaßen als logische Symbole bezeichnen.

GENTZEN selbst führt für die Regeln seines Kalküls eine baumartige Schreibweise ein:

**Definition 3.16 (Beweisbäume):** Seien  $A_1, \dots, A_n, A, B$   $\mathcal{HOL}$ -Formeln, dann sagt man  $B$  ist die *Konklusion* der *Hypothesen*  $A_1, \dots, A_n$ , geschrieben  $\frac{A_1 \dots A_n}{B} \mathcal{R}$ , wenn  $B$  aus

$A_1, \dots, A_n$  durch Anwendung einer *Inferenzregel*  $\mathcal{R}$  (siehe Definition 3.18) entsteht. Man schreibt  $\frac{[A]}{B}$ , wenn die Formel  $B$  aus  $A$  mittels endlich vieler *Inferenzschritte* ableitbar ist.

Darstellungen der Form  $\frac{A_1 \dots A_n}{B} \mathcal{R}$  und  $\frac{[A]}{B}$  heißen *Beweisbäume*.

Im Unterschied zu manch anderen Kalkülen benötigt der ND-Kalkül nur genau zwei Axiome.

**Definition 3.17 (Axiome):** Sei  $A$  eine  $\mathcal{HOL}$ -Formel, dann gelte stets

- das Axiom für Hypothesen:  $A \models A$
- das Axiom vom ausgeschlossenen Dritten (tertium non datur):  $A \vee \neg A$ .

Im Gegensatz zum einzigen Axiom benutzt der ND-Kalkül relativ viele Regeln, um gegebene Formeln schrittweise zu manipulieren. (Man betrachte zum Beispiel vergleichend den Resolutionskalkül für Prädikatenlogik erster Stufe, der minimal zwei Inferenzregeln benötigt [Rob65].) Mit diesen Regeln, vorgestellt in Definition 3.18, können Formeln mittels Einführungs- und Beseitigungsregeln für die logischen Symbole inferiert werden.

**Definition 3.18 (Inferenzregeln):** Seien  $P, Q, R$   $\mathcal{HOL}$ -Formeln, dann definieren wir bezüglich der logischen Symbole  $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \forall, \exists, =$  als *Inferenzregeln* des ND-Kalküls:

- Eine Inferenzregel, die besagt, daß wir aus dem *Falsum* ( $\perp$ ) jede Formel herleiten können (ex falso quodlibet):

$$\frac{\perp}{P} \perp E$$

- Inferenzregeln für Junktoren

$$\frac{[P]}{\perp} \neg I_1 \quad \frac{[\neg P]}{\perp} \neg I_2 \quad \frac{P \quad \neg P}{\perp} \neg E$$

$$\frac{P}{P \vee Q} \vee I_l \quad \frac{Q}{P \vee Q} \vee I_r \quad \frac{\begin{array}{c} [P] \quad [Q] \\ \vdots \quad \vdots \\ P \vee Q \quad R \end{array}}{R} \vee E$$

$$\frac{P \quad Q}{P \wedge Q} \wedge I \quad \frac{P \wedge Q}{P} \wedge E_l \quad \frac{P \wedge Q}{Q} \wedge E_r$$

$$\begin{array}{c}
[P] \\
\vdots \\
Q \\
\hline
P \Rightarrow Q \Rightarrow I
\end{array}
\qquad
\frac{P \quad P \Rightarrow Q}{Q} \Rightarrow E$$

$$\frac{P \Rightarrow Q \quad Q \Rightarrow P}{P \Leftrightarrow Q} \Leftrightarrow I
\qquad
\frac{P \Leftrightarrow Q}{Q \Rightarrow P} \Leftrightarrow E_l
\qquad
\frac{P \Leftrightarrow Q}{P \Rightarrow Q} \Leftrightarrow E_r$$

- Inferenzregeln für Quantoren. Die Schreibweise  $[t/x]P$  bedeutet, daß alle Vorkommen des Termes  $x$  in  $P$  durch den Term  $t$  ersetzt werden.

$$\frac{[x/t]P}{\forall x.P} \forall I \text{ mit Konstante } t \text{ neu in } P
\qquad
\frac{\forall x.P}{[t/x]P} \forall E$$

$$\frac{[x/t]P}{\exists x.P} \exists I
\qquad
\frac{[t/x]P}{Q} \exists E \text{ mit Term } t \text{ neu}$$

Bei der Regel zur Elimination des Existenzquantors, ist zu beachten, daß der einzusetzende Term  $t$  weder in der Konklusion noch in den Hypothesen vorkommt; bei Introdution des Allquantors darf  $t$  nicht in der Formel  $P$  enthalten sein.

- Inferenzregeln für Gleichheit:

Für Gleichheit definieren wir drei Regeln, um die Eigenschaften der Äquivalenzrelation zu nutzen,

$$\frac{}{P = P} = I
\qquad
\frac{P = Q}{Q = P} = comm
\qquad
\frac{P = Q \quad Q = R}{P = R} = trans$$

und eine weitere Regel, um Gleichheitssubstitutionen auszuführen

$$\frac{\Phi[P] \quad P = Q}{\Phi[Q]} = subst$$

Mit diesen Regeln erhalten wir einen korrekten Kalkül für unsere Logik  $\mathcal{HOL}$ . Um nun für Henkinsemantik auch vollständig zu sein, erweitern wir den Kalkül um zwei Regeln für die Extensionalität (siehe hierzu auch [And86]).

**Definition 3.19 (Extensionalität):** Seien  $\mathbf{P}, \mathbf{Q}$   $\mathcal{HOL}$ -Formeln mit  $\tau(\mathbf{P}) = \tau(\mathbf{Q}) = \alpha \rightarrow \beta$ , dann gilt:

$$\mathbf{P} = \mathbf{Q} \Leftrightarrow \forall X_\alpha (\mathbf{P}X) = (\mathbf{Q}X) \tag{3.3}$$

Aus dieser Äquivalenz ergeben sich nun zwei weitere ND-Regeln:

$$\frac{P = Q}{\forall X.(PX) = (QX)} extI
\qquad
\frac{\forall X.(PX) = (QX)}{P = Q} extE$$

### 3.3 Taktische Theorembeweiser

Die Beweisentwicklungsumgebung  $\Omega\text{-MKRP}$  stellt den im vorangegangenen Abschnitt eingeführten Kalkül zum interaktiven Beweisen zur Verfügung. Um diesen aber auch benutzerfreundlich einsetzen zu können, führt man einige Hilfsmittel für die Darstellung und die Anwendung des Kalküls ein.

Zum einen linearisiert man die baumartige Schreibweise des Kalküls aus Abschnitt 3.2. Dies ermöglicht es auch große ND-Beweise übersichtlich darzustellen. Zum anderen werden sogenannte Taktiken eingeführt, um Sequenzen einzelner Inferenzschritte anwenden zu können. Mit diesen können komplexere Beweisschritte definiert werden oder auch ganze Beweisschemata wie Induktion oder Diagonalisierung implementiert werden.

Im nächsten Unterabschnitt führen wir die linearisierte Schreibweise des ND-Kalküls von ANDREWS [And80] ein. Danach definieren wir formal die bereits in Kapitel 2 angesprochenen Begriffe *Taktik* und *Taktikal*. Diese entsprechen im wesentlichen den Definitionen in der Literatur (vergleiche zum Beispiel [Pau90, GMW79, GT94]).

#### 3.3.1 Der linearisierte ND-Kalkül

Im linearisierten ND-Kalkül besteht ein ND-Beweis nicht mehr aus einem Beweisbaum, sondern aus einer Sequenz von Beweiszeilen der Form:

$$\textit{Marke} \quad \textit{Hypothesen} \quad \vdash \quad \textit{Formel} \quad (\textit{Regel} \quad \textit{Voraussetzungen})$$

deren einzelne Teile von folgender Bedeutung sind:

- *Formel* ist eine *HOL*-Formel.
- *Marke* ist der Name der Beweiszeile, auf den sich andere Zeilen beziehen können.
- *Hypothesen* sind die Voraussetzungen, unter welchen *Formel* gültig ist. Sie sind dargestellt als eine Liste von *Marken*.
- *Regel* ist eine der Inferenzregeln aus den Definitionen 3.18 oder 3.19.
- *Voraussetzungen* sind die *Marken* der Zeilen mit den direkten Voraussetzungen für die *Regel*.

Insbesondere gilt nun, daß eine Formel allgemeingültig ist, wenn sie aus der leeren Hypothesenliste abgeleitet werden kann. Eine Zeile der Form

$$L. \quad L \quad \vdash F \quad (\text{Hyp})$$

entspricht unserem Hypothesenaxiom aus Definition 3.17. Eine noch unbewiesene Zeile in einem Beweis, also eine, für die noch keine vollständige Ableitung aus den Hypothesen erstellt wurde, ist eine sogenannte *offene* oder *geplante* Zeile und erhält als „Begründung“ das Attribut (Open) zugewiesen. In  $\Omega$ -MKRP ist anfänglich jedes Theorem eine solche offene Zeile und hat folgende Form (wobei  $F$  eine Formel und  $\mathcal{H}$  die Hypothesenliste sei):

$$\text{THM.} \quad \mathcal{H} \quad \vdash F \quad (\text{Open})$$

Da die eben vorgestellte Form der Beweisdarstellung im ND-Kalkül noch eine wichtige Rolle im restlichen Verlauf dieser Arbeit spielen wird, betrachten wir zum Abschluß dieses Abschnitts noch zwei erläuternde Beispiele.

**Beispiel 3.20:** Seien  $P, P \Rightarrow Q$  Hypothesen, dann hat die  $\Rightarrow E$ -Regel aus Def.3.18 folgende linearisierte Darstellung:

$$\begin{array}{llll} \text{L1.} & \text{L1} & \vdash P & (\text{Hyp}) \\ \text{L2.} & \text{L2} & \vdash P \Rightarrow Q & (\text{Hyp}) \\ \text{L3.} & \text{L1, L2} & \vdash Q & (\Rightarrow E \text{ L1, L2}) \end{array}$$

**Beispiel 3.21:**

Der Beweis für die Formel:  $[\forall X. P(X) \Rightarrow Q(X)] \Rightarrow [[\forall X. P(X)] \Rightarrow [\forall X. Q(X)]]$  geführt in  $\Omega$ -MKRP.

$$\begin{array}{llll} \text{L3.} & \text{L3} & \vdash [\forall X. P(X)] & (\text{Hyp}) \\ \text{L6.} & \text{L3} & \vdash P(a) & (\forall E \text{ L3}) \\ \text{L1.} & \text{L1} & \vdash [\forall X. [P(X) \Rightarrow Q(X)]] & (\text{Hyp}) \\ \text{L5.} & \text{L1} & \vdash [P(a) \Rightarrow Q(a)] & (\forall E \text{ L1}) \\ \text{L7.} & \text{L1, L3} & \vdash Q(a) & (\Rightarrow E \text{ L6, L5}) \\ \text{L4.} & \text{L1, L3} & \vdash [\forall X. Q(X)] & (\forall I \text{ L7}) \\ \text{L2.} & \text{L1} & \vdash [[\forall X. P(X)] \Rightarrow [\forall X. Q(X)]] & (\Rightarrow I \text{ L4}) \\ \text{THM.} & & \vdash [[\forall X. [P(X) \Rightarrow Q(X)]] \Rightarrow [[\forall X. P(X)] \Rightarrow [\forall X. Q(X)]]] & (\Rightarrow I \text{ L2}) \end{array}$$

Zu beachten ist bei obigen Beweis, daß die Hypothesen L1 und L3 durch rückwärtsanwenden der  $\Rightarrow I$ -Regel auf das Theorem THM bzw. die Zeile L2 entstehen.

Der gleiche Beweis in Baumstruktur:

$$\frac{\frac{\frac{[\forall X. P(X)]}{P(a)} \forall E \quad \frac{[\forall X. P(X) \Rightarrow Q(X)]}{P(a) \Rightarrow Q(a)} \forall E}{Q(a)} \Rightarrow E}{\frac{Q(a)}{[\forall X. Q(X)]} \forall I} \Rightarrow I}{[\forall X. P(X)] \Rightarrow [\forall X. Q(X)]} \Rightarrow I}{[\forall X. [P(X) \Rightarrow Q(X)]] \Rightarrow [[\forall X. P(X)] \Rightarrow [\forall X. Q(X)]]} \Rightarrow I$$

Die  $\Rightarrow I$ -Regeln sind hier anwendbar, weil  $[\forall X. P(X)]$  bzw.  $[\forall X. P(X) \Rightarrow Q(X)]$  jeweils als Voraussetzungen im Beweisbaum auftreten.

### 3.3.2 Taktiken

Informell gesehen sind Taktiken Funktionen, die, angewendet auf einen unvollständigen Beweis, eine Reihe von Inferenzregeln ausführen. Dies bedeutet, jede Taktik hat einen Beweis (oder Beweisteil) als *Eingabezustand* und liefert den manipulierten Beweis als *Ausgabezustand*. Um aus einzelnen Taktiken komplexere Einheiten zu bilden, definiert man Operationen zu deren Verknüpfung, die Taktikale.

**Definition 3.22 (Taktik):** Seien  $P_1, \dots, P_n, Q_1, \dots, Q_m$  *HOL*-Formeln und  $R_1, \dots, R_l$  Inferenzregeln, so daß  $Q_1, \dots, Q_m$  aus  $P_1, \dots, P_n$  durch die Anwendung von  $R_1, \dots, R_l$  ableitbar ist bzw. gilt:

$$\frac{P_1 \cdots P_n}{\vdots} \frac{R_1}{\vdots} \frac{R_l}{Q_1 \cdots Q_m}$$

dann heißt eine Funktion  $T$ , die angewandt auf  $P_1, \dots, P_n$  obigen Beweisbaum liefert, eine *Taktik*. Man schreibt:

$$\frac{P_1 \cdots P_n}{\vdots} T \frac{Q_1 \cdots Q_m}{\vdots}$$

**Definition 3.23 (Taktikal):** Die folgenden auf Taktiken definierten Operationen, heißen *Taktikale*:

**APPLY** wendet eine Taktik an.

Sei  $T$  eine Taktik, dann liefert  $S := \text{APPLY}(T)$  den Beweiszustand nach Ausführung von  $T$ .

**APPEND** verknüpft eine Sequenz von Taktiken.

Seien  $T_1, \dots, T_n$  Taktiken, dann liefert die Taktik  $S := \text{APPEND}(T_1, \dots, T_n)$  die Hintereinanderausführung der Taktiken  $T_1, \dots, T_n$ .

**COND** selektiert Taktiken nach bestimmten Bedingungen.

Seien  $T_1, \dots, T_n$  Taktiken und  $C_1, \dots, C_n$  Bedingungen (z.B. bezüglich des aktuellen Beweiszustands), dann liefert  $S := \text{COND}((C_1 T_1), (C_2 T_2), \dots, (C_n T_n))$  die Taktik  $T_i$ , wenn die Bedingung  $C_i$   $i = 1, \dots, n$  erfüllt ist und zugleich alle Bedingungen  $C_j$ ,  $1 \leq j \leq i \Leftrightarrow 1$ , nicht gelten.

**REPEAT-IF** wiederholt eine Taktik, solange eine Bedingung gilt.

Sei  $T$  eine Taktik und  $C$  eine Bedingung, dann liefert die Taktik  $S := \text{REPEAT-IF}(C, T)$  eine Sequenz der Taktik  $T$ .

**Bemerkung 3.24:** Im Verlauf dieser Arbeit beziehen sich die Bedingungen innerhalb der Taktikale stets darauf, ob ein vorliegender Term von einer bestimmten Form ist. Streng formal müßten wir hierfür ein Konstrukt einführen, daß dies explizit ermittelt. Um aber die Schreibweise der Taktikale zu vereinfachen, wollen wir darauf verzichten, und geben stattdessen in einer Bedingung nur den entsprechenden Term an, für den diese erfüllt ist.

**Behauptung 3.25:** *Taktiken sind korrekt.*

**Beweis:** Sei  $T$  eine Taktik, die keine Taktikale involviert. Dann kann  $T$  nach Definition 3.22 als eine endliche Sequenz  $R_1, \dots, R_l$  von ND-Regeln aus Definition 3.18 geschrieben werden. Jedes  $R_i$ ,  $i = 1 \dots l$  ist korrekt (aufgrund der Korrektheit des ND-Kalküls [Gen35]), und damit ist auch  $T$  korrekt.

Sei nun  $T'$  eine Taktik, die Taktikale involviert. Dann kann  $T'$  aufgefaßt werden als Verknüpfung von Taktiken ohne Taktikalen, und damit als Verknüpfungen von Sequenzen von ND-Regeln. Damit ist  $T'$  mit dem gleichen Argument wie oben korrekt. ■

**Bemerkung 3.26:** Behauptung 3.25 gibt ausschließlich Auskunft über die Korrektheit von Taktiken und Taktikalen. Damit ist weder eine Aussage getroffen, wann sie anwendbar sind, noch gewährleistet, daß eine Taktik terminiert, wenn ein REPEAT-IF-Taktikal benutzt wird. Letzteres ist bei jeder Taktik separat zu überprüfen.

Da die Begriffe Taktik und Taktikal im weiteren Verlauf dieser Arbeit eine entscheidende Rolle spielen werden, wollen wir sie an dieser Stelle noch mit zwei ausführlichen Beispielen verdeutlichen. In Beispiel 3.27 wird zunächst die ASSOC-PLUS-Taktik vorgestellt, die die Definition der Assoziativität der Addition anwendet, und in Beispiel 3.28 betrachten wir die POP-SECOND-Taktik, die zur Umordnung von Summen benutzt wird.

**Beispiel 3.27:** Als Eingabezustand benötigt die ASSOC-PLUS-Taktik das Axiom der Assoziativität der Addition

$$A+. \quad A+ \quad \vdash \forall a. \forall b. \forall c. ((a + b) + c) = (a + (b + c)) \quad (\text{Hyp})$$

und die Zeile, in der sie angewendet werden soll (hierbei ist  $\Phi$  eine beliebige Funktion und  $\mathcal{H}$  eine Hypothesenliste in der  $A+$  enthalten ist):

$$L1. \quad \mathcal{H} \quad \vdash \Phi[(m_1 + (m_2 + m_3))] \quad (\text{Open})$$

Das Ausführen der Taktik liefert dann drei Anwendungen der  $\forall E$ -Regel und eine Anwendung der  $= \text{subst}$ -Regel und somit folgenden ND-Beweisteil:

$$\begin{array}{lll} A+. & A+ & \vdash \forall a. \forall b. \forall c. ((a + b) + c) = (a + (b + c)) \quad (\text{Hyp}) \\ L2. & A+ & \vdash \forall b. \forall c. ((m_1 + b) + c) = (m_1 + (b + c)) \quad (\forall E \ A+) \\ L3. & A+ & \vdash \forall c. ((m_1 + m_2) + c) = (m_1 + (m_2 + c)) \quad (\forall E \ L2) \\ L4. & A+ & \vdash c. ((m_1 + m_2) + m_3) = (m_1 + (m_2 + m_3)) \quad (\forall E \ L3) \\ L5. & \mathcal{H} & \vdash \Phi[(m_1 + m_2) + m_3] \quad (\text{Open}) \\ L1. & \mathcal{H} & \vdash \Phi[(m_1 + (m_2 + m_3))] \quad (=Subst \ L4,L5) \end{array}$$

**Beispiel 3.28:** Die POP-SECOND-Taktik ist eine von drei Taktiken zur Umordnung von zwei Summen zu einer und formalisiert folgenden Rechenschritt:

$$\sum_{i=1}^k m_i + \sum_{i=1}^l n_i = n_1 + \sum_{i=1}^k m_i + \sum_{i=2}^l n_i \quad (3.4)$$

Sie benötigt als Eingabezustand die Axiome der Assoziativität und Kommutativität der Addition und die jeweilige Zeile, auf die es angewandt werden soll. Ihr Verhalten läßt sich formal in den Operationen aus Definition 3.23 beschreiben:

*Tactic* POP-SECOND COND((( $a + (b + c)$ ) APPEND(APPLY(ASSOC-PLUS)  
 APPLY(COMMU-PLUS)  
 APPLY(ASSOC-PLUS)))  
 (( $a + b$ ) APPLY(COMMU-PLUS)))

Hierbei stehen in den Bedingungen  $a$  für die erste Summe,  $b$  für den ersten Summanden und  $c$  für den Rest der zweiten Summe in (3.4). Damit wird das Verhalten der POP-SECOND-Taktik durch die Zusammensetzung der zweiten Summe bestimmt: Verfügt diese über zwei oder mehr Summanden, werden Assoziativität, Kommutativität und wieder Assoziativität in dieser Reihenfolge angewendet, hat sie aber nur einen Summanden, wird ausschließlich Kommutativität benutzt. Konkret sieht das Verhalten der Taktik, angewendet auf unterschiedliche Eingabezeilen, folgendermaßen aus: Zunächst sei die zu zeigende Zeile

$$L1. \quad \mathcal{H} \quad \vdash \Phi[(m + (n_1 + n_2))] \quad (\text{Open})$$

wobei hier, wie auch im nächsten kleinen Beweis  $\Phi$  einer beliebigen Funktion und  $\mathcal{H}$  einer Hypothesenliste, in der A+ und C+ enthalten sind, entspricht. Die Anwendung der POP-SECOND-Taktik liefert als  $\Omega$ -MKRP-Beweis:

$$\begin{array}{llll} A+. & A+ & \vdash \forall a \bullet \forall b \bullet \forall c \bullet ((a + b) + c) = (a + (b + c)) & (\text{Hyp}) \\ L9. & A+ & \vdash \forall b \bullet \forall c \bullet ((n_1 + b) + c) = (n_1 + (b + c)) & (\forall E A+) \\ L10. & A+ & \vdash \forall c \bullet ((n_1 + m) + c) = (n_1 + (m + c)) & (\forall E L9) \\ L11. & A+ & \vdash ((n_1 + m) + n_2) = (n_1 + (m + n_2)) & (\forall E L10) \\ L2. & A+ & \vdash \forall b \bullet \forall c \bullet ((m + b) + c) = (m + (b + c)) & (\forall E A+) \\ L3. & A+ & \vdash \forall c \bullet ((m + n_1) + c) = (m + (n_1 + c)) & (\forall E L2) \\ L4. & A+ & \vdash ((m + n_1) + n_2) = (m + (n_1 + n_2)) & (\forall E L3) \\ C+. & C+ & \vdash \forall a \bullet \forall b \bullet (a + b) = (b + a) & (\text{Hyp}) \\ L6. & C+ & \vdash \forall b \bullet (m + b) = (b + m) & (\forall E C+) \\ L7. & C+ & \vdash (m + n_1) = (n_1 + m) & (\forall E L6) \\ L12. & \mathcal{H} & \vdash \Phi[(n_1 + (m + n_2))] & (\text{Open}) \\ L8. & \mathcal{H} & \vdash \Phi[((n_1 + m) + n_2)] & (=Subst L11,L12) \\ L5. & \mathcal{H} & \vdash \Phi[((m + n_1) + n_2)] & (=Subst L7,L8) \\ L1. & \mathcal{H} & \vdash \Phi[(m + (n_1 + n_2))] & (=Subst L4,L5) \end{array}$$

Zur Illustration des unterschiedlichen Verhaltens bezüglich der Eingabezeilen wenden wir nun die Taktik auf die Zeile

$$L1. \quad \mathcal{H} \quad \vdash \Phi[(m + n)] \quad (\text{Open})$$

an und erhalten die einfachere Ausführung der COMMU-PLUS-Taktik:

C+.	c+	$\vdash \forall a, \forall b, (a + b) = (b + a)$	(Hyp)
L2.	c+	$\vdash \forall b, (m + b) = (b + m)$	( $\forall E$ C+)
L3.	c+	$\vdash (m + n) = (n + m)$	( $\forall E$ L2)
L4.	$\mathcal{H}$	$\vdash \Phi[(n + m)]$	(Open)
L1.	$\mathcal{H}$	$\vdash \Phi[(m + n)]$	( $=$ Subst L3,L4)

## Kapitel 4

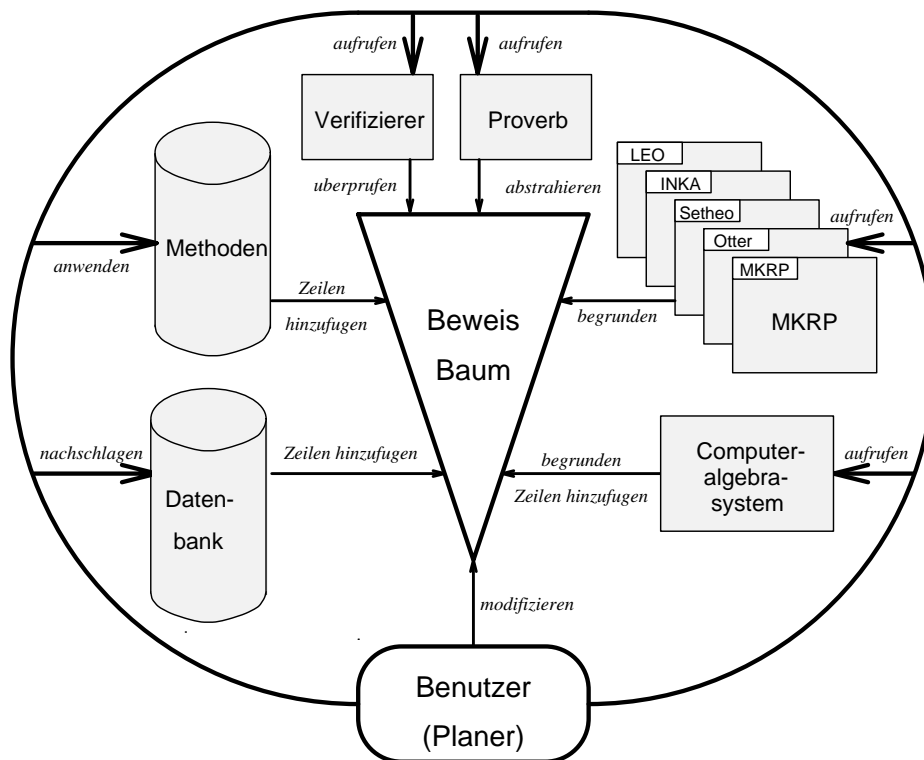
# SAPPER— Eine Schnittstelle zwischen Deduktion und Computeralgebra

In diesem Kapitel wollen wir nun das SAPPER<sup>1</sup>-System vorstellen, mit dem unser eigentliches Ziel, das Extrahieren hierarchischer Beweispläne aus computeralgebraischen Berechnungen realisiert wird. Die zentrale Idee von SAPPER ist das Verbinden der in Abschnitt 4.2 beschriebenen Computeralgebrakomponente mit der interaktiven Beweisentwicklungsumgebung  $\Omega$ -MKRP [HKK<sup>+</sup>94a] (siehe Abschnitt 4.1). Zunächst geben wir einen allgemeinen Überblick des  $\Omega$ -MKRP-Systems und verfeinern diesen im Hinblick auf die in dieser Arbeit vorgenommenen Erweiterungen und Änderungen. Im darauffolgenden Abschnitt erläutern wir  $\mu$ CAS, einen einfachen Prototyp eines Computeralgebrasystems zur Manipulation von Polynomen und erläutern anhand dieses Prototyps die Anforderungen an ein solches System als Bestandteil von SAPPER. Aus diesen Anforderungen entwickeln wir im letzten Abschnitt dieses Kapitels die Spezifikation einer allgemeinen Schnittstelle, welche  $\Omega$ -MKRP mit  $\mu$ CAS verbindet.

Das im folgenden vorgestellte System wurde im Rahmen dieser Arbeit vom Autor in KEIM [HKK<sup>+</sup>94b, Nes94], einer Programmbibliothek mit Datenstrukturen und Algorithmen für das automatische Beweisen, implementiert. Die Funktionen von KEIM stellen eine Erweiterung der Programmiersprache CLOS [Kee89] dar, welche ihrerseits wiederum die objektorientierte Weiterentwicklung von COMMON LISP [Ste90] ist. Da KEIM ebenfalls die Grundlage für das  $\Omega$ -MKRP-System ist, ist es naheliegend, daß die Schnittstelle zwischen  $\Omega$ -MKRP und dem Computeralgebrasystem  $\mu$ CAS auf die gleichen Funktionen zurückgreifen kann. Aus diesem Grund ist der Programmcode für die Schnittstelle nicht nur in KEIM ge-

---

<sup>1</sup>System for Algorithmic Proofplan Extraction and Reasoning

Abbildung 4.1: Die  $\Omega$ -MKRP Beweisentwicklungsumgebung

halten, sondern benutzt zusätzlich auch einige spezielle Funktionen von  $\Omega$ -MKRP. Im Gegensatz hierzu stellt  $\mu CAS$  ein eigenständiges, von KEIM und  $\Omega$ -MKRP unabhängiges System dar, weswegen es vom Autor in reinem CLOS-Code realisiert wurde.

## 4.1 $\Omega$ -MKRP

Das  $\Omega$ -MKRP-System ist konzipiert als mathematisches Assistenzsystem, in dem der Benutzer interaktiv Beweise führen kann [HKK<sup>+</sup>92]. Als Schnittstelle zwischen Mensch und Maschine dient dazu der ND-Kalkül [Gen35], wie er in Kapitel 3.2 vorgestellt wurde. Das Problem selbst wird am Anfang als trivialer, partieller Beweisbaum eingeführt, wobei hierbei auch die Deklaration der benutzten Typen und Konstanten, die Signatur der Logik, festgelegt wird. Dieser anfängliche, partielle Beweisbaum, bestehend aus Hypothesen und dem zu beweisenden Theorem, dient als zentrale Datenstruktur, welche so lange modifiziert wird, bis ein vollständiger Beweis erzeugt ist.

Die Modifikation des unvollständigen Beweisbaumes kann nun auf verschiedene Arten erfolgen, bei denen der Benutzer mehrere Möglichkeiten hat einzugreifen (siehe Abbil-

dung 4.1). Dies kann im einfachsten Fall durch die bloße Anwendung einzelner Inferenzschritte des ND-Kalküls geschehen. Darüber hinaus ist es auch möglich, Planer einzusetzen, die zum Beispiel mittels Methoden oder Analogie Teilbeweise konstruieren und einzelne Beweisschritte ausführen. Zusätzlich zu diesen in  $\Omega$ -MKRP integrierten Funktionen können auch verschiedene externe Komponenten eingebunden werden, um Beweise zu führen. Damit können zum Beispiel Teilbeweise mit Hilfe bestehender automatischer Beweiser (wie OTTER [McC90] oder MKRP [Prä92]) erzeugt werden. Die einzige Anforderung an solche externen Komponenten ist die, daß sie einen Beweis im Kalkül des natürlichen Schließens liefern oder aber ihre Ausgabe in einen solchen übersetzt werden kann. Ein fertiger Beweis kann von einem einfachen Beweisprüfer, versehen mit den wenigen korrekten Inferenzregeln der ND-Kalküls, verifiziert werden.

Da sich reine Kalkülbeweise meist nicht eignen, um vom Benutzer direkt gelesen zu werden (vergleiche Kapitel 2.1), werden in  $\Omega$ -MKRP unterschiedliche Mechanismen zur Verfügung gestellt, um Beweise zu präsentieren und mit Erklärungen zu versehen. Dabei werden Beweise – wie bereits in Kapitel 2.3 angesprochen – als sogenannte *Beweispläne* in hierarchischer Form repräsentiert. Die unterste Ebene eines solchen Planes stellt einen ND-Beweis dar; die Ebenen darüber den gleichen Beweis in abstrakterer Struktur. Hierfür werden Gruppen von Inferenzschritten zu abstrakten Ableitungsschritten zusammengefaßt. Führen wir eine solche Gruppierung mehrfach durch, erhalten wir ein und denselben Beweis in verschiedenen Graden der Abstraktion, wobei jeweils einem abstrakten Ableitungsschritt mehrere Ableitungsschritte auf der darunterliegenden Hierarchieebene entsprechen. Um schließlich auch noch einen Beweis anstelle im ND-Kalkül in mathematischer Umgangssprache zu präsentieren, können mit Hilfe des *PROVERB*-Systems [Hua94b] Beweise in natürliche Sprache überführt werden.

Wie angesprochen, ist im Gegensatz zu anderen Systemen (vergleiche zum Beispiel das *LCF*- [GMW79] oder das Nuprl-System [CAB<sup>+</sup>86]), die einzige Anforderung von  $\Omega$ -MKRP an externe Komponenten die nach einer verwertbaren Protokollierung. Damit wird der Anspruch auf Korrektheit der mittels dieser Komponenten erstellten Beweise bewußt aufgegeben, wodurch Einheiten benutzt werden können, die zwar effizient, aber nicht immer korrekt sind. Zusätzlich können Taktiken erstellt oder Planer mit Methoden versehen werden, deren Korrektheit nicht a priori gewährleistet ist, da generell jeder erstellte Beweis verifiziert werden muß.

In diese Systemarchitektur wollen wir nun ein CAS als externe Komponente einpassen, die sowohl zum Ausführen von algebraischen Berechnungen als auch zur Beweisplanung genutzt werden kann. Hierzu müssen allerdings einige Erweiterungen von  $\Omega$ -MKRP vorgenommen werden, die wir uns in den nächsten Abschnitten betrachten wollen.

### 4.1.1 Zahlen in $\Omega$ -MKRP

Wie bereits in Kapitel 2 Abbildung 2.1 angedeutet, ist die Behandlung numerischer Rechnungen auf Logikebene sehr umständlich. Denn um Theoreme so allgemeingültig wie möglich zu halten, beschränkt man die verwendeten logischen Konstanten und damit auch die Zahl der Hypothesen, um deren Verhalten zu axiomatisieren, auf ein Minimum. Aus diesem Grund führt man Zahlen in der Logik meist *mengentheoretisch* ein und baut neue Zahlenmengen auf bestehenden auf.

In der Mengenlehre von ZERMELO-FRAENKEL [Zer08] kann mit Hilfe des *Extensionalitäts-* und des *Aussonderungsaxioms* eindeutig eine kleinste *induktive Menge* definiert werden, die die Rolle der natürlichen Zahlen übernehmen kann. Hierzu benutzt man ausschließlich die auf jeden Fall existierende leere Menge  $\emptyset$  und baut auf dieser induktiv weitere Mengen wie folgt auf:  $\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \dots$ . Dabei entspricht die leere Ausgangsmenge der 0 und das Prinzip der Mengenschachtelung der *Nachfolgerfunktion*  $s(n) = n + 1$  [Qui67, EFT92, EHH<sup>+</sup>92]. Mit Hilfe dieser beiden Konstanten lassen sich nun natürliche Zahlen in Logik fassen und sich bereits einige Grundrechenarten auf ihnen definieren, so zum Beispiel die Addition, wie dargestellt in Abbildung 2.1, oder die Multiplikation durch iterierte Addition. Aufbauend auf der Menge der natürlichen Zahlen, lassen sich durch Definition weiterer Funktionen größere Zahlenbereiche bilden: durch die Einführung einer *Vorgängerfunktion*  $p(n) = n \Leftrightarrow 1$  die ganzen Zahlen, durch Bildung von *Paaren* ganzer Zahlen die rationalen Zahlen, mittels Axiomatisierung *Dedekindscher Schnitte* die reellen Zahlen usw.

Um aber aufwendiges Rechnen mit Nachfolgerfunktionen etc. zu vermeiden, führen wir Zahlen direkt in  $\Omega$ -MKRP ein. In der Praxis hat es sich als Vorteil erwiesen, reelle Zahlen zur Verfügung zu haben, weil sich auf diesen die wichtigsten Grundrechenarten und auf reellen Funktionen Begriffe wie Stetigkeit und Differenzierbarkeit problemlos definieren lassen. Da im weiteren Verlauf dieser Arbeit Polynome und deren Manipulation eine große Rolle spielen werden, ist es günstig, reelle Zahlen als Polynomkoeffizienten zu nutzen, mit denen sich relativ umfangreich algebraisch rechnen läßt. (Ein bekanntes Beispiel ist die Nullstellenbestimmung eines Polynoms wie  $x^2 = 2$  in  $\mathbb{Q}[x]$  und  $\mathbb{R}[x]$ .)

Damit  $\Omega$ -MKRP direkt mit reellen Zahlen rechnen kann, muß die Logik aus Kapitel 3 erweitert werden. Zunächst vergrößern wir die Menge der Basistypen  $\mathcal{T}_B$  um den Typ der reellen Zahlen  $\nu$ , dann fügen wir der Signatur  $\Sigma$  folgende Grundrechenarten als Konstanten hinzu

$$\{+, \Leftrightarrow, *, /, \uparrow\} \in \Sigma_{(\nu, \nu) \rightarrow \nu}, \quad (4.1)$$

Deren Semantik versehen wir mit der in der Mathematik üblichen Bedeutung, also Addition, Subtraktion, Multiplikation, Division und Exponentiation in den reellen Zahlen.

Zur Realisierung dieses Konzeptes innerhalb eines Beweises bedient sich  $\Omega$ -MKRP der Taktik SIMPLIFY-NUM, mit der Berechnungen auf reellen Zahlen innerhalb eines Beweisschrittes durchgeführt werden. So entspräche die Berechnung aus Abbildung 2.1 folgendem Inferenzschritt:

$$\begin{array}{lll} \text{L1.} & \vdash 5 = 5 & (= I) \\ \text{THM.} & \vdash 3 + 2 = 5 & (\text{SIMPLIFY-NUM L1}) \end{array}$$

Die Taktik könnte so erweitert werden, daß sie bei Bedarf expandiert werden kann. Dies führte zu einem Teilbeweis wie angegeben in Abbildung 2.1. Um die konkreten Berechnungen außerhalb der Logik durchzuführen, bedient sich die Taktik eines Aufrufs des Lisp-Interpreters, in dem das  $\Omega$ -MKRP-System arbeitet.

### 4.1.2 Polynome in $\Omega$ -MKRP

Da wir uns im Verlauf dieser Arbeit noch viel mit der Manipulation von Polynomen beschäftigen werden, soll an dieser Stelle die Darstellung von Polynomen in  $\Omega$ -MKRP eingeführt und Operationen auf diesen definiert werden. Hierzu bedienen wir uns der reellen Zahlen aus Abschnitt 4.1.1 und der in Gleichung (4.1) angegebenen Grundrechenarten.

Allgemein kann ein Polynom aus  $\mathbb{K}[x_1, \dots, x_r]$  (also dem Ring der Polynome über einem Körper  $\mathbb{K}$  in  $r$  Variablen) als eine Abbildung  $\underbrace{\mathbb{K} \times \dots \times \mathbb{K}}_{r\text{-mal}} \rightarrow \mathbb{K}$  betrachtet werden. (Vergleiche hierzu zum Beispiel [vdW50, Bos93].) Um dieses funktionale Verhalten in  $\Omega$ -MKRP zu simulieren, werden Polynome als  $\lambda$ -Abstraktion über den Variablen des Polynomringes dargestellt. Konkret wird für Polynome über  $\mathbb{R}$  folgende Darstellung gewählt:

Sei  $p \in \mathbb{R}[x_1, \dots, x_r]$ ,  $p = \sum_{i=1}^r \alpha_i x_1^{e_{1i}} \cdots x_n^{e_{ri}}$ , wobei  $\alpha_i \in \mathbb{R}$  und  $e_{1i}, \dots, e_{ri} \in \mathbb{N}$ ,  $i = 1, \dots, r$ , dann entspricht  $p$  einer  $\lambda$ -Abstraktion

$$\lambda x_1 \dots \lambda x_r. (\alpha_n x_1^{e_{1n}} \cdots x_r^{e_{rn}} + \dots + (\alpha_2 x_1^{e_{12}} \cdots x_r^{e_{r2}} + \alpha_1 x_1^{e_{11}} \cdots x_r^{e_{r1}}) \cdots), \quad (4.2)$$

und es gilt für den Typ von  $p$ :  $\tau(p) = \underbrace{(\nu, \dots, \nu)}_{r\text{-mal}} \rightarrow \nu$ .

Zu beachten ist bei obiger Darstellung, daß  $p$  in der Ordnung von Definition 4.3 in Abschnitt 4.2 ist. Zusätzlich gelte die Forderung, daß in den einzelnen Monomen immer die Variablen mit Exponent 0 aufgeführt sind. (Zu den Begriffen der Polynomdarstellung vergleiche Abschnitt 4.2) Die Schreibweise des Polynoms in 4.2 entspricht bereits einer vereinfachten infix Repräsentation, die die Lesbarkeit hier und im folgenden erhöhen soll. In der Logik von  $\Omega$ -MKRP ist ein Polynom in Präfixschreibweise wie folgt repräsentiert

$$\lambda x_1 \dots \lambda x_r. (+ (* \alpha_n (* (\uparrow x_1 e_{1n}) \cdots)) \cdots (* \alpha_1 (* (\uparrow x_1 e_{11}) \cdots)) \cdots) \quad (4.3)$$

Nachdem nun Polynome in  $\Omega$ -MKRP verfügbar sind, definieren wir als nächstes die logischen Konstanten, die zu deren Manipulation benötigt werden:

- $\oplus$  für Polynomaddition
- $\otimes$  für Polynommultiplikation
- $\partial_p$  für die Differentiation eines Polynoms
- $\int_p$  für die Integration eines Polynoms

Dabei ist zu beachten, daß Polynomaddition und -multiplikation je zwei Polynome als Argumente benötigen, wogegen Polynomdifferenziation und -integration jeweils ein Polynom und eine natürliche Zahl  $j$  fordern.  $j$  gibt die Position der jeweiligen Variable an, bezüglich deren differenziert bzw. integriert wird. Ein Beispiel für die Anwendung einer der beiden ersten Operationen ist bereits in Kapitel 2 Abbildung 2.2 gegeben. Zur Erläuterung des Verhaltens der beiden letzteren Operationen seien hier kurz zwei Polynomdifferenziationen angegeben

$$\begin{aligned}\partial_p(\lambda x_\bullet \lambda y_\bullet (x^2 + 2xy + y), 1) &= \lambda x_\bullet \lambda y_\bullet (2x + 2y) \\ \partial_p(\lambda x_\bullet \lambda y_\bullet (x^2 + 2xy + y), 2) &= \lambda x_\bullet \lambda y_\bullet (2x + 1)\end{aligned}$$

Die soeben eingeführten und im Rest dieser Arbeit benutzten Symbole entsprechen wiederum nicht der eigentlichen Darstellung in  $\Omega$ -MKRP. Und dies nicht nur weil wir die Symbole in  $\Omega$ -MKRP computergerecht darstellen müssen, sondern auch weil von der logischen Sichtweise verschiedene Symbole für intuitiv gleiche Operationen auf verschiedenen Polynomringen benutzt werden müssen. Um diesen Punkt näher zu erläutern, setzen wir den Umgang mit syntaktischen Symbolen in der mathematischen Umgangssprache in Bezug zu dem in einer Logik wie  $\mathcal{HOL}$ .

In der Mathematik werden häufig für verschiedene Funktionen, die gleichartige Abbildungsvorschriften auf unterschiedlichen Mengen bezeichnen, die gleichen Funktionssymbole benutzt. So bezeichnen zum Beispiel

$$\bullet + : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \quad \bullet + : \mathbb{R}[x] \times \mathbb{R}[x] \rightarrow \mathbb{R}[x] \quad \bullet + : \mathbb{R}[x, y] \times \mathbb{R}[x, y] \rightarrow \mathbb{R}[x, y]$$

drei verschiedene Arten der Addition: auf den reellen Zahlen und auf den reellen Polynomringen in ein und zwei Variablen. Auf eine Unterscheidung der Symbolik kann verzichtet werden, weil einem Mathematiker anhand der Summanden klar ist, auf welchen Mengen die Operation ausgeführt und damit welche der Funktionen benutzt werden muß. Dieses *semantische Überladen* eines syntaktischen Operators ist in der Logik  $\mathcal{HOL}$  jedoch nicht möglich. Betrachten wir wieder die drei verschiedenen Additionsfunktionen, so stellen wir fest, daß sie auf Mengen operieren, deren einzelne Entitäten Objekten unterschiedlichen Typs in  $\mathcal{HOL}$  entsprechen. Damit sind auch die jeweiligen Funktionssymbole von verschiedenem Typ. Nun wurde aber mit Definition 3.4 gefordert, daß die Signatur eine disjunkte Familie von Mengen ist, womit in  $\mathcal{HOL}$  alle drei Additionsoperatoren

durch verschiedene Symbole ausgedrückt werden müssen. In  $\Omega$ -MKRP wird nun, um die Korrektheit der Logik zu gewährleisten, jeder Operation auf jedem Polynomring ein eigenes Symbol zugeordnet. So wird die Addition auf Polynomen aus  $\mathbb{R}[x_1, \dots, x_r]$ ,  $n \geq 1$  durch  $pplus-rn \in \Sigma_{\substack{((\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}}, \underbrace{\nu, \dots, \nu}_{n+1\text{-mal}})) \rightarrow (\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}})}} \in \Sigma_{\substack{((\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}}, \underbrace{\nu, \dots, \nu}_{n+1\text{-mal}})) \rightarrow (\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}})}} \in \Sigma_{\substack{((\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}}, \nu) \rightarrow (\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}})) \cdot}}$  und  $pderiv-rn, pinteg-rn \in \Sigma_{\substack{((\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}}, \nu) \rightarrow (\underbrace{\nu, \dots, \nu}_{n+1\text{-mal}})) \cdot}}$  In dieser Symbolik steht das Suffix  $rn$  für den Polynomring über dem Koeffizientenkörper  $\mathbb{R}$  in  $n$  Variablen. Damit erhalten wir als logisches Äquivalent für die als Beispiel angeführten Additionen:

$$\bullet + \in \Sigma_{(\nu, \nu) \rightarrow \nu} \quad \bullet pplus-r1 \in \Sigma_{((\nu, \nu), (\nu, \nu)) \rightarrow (\nu, \nu)} \quad \bullet pplus-r2 \in \Sigma_{((\nu, \nu, \nu), (\nu, \nu, \nu)) \rightarrow (\nu, \nu, \nu)}$$

Bei entsprechender Einschränkung des Basistyps  $\nu \in \mathcal{T}_B$  sind obige Funktionen auch über anderen Polynomringen definierbar. So wäre  $pplus-zn$  die Polynomaddition in  $\mathbb{Z}[x_1, \dots, x_n]$ ,  $n \geq 1$  und  $pplus-qn$  entsprechend in  $\mathbb{Q}[x_1, \dots, x_n]$ ,  $n \geq 1$ .

Um den Unterschied zwischen der in dieser Arbeit benutzten Schreibweise und der eigentlichen  $\Omega$ -MKRP Darstellung zu verdeutlichen, betrachten wir abschließend die binomische Formel des Theorems in Abbildung 2.2 in vereinfachter Form

$$(\lambda x. \lambda y. (x + y) \otimes \lambda x. \lambda y. (x + y)) = \lambda x. \lambda y. (x^2 + (2xy + y^2))$$

und in vollständiger  $\mathcal{HOL}$ -Repräsentation

$$\begin{aligned} (= & (p-mult-r2 \ \lambda X. \lambda Y. (+ (* 1 (* (\uparrow X 1) (\uparrow Y 0))) \\ & \quad (* 1 (* (\uparrow X 0) (\uparrow Y 1)))) \\ & \quad \lambda X. \lambda Y. (+ (* 1 (* (\uparrow X 0) (\uparrow Y 1))) \\ & \quad \quad (* 1 (* (\uparrow X 1) (\uparrow Y 0)))))) \\ & \lambda X. \lambda Y. (+ (* 1 (* (\uparrow X 2) (\uparrow Y 0))) \\ & \quad (* 2 (* (\uparrow X 1) (\uparrow Y 1))) \\ & \quad (* 1 (* (\uparrow X 0) (\uparrow Y 2)))))) \end{aligned}$$

## 4.2 $\mu\mathcal{CAS}$

Das Herz vieler in der Praxis verwendeter Computeralgebrasysteme bildet oft ein leistungsfähiger Mechanismus zur Manipulation von Polynomen. Diese Mechanismen sind einerseits sehr weit entwickelt, weil viele gute Algorithmen zur Polynommanipulation existieren, andererseits auch von zentralem Interesse, weil sich viele Berechnungen auf weitaus komplizierteren mathematischen Objekten auf Polynome zurückführen lassen. Daher ist es naheliegend, in SAPPER ein CAS mit Polynomalgorithmen zu benutzen.  $\mu\mathcal{CAS}$  ist eine Sammlung solcher Algorithmen, die in CLOS implementiert sind und teilweise vom Autor selbst entworfen wurden oder auf Schemata in [Zip93, DST88, ASS86] beruhen. In diesem Abschnitt betrachten wir zunächst die Darstellung von Polynomen in  $\mu\mathcal{CAS}$  und dann die in diesem System integrierten Algorithmen. Ein besonderer Schwerpunkt liegt dabei auf den notwendigen Erweiterungen, so daß sie in SAPPER zur Beweisplanung benutzt werden können.

### 4.2.1 Polynome in $\mu\mathcal{CAS}$

Bevor Algorithmen zur Manipulation von Polynomen für ein CAS erstellt werden können, muß grundsätzlich geklärt sein, wie Polynome repräsentiert werden bzw. wie ihre Datenstruktur aussieht. In  $\mu\mathcal{CAS}$  wird besonders darauf Wert gelegt, daß Polynome bezüglich der Zugehörigkeit zu einem Polynomring spezifiziert sind. Das bedeutet insbesondere, daß Koeffizientenkörper und Variablenanzahl festgelegt sind.

Bevor wir uns den eigentlichen Datenstrukturen zuwenden, definieren wir zunächst einige Begriffe für die Polynomdarstellung. (Vergleiche hierzu auch [Zip93], Seite 109f.)

**Definition 4.1:** Sei  $p \in \mathbb{K}[x_1, \dots, x_r]$ ,  $p = \sum_{i=1}^n \alpha_i x_1^{e_{1i}} \cdots x_r^{e_{ri}}$  ein Polynom in  $r$  Variablen mit Koeffizienten  $\alpha_i \in \mathbb{K}$ ,  $i = 1, \dots, r$ , dann heißt die Darstellung von  $p$

- *variablendicht*, wenn in jedem seiner Monome alle  $r$  Variablen vertreten sind, d.h. auch Nullexponenten werden aufgeführt; *variablenarm*, wenn sie nicht variablendicht ist.
- *rekursiv*, wenn  $p$  ein *univariates* Polynom ist, dessen Koeffizienten wieder Polynome in  $r \Leftrightarrow 1$  Variablen sind. Das bedeutet,  $p$  kann aufgefaßt werden als ein Polynom in  $\mathbb{K}[x_r][x_{r \Leftrightarrow 1}] \cdots [x_1]$ ; *expandiert*, wenn es ein *multivariates* Polynom mit Koeffizienten ausschließlich aus  $\mathbb{K}$  ist.
- *graddicht*, wenn  $p$  auch alle Monome mit Koeffizienten 0 umfaßt; *gradarm*, wenn sie nicht graddicht ist.

Zum besseren Verständnis der in Definition 4.1 eingeführten Begriffe betrachten wir das Polynom  $p = x^2y^3 + x^2z + yz^3 + yz^2 + z$ ,  $p \in \mathbb{Z}[x, y, z]$  in verschiedenen gradarmen Darstellungen.  $p_1 = x^2y^3 + x^2z + yz^3 + yz^2 + z$  ist dann in expandierter und  $p_2 = x^2(y^3 + z) + y(z^3 + z^2) + z$  in rekursiver, jeweils variablenarmer Darstellung. In den variablendichten Repräsentationen werden auch diejenigen Variablen aufgeführt, deren Exponenten 0 sind, wobei für  $p$  die expandierte Darstellung  $p_3 = x^2y^3z^0 + x^2y^0z^1 + x^0y^1z^3 + x^0y^1z^2 + x^0y^0z^1$  und  $p_4 = ((z^0)y^3 + (z^1)y^0)x^2 + ((z^3 + z^2)y + (z^1)y^0)x^0$  die rekursive ist. Man erkennt nun, daß  $p_2$  und  $p_4$  jeweils als Elemente in  $\mathbb{Z}[z][y][x]$  zu verstehen sind, also als Polynome in der Variablen  $x$ , deren Koeffizienten wiederum Polynome aus  $\mathbb{Z}[z][y]$  sind.

Betrachten wir nun im folgenden Polynome in variablendichter, expandierter Darstellung, dann ist es uns möglich, bezüglich der einzelnen Monome eine Ordnung zu definieren.

**Definition 4.2 (Monom-Ordnung):** Seien  $m = \alpha x_1^{e_1} \cdots x_r^{e_r}$  und  $n = \beta x_1^{f_1} \cdots x_r^{f_r}$  Monome in  $\mathbb{K}[x_1, \dots, x_r]$ , mit  $\alpha, \beta \in \mathbb{K}$  und  $e_i, f_i \in \mathbb{N}$ ,  $i = 1, \dots, r$ , dann ist

- $m$  exponentengleich  $n$ ,  $m =_\varepsilon n$ , wenn für alle  $i = 1, \dots, r$  gilt  $e_i = f_i$ ,
- $m$  exponentengrößer  $n$ ,  $m >_\varepsilon n$ , wenn  $e_i = f_i$  und  $e_{i+1} > f_{i+1}$  gilt für  $0 \leq i < r$ ,

ansonsten ist  $m$  exponentenkünder  $n$ ,  $m <_\varepsilon n$ .

Mit Hilfe von Definition 4.2 können die einzelnen Monome eines Polynoms so geordnet werden, daß wir immer eine Normalform für Polynome angeben können.

**Definition 4.3 (Polynom-Ordnung):** Sei  $p = \sum_{i=1}^n m_i$  ein Polynom, dann ist  $p$  in *normaler Ordnung*, wenn für seine Monome gilt:  $m_i <_\varepsilon m_{i+1}$ ,  $i = 1, \dots, i \Leftrightarrow 1$

Für die konkrete Datenstruktur von Polynomen in  $\mu\mathcal{CAS}$  hat sich der Autor für eine gradarme, variablendichte, expandierte und normalgeordnete Darstellung entschieden. Variablendichte, expandierte Darstellung ermöglicht direkte Exponentenvergleiche und damit effiziente Algorithmen; durch gradarme Repräsentation wird der Speicherverbrauch auch bei schwachbesetzten multivariaten Polynomen in Grenzen gehalten. Implementiert ist diese Darstellung durch eine gemäß Definition 4.3 geordnete Liste von Monomen. Die Monome selbst sind wiederum Listen der Form (Koeffizient Exponent<sub>1</sub> ... Exponent<sub>r</sub>) für Polynome in  $r$  Variablen. Damit ergibt sich allgemein für ein Polynom  $p \in \mathbb{K}[x_1, \dots, x_r]$ ,  $p = \sum_{i=1}^n \alpha_i x_1^{e_{1i}} \cdots x_r^{e_{ri}}$  folgende Listenrepräsentation:

$$p: ((\alpha_n e_{1n} \cdots e_{rn}) (\alpha_{n \Leftrightarrow 1} e_{1n-1} \cdots e_{rn-1}) \cdots (\alpha_1 e_{11} \cdots e_{r1}))$$

Unser Polynom  $p = x^2y^3 + x^2z + yz^3 + yz^2 + z$  entspricht in  $\mu\mathcal{CAS}$  also einer Liste der Form ((1 2 3 0) (1 2 0 1) (1 0 1 3) (1 0 1 2) (1 0 0 1))

```

(defclass ca+polynomial ()
  ((var-number :initarg :var-number
               :initform 1
               :reader ca~poly-var-number)
   (data :initarg :data-list
         :initform nil
         :accessor ca~poly-data)))

```

Abbildung 4.2: Die CLOS-Definition der Oberklasse von Polynomen in  $\mu\mathcal{CAS}$

Zusätzlich zu dieser Datenliste enthält jedes Polynom explizit die Variablenanzahl beigeordnet. Abbildung 4.2 zeigt die allgemeine Definition der CLOS-Klasse für Polynome in  $\mu\mathcal{CAS}$ , deren Attribute sowohl die Liste als auch die Variablenanzahl sind. (Zu den allgemeinen Begriffen des objektorientierten Programmierens siehe auch [CN94], zu Begriffen aus CLOS [Kee89].)

```

(defclass ca+nat-polynomial (ca+polynomial)
  ())
(defclass ca+int-polynomial (ca+polynomial)
  ())
(defclass ca+rat-polynomial (ca+polynomial)
  ())
(defclass ca+real-polynomial (ca+polynomial)
  ())
(defclass ca+complex-polynomial (ca+polynomial)
  ())

```

Abbildung 4.3: Die CLOS-Definition der Unterklassen von `ca+polynomial`

Um Polynome auch nach ihren Koeffizienten spezifizieren zu können, werden Unterklassen für Polynome mit verschiedenen Koeffizientenkörpern zur Verfügung gestellt. Im derzeitigen Stand der Implementation von  $\mu\mathcal{CAS}$  existieren Klassen für Polynome aus den natürlichen, ganzen, rationalen, reellen und komplexen Zahlen (vergleiche Abbildung 4.3).

## 4.2.2 Algorithmen in $\mu\mathcal{CAS}$

$\mu\mathcal{CAS}$  stellt die grundlegenden und einfachen Algorithmen zur Manipulation von Polynomen zur Verfügung. Dabei handelt es sich hauptsächlich um die in Abschnitt 4.1.2 vorgestellten Operationen auf Polynomen; darüber hinaus werden aber auch noch einige andere Funktionen bereitgestellt, wie z.B. Operationen zur Nullstellenbestimmung oder zur Auswertung von Polynomen. Eine Besonderheit der Algorithmen ist eine Erweiterung, so daß

sie genügend Informationen über ihre einzelnen Rechenschritte liefern und es damit ermöglichen, einen vollständigen ND-Beweis für die jeweiligen Berechnungen zu konstruieren. Im folgenden sollen nur die notwendigen Erweiterungen im Vordergrund stehen, während die Algorithmen selbst nur eine untergeordnete Rolle spielen.

Die *Erweiterungen* der Algorithmen von  $\mu CAS$  wurde nach zwei Kriterien vorgenommen. Zum einen sollen die Algorithmen ausreichende Informationen über ihr Rechenverhalten liefern, so daß daraus ein vollständiger Beweisplan konstruiert werden kann. Hierfür muß jeder signifikante Rechenschritt des Algorithmus, also jede Manipulation der Datenstruktur, auf der der Algorithmus arbeitet, protokolliert werden. Zum anderen soll das Rechenverhalten der Algorithmen durch die vorgenommenen Erweiterungen nicht verändert werden. Das bedeutet, daß außer durch Zusatzausgaben oder einige zusätzliche mitverwaltete Variablen das Schema eines Algorithmus nicht modifiziert werden soll.

```
(defgeneric ca~polynomial-addition (poly1 poly2)
  (:method ((poly1 ca+polynomial) (poly2 ca+polynomial))
    (CA=OUTPUT-TACTIC 'POLYNOMIAL-ADDITION)
    (ca~create-polynomial
     (ca=padd-recursive (ca~poly-data poly1) (ca~poly-data poly2))
     :field (ca~get-poly-field poly1)
     :var (ca~poly-var-number poly1))))

(defun ca=padd-recursive (data1 data2)
  (cond ((null data1) data2)
        ((null data2) data1)
        ((ca=exp-equal (ca=exps (first data1)) (ca=exps (first data2)))
         (CA=OUTPUT-TACTIC 'MONOMIAL-ADDITION)
         (append (list
                  (append (list
                           (+ (ca=coeff (first data1))
                              (ca=coeff (first data2))))
                          (ca=exps (first data1))))
                  (ca=padd-recursive (rest data1) (rest data2))))
        ((ca=exp-greater (ca=exps (first data1)) (ca=exps (first data2)))
         (CA=OUTPUT-TACTIC 'POP-FIRST)
         (append (list (first data1))
                  (ca=padd-recursive (rest data1) data2)))
        (t (CA=OUTPUT-TACTIC 'POP-SECOND)
            (append (list (first data2))
                     (ca=padd-recursive data1 (rest data2))))))
```

Abbildung 4.4: Rekursiver Algorithmus zur Polynomaddition in CLOS

Als Beispiel betrachten wir die Polynomaddition in  $\mu CAS$ , deren CLOS-Code in Ab-

bildung 4.4 zu sehen ist. Der Algorithmus selbst ist in zwei Funktionen unterteilt: In eine Methode für die polymorphe Funktion `ca~polynomial-addition`, die auf zwei Objekten des Typs `ca+polynomial`, also zwei Polynomen (siehe Abbildung 4.2), arbeitet, und eine Funktion `ca=padd-recursive`, die rekursiv mit den Datenlisten der beiden Argumente von `ca~polynomial-addition`, also den einzelnen Monomen, rechnet.

Wird die Polynomaddition mit zwei Argumenten  $p = \sum_{i=1}^r m_i$  und  $q = \sum_{i=1}^s n_i$  aufgerufen, werden zunächst die Datenlisten von  $p$  und  $q$  an die rekursive Funktion übergeben. Bei jedem Lauf von `ca=padd-recursive` werden die Exponentenlisten der jeweils ersten Monome von  $p$  und  $q$  verglichen. Überlegen wir uns den Algorithmus für die erste Rekursionstiefe, also für die Monome  $m_1$  und  $n_1$ , dann geschieht Folgendes: Gilt  $m_1 =_{\varepsilon} n_1$ , so werden die Monome addiert und `ca=padd-recursive` mit  $\sum_{i=2}^r m_i$  und  $\sum_{i=2}^s n_i$  (bzw. den entsprechenden Datenlisten) erneut aufgerufen. Gilt  $m_1 >_{\varepsilon} n_1$  (bzw.  $m_1 <_{\varepsilon} n_1$ ), so sind  $\sum_{i=2}^r m_i$  und  $\sum_{i=1}^s n_i$  ( $\sum_{i=1}^r m_i$  und  $\sum_{i=2}^s n_i$ ) die Argumente für den rekursiven Aufruf von `ca=padd-recursive`. Dieser Algorithmus wird so lange durchgeführt, bis eine der beiden Listen leer ist. Da wir in Abschnitt 4.2.1 die normale Ordnung unserer Polynome gefordert haben, ist das Ergebnispolynom des Algorithmus erneut in normaler Ordnung.

Die Erweiterung für die Beweisplanung erfolgt durch Einfügen von Aufrufen an die Funktion `ca=output-tactic`, deren Aufgabe es ist, einen Taktiknamen und zusätzliche Argumente auszugeben. Beim vorliegenden Algorithmus ist aber die Ausgabe bloßer Taktiknamen ausreichend. Bei der Polynomaddition werden hierfür vier verschiedene Taktiken benutzt: `POLYNOMIAL-ADDITION`, `POP-FIRST`, `POP-SECOND` und `MONOMIAL-ADDITION`. (Eine Aufstellung dieser Taktiken befindet sich in Anhang B) Dabei entspricht die erste Taktik der Addition der beiden Summen, als die Polynome aufgefaßt werden können, während die restlichen drei die Operationen auf Monomen beschreiben.

Rechenschritt	Taktikausgabe
$(3x^2 + (2x + 5)) + (5x^3 + (x + 2))$	(polynomial-addition)
$(5x^3 + ((3x^2 + (2x + 5)) + (x + 2)))$	(pop-second)
$(5x^3 + (3x^2 + ((2x + 5) + (x + 2))))$	(pop-first)
$(5x^3 + (3x^2 + (3x + (5 + 2))))$	(monomial-addition)
$(5x^3 + (3x^2 + (3x + 7)))$	(monomial-addition)

Abbildung 4.5: Taktikausgabe der  $\mu$ CAS-Berechnung von (4.4)

Zur Erläuterung der Rechenschritte und damit der Bedeutung der einzelnen Taktiken untersuchen wir das Verhalten des Algorithmus anhand der einfachen Polynomaddition

$$p \oplus q = (3x^2 + (2x + 5)) \oplus (5x^3 + (x + 2)). \quad (4.4)$$

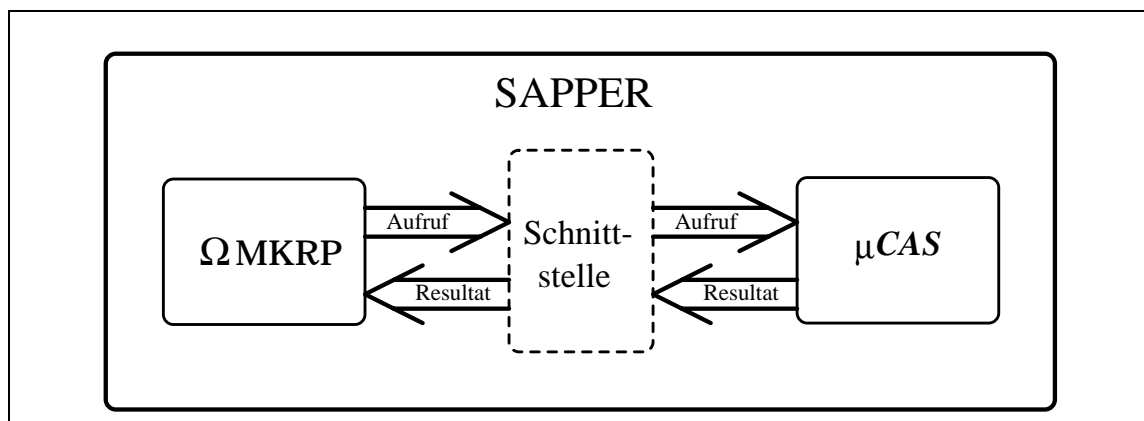


Abbildung 4.6: Die Systemarchitektur von SAPPER

Die Rechenschritte und die dazu korrespondierenden Ausgaben der `ca=output-tactic`-Funktion sind angegeben in Abbildung 4.5. Zunächst werden die beiden Polynome  $p$  und  $q$  zu einem Polynom zusammengefügt, was der `POLYNOMIAL-ADDITION`-Taktik entspricht. Dann wird das kubische Monom in  $q$  – das zweite Argument der Polynomaddition – an den Anfang der Summe gestellt. Diese Information wird mit dem Schlüsselwort `POP-SECOND` protokolliert. Ebenso entspricht die `POP-FIRST`-Taktik einer Umordnung des ersten Summanden im ersten Argument, also dem quadratischen Monom in  $p$ . Schließlich werden die restlichen Monome in zwei Schritten addiert. Dies wird jeweils mit `MONOMIAL-ADDITION` protokolliert.

### 4.3 Schnittstelle

Das SAPPER-System kann als eine generische Schnittstelle angesehen werden, das  $\Omega$ -MKRP mit einem oder mehreren CAS verbindet. Dabei bedeutet generisch, daß das System erweiterbar ist, ohne daß Änderungen an den bereits bestehenden Funktionen vorgenommen werden müssen. Im jetzigen Stadium der Implementation stellt SAPPER im wesentlichen eine Verbindung zwischen  $\Omega$ -MKRP und  $\mu$ CAS dar und wird im folgenden auch anhand dieser beiden Systemteile erläutert.

Der grundlegende Aspekt für die Systemintegration zwischen  $\Omega$ -MKRP und  $\mu$ CAS ist eine Master-Slave-Beziehung der beiden Systeme, die durch eine Brücke, die Schnittstelle, realisiert wird. Die Arbeitsweise ist schematisch in Abbildung 4.6 dargestellt und entspricht dem generellen Paradigma für Systemkombinationen aus [CMP91]. Von der technischen Seite können  $\Omega$ -MKRP das CAS als zwei voneinander unabhängige Prozesse gesehen werden, von denen der erste Anfragen an letzteren stellen kann und dieser Resultate zurückgibt. Dabei läuft die Kommunikation nicht direkt zwischen den beiden Systemen, sondern über

einen dritten, ebenfalls unabhängigen Prozeß, den der zwischengeschalteten Schnittstelle. Deren Rolle ist dabei, das Übermitteln von Nachrichten zwischen beiden Systemen zu automatisieren, indem sie Ausgaben des einen Systems in Daten des anderen umwandelt. Ein Aufruf von  $\mu\text{CAS}$  in  $\Omega\text{-MKRP}$  wird also über die Schnittstelle an dieses weitergeleitet. Der Prozeß von  $\Omega\text{-MKRP}$  befindet sich dann so lange im Wartezustand, bis das Resultat einer Berechnung von  $\mu\text{CAS}$  über die Schnittstelle zurückgegeben wird. Diese Handhabung der Prozesse ist durch die jetzige Version von  $\Omega\text{-MKRP}$  bestimmt und könnte in einer späteren Implementation durchaus geändert werden.

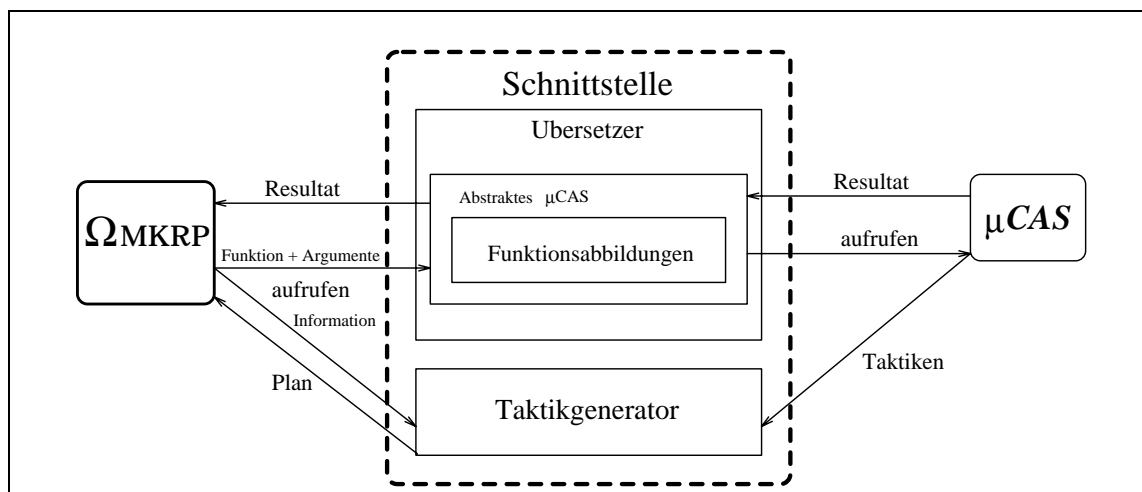
Um nun die inhaltlichen Designentscheidungen für die einzelnen Komponenten der Schnittstelle zu motivieren, betrachten wir zunächst die an diese gestellten Anforderungen, von denen bereits einige in Kapitel 2.4 aufgeführt wurden:

Selbstverständlich soll uns die Schnittstelle ermöglichen, Beweispläne aus CAS-Berechnungen nach  $\Omega\text{-MKRP}$  zu importieren. Hierfür sollen  $\Omega\text{-MKRP}$  und CAS so verbunden werden, daß weder die Logik in  $\Omega\text{-MKRP}$  noch die Schemata der Computeralgebra-Algorithmen geändert werden müssen. Darüber hinaus sollen die beteiligten Systeme ihre Eigenständigkeit erhalten. (Man vergleiche hierzu gegensätzliche Ansätze wie zum Beispiel in [GPT94, HC95], bei denen Logik und Kontrolleinheiten der integrierten Systeme verändert werden müssen.) Um nicht bei jeder CAS-Berechnung einen vollständigen Beweis für diese zu erhalten – zum Beispiel während der Beweissuche – soll es auch möglich sein, nur Resultate des CAS zu benutzen. Das CAS soll also in einem sogenannten *Verbose-Mode*, in dem es ausreichende Informationen über die Berechnungen liefert, und daneben als *Black-Box*, also nur als Lieferant von Ergebnissen, eingesetzt werden können. Hierfür könnten wir, wenn nötig, sogar verschiedene Algorithmen für die gleiche Berechnung einsetzen und diese eventuell auch aus verschiedenen CAS. Also sollten auch mehrere CAS gleichzeitig und parallel in SAPPER integrierbar sein. Schließlich soll es möglich sein, dem System neue Algorithmen und CAS hinzuzufügen, ohne die bestehende Implementation verändern zu müssen, wofür die Funktionalität der Schnittstelle möglichst allgemein und generisch gehalten werden muß.

SAPPERS Schnittstelle erfüllt all diese Anforderungen. Abbildung 4.7 ist die schematische Darstellung der Schnittstelle zwischen  $\Omega\text{-MKRP}$  und  $\mu\text{CAS}$ . In ihr ist zu erkennen, daß die Schnittstelle in zwei voneinander unabhängige Module eingeteilt werden kann:

- den *Übersetzer*, der Syntaxübersetzungen zwischen  $\Omega\text{-MKRP}$  und  $\mu\text{CAS}$  in beiden Richtungen ausführt (siehe Abschnitt 4.3.1),
- und den *Taktikgenerator*, der die Zusatzinformationen eines  $\mu\text{CAS}$ -Algorithmus in  $\Omega\text{-MKRP}$  Beweispläne umwandelt (siehe Abschnitt 4.3.2).

In den folgenden Abschnitten werden diese beiden Module näher beschrieben, wobei darauf verwiesen wird, wie diese Realisierung den oben spezifizierten Anforderungen gerecht wird.

Abbildung 4.7: Die Schnittstelle zwischen  $\Omega$ -MKRP und  $\mu$ CAS

### 4.3.1 Übersetzer

Die Übersetzungseinheit der Schnittstelle ist zuständig für die *Syntaxtransformation* zwischen den angeschlossenen Systemen<sup>2</sup>, der Deduktionskomponente einerseits und der Computeralgebra-Komponente auf der anderen Seite. Darüber hinaus wird aus dem Übersetzer heraus das eigentliche CAS aufgerufen. Um die Verbindung zu  $\mu$ CAS zu spezifizieren, wird ein sogenanntes *abstraktes CAS* definiert, in dem sowohl die Syntaxtransformationen als auch eine Funktion zum Aufruf von  $\mu$ CAS angegeben sind. Abbildung 4.8 zeigt die Definition eines solchen abstrakten CAS für  $\mu$ CAS.

```
(ca~defsystem :myCAS
  (translations
    (p-plus-r1 (ca~polynomial-addition (:realpoly :realpoly :realpoly)))
    (p-plus-r2 (ca~polynomial-addition (:realpoly :realpoly :realpoly)))
    (p-mult-r1 (ca~polynomial-multiplication (:realpoly :realpoly :realpoly)))
    (p-mult-r2 (ca~polynomial-multiplication (:realpoly :realpoly :realpoly)))
    (p-deriv-r1 (ca~differentiate-polynomial (:realpoly :nat :realpoly)))
    (p-deriv-r2 (ca~differentiate-polynomial (:realpoly :nat :realpoly)))
    (p-integ-r1 (ca~integrate-polynomial (:realpoly :nat :realpoly)))
    (p-integ-r2 (ca~integrate-polynomial (:realpoly :nat :realpoly)))
  )
  (call eval))
```

Abbildung 4.8: Definition eines abstrakten CAS für  $\mu$ CAS

<sup>2</sup>Diese speziellen Syntaxtransformationen könnten in Zukunft durch eine einheitliche Übersetzung in das OpenMath-Protokoll [AvLS95] ersetzt werden.

Darin wird zunächst der Name festgelegt, mit welchem das CAS aus  $\Omega_{\text{MKRP}}$  heraus angesprochen wird (im Beispiel `:mycas`); dieser kann, muß aber nicht, dem Namen des tatsächlichen CAS entsprechen. Das Schlüsselwort `translations` leitet eine Reihe von Syntaxabbildungen ein. Diese sind jeweils von der Form:

```
(omegafunction (casfunction (argtype1...argtypen resulttype)))
```

Dies bedeutet, daß eine  $\Omega_{\text{MKRP}}$ -Funktion einer  $\mu\text{CAS}$ -Funktion und ihre  $n$  Argumente entsprechenden  $\mu\text{CAS}$ -Objekten vom Typ `argtype1, ..., argtypen` zugeordnet werden. Schließlich gibt `resulttype` den Typ des  $\mu\text{CAS}$ -Objekts an, das als Resultat erwartet wird. Folglich entspricht in Abbildung 4.8 zum Beispiel die Zeile

```
(p-plus-r1 (ca~polynomial-addition (:realpoly :realpoly :realpoly)))
```

einer Abbildung der logischen  $\Omega_{\text{MKRP}}$ -Funktion `p-plus-r1` auf die  $\mu\text{CAS}$ -Funktion `ca~polynomial-addition`, deren beide Argumente reelle Polynome sind und die ein ebensolches als Ergebnis liefert.

Die letzte Komponente eines abstrakten CAS besteht aus dem Schlüsselwort `call` und einer Lisp-Funktion, mit deren Hilfe das eigentliche CAS aufgerufen wird. Im Falle von  $\mu\text{CAS}$  handelt es sich dabei um die COMMON LISP-Funktion `eval`, da  $\mu\text{CAS}$  im gleichen Lisp-Prozeß wie SAPPER läuft.

Um die im abstrakten CAS spezifizierten Übersetzungen auf Objektebene zu realisieren, stehen in der Übersetzungseinheit zwei *polymorphe Funktionen* zur Verfügung:

**ca~build-object** übersetzt  $\Omega_{\text{MKRP}}$ -Syntax in  $\mu\text{CAS}$ -Objekte

**ca~rebuild-object** transformiert  $\mu\text{CAS}$ -Syntax in  $\Omega_{\text{MKRP}}$ -Syntax.

Die Zusammenarbeit der einzelnen Objekte und Funktionen läuft nach dem in Abbildung 4.9 angegebenen Schema. Nachdem von  $\Omega_{\text{MKRP}}$   $\mu\text{CAS}$  zur Vereinfachung einer bestimmten Funktion aufgerufen wurde, wird deren computeralgebraisches Äquivalent in den Funktionsabbildungen des entsprechenden abstrakten CAS nachgesehen. Mit diesem Wissen werden die Argumente der  $\Omega_{\text{MKRP}}$ -Funktion durch `ca~build-object` in  $\mu\text{CAS}$ -Objekte übersetzt und der vollständige Funktionsaufruf an das CAS übergeben. Nachdem die Berechnung in  $\mu\text{CAS}$  ein Resultat liefert, wird dieses anhand des in der Zuordnungstabelle `translations` angegebenen Typs durch `ca~rebuild-object` in  $\Omega_{\text{MKRP}}$ -Syntax transformiert und an  $\Omega_{\text{MKRP}}$  schließlich zurückgegeben.

Im Falle der beiden Funktionen `ca~build-object` und `ca~rebuild-object` bedeutet polymorph, daß bezüglich verschiedener Objekte verschiedene Methoden einer Funktion implementiert sind. Abbildung 4.10 zeigt dies anhand zweier unterschiedlicher Methoden der `ca~build-object` Funktion: die erste Methode spezifiziert, wie reelle Polynome, die

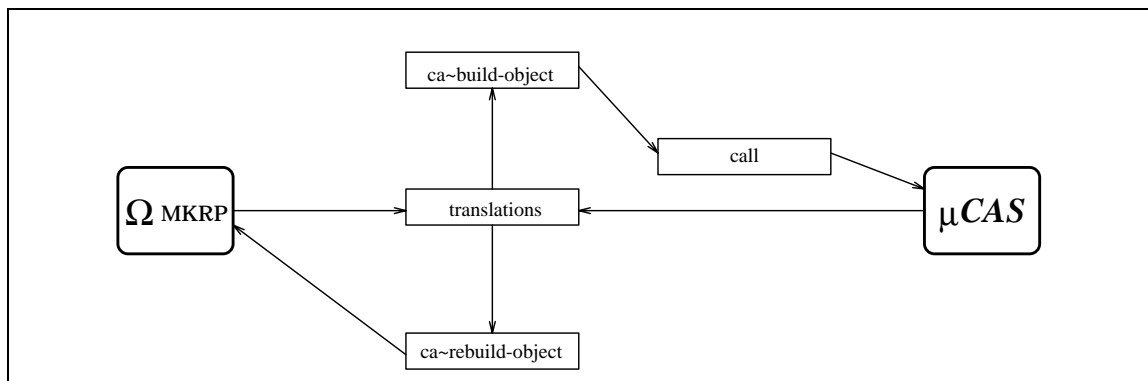


Abbildung 4.9: Datenflußdiagramm für die Übersetzungseinheit

zweite wie natürliche Zahlen in  $\mu CAS$ -Syntax umgewandelt werden. Dabei wird mit Hilfe der letzten beiden von drei Argumenten der Funktion festgelegt, wann die entsprechende Methode angewandt wird. Die eigentlichen Syntaxtransformationen werden bei beiden Methoden jeweils in Hilfsfunktionen (`ca=build-mycas-poly` und `ca=build-mycas-number`) durchgeführt.

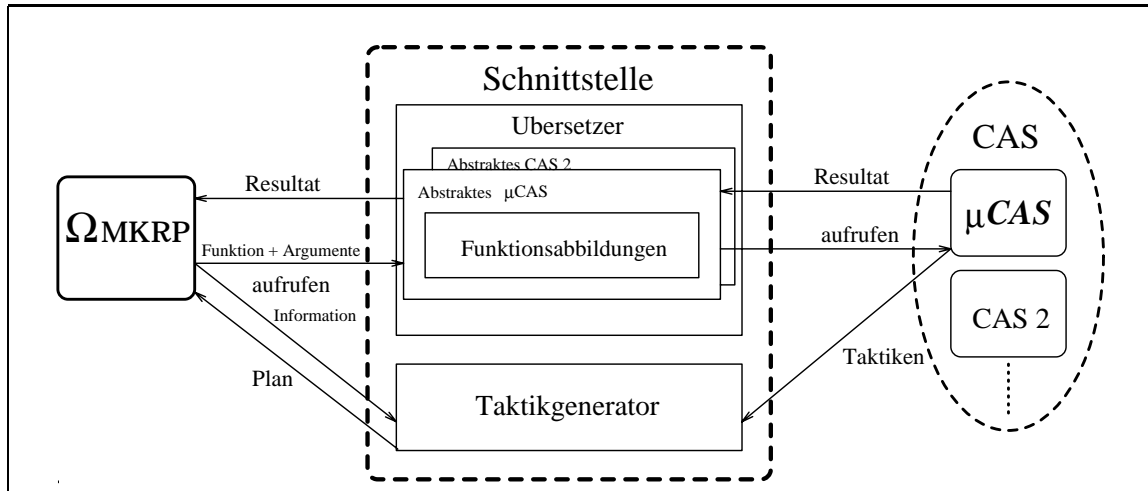
```

(defmethod ca~build-object (term (object (eql :realpoly)) (system (eql :mycas)))
  (ca=build-mycas-poly :real term))

(defmethod ca~build-object (term (object (eql :nat)) (system (eql :mycas)))
  (let ((number (ca=build-mycas-number term)))
    (unless (typep number '(integer 1))
      (error number))
    number))
  
```

Abbildung 4.10: Methoden für die polymorphe Funktion `ca~build-object`

Der Begriff der polymorphen Funktion ist ein wesentlicher Bestandteil des Paradigmas der *objektorientierten Programmierung* [CN94, KdRB91], der es ermöglicht, für bereits bestehende Funktionen neue Methoden bezüglich neuer Objekte zu definieren. Ebenso sind abstrakte CAS nichts weiter als Objekte, also konkrete Instanzen einer Klasse. Damit ist es nun möglich, den Übersetzungsteil zu erweitern, indem man neue abstrakte CAS definiert und neue Methoden für die beiden polymorphen Funktionen. Dafür muß an der bestehenden Implementation nichts verändert werden. Darüber hinaus können durch Definition mehrerer abstrakter CAS auch mehrere CAS gleichzeitig in SAPPER integriert werden. (Zu dieser erweiterten Systemarchitektur siehe auch Abbildung 4.11)

Abbildung 4.11: Schnittstelle zwischen  $\Omega$ -MKRP und mehreren CAS

### 4.3.2 Taktikgenerator

Der Taktikgenerator der Schnittstelle stellt die Mechanismen zur Verfügung, um Zusatzausgaben der Computeralgebra-Algorithmen in einen für  $\Omega$ -MKRP verwertbaren Beweisplan zu transformieren (vergleiche hierzu Abbildungen 4.7 und 4.12). Er ist in zwei miteinander kommunizierende Einheiten unterteilt: in den Generator selbst und eine Kontrolle, die wichtige zusätzliche Beweisinformationen verwaltet. Hierzu zählen zum Beispiel die Zeile, auf die das CAS angewandt wird, oder die Termposition der vom CAS zu vereinfachenden Funktion. Diese Informationen werden bei Aufruf eines CAS in Verbose-Mode von  $\Omega$ -MKRP übergeben. Demgegenüber erhält der Generator seine Informationen vom jeweiligen Computeralgebra-Algorithmus, der gemäß seinem Rechenverhalten Namen von Taktiken übergibt. Im Generator werden diese protokolliert und entsprechend des Kontrollwissens bearbeitet, was bedeutet, daß die übergebene Taktik in eine Sequenz von Taktiken der *Faktenebene* expandiert wird. Infolgedessen kann es auch zu Änderungen innerhalb der Kontrolle kommen (zum Beispiel müssen etwaige Termpositionen aktualisiert werden). So entsteht während des Laufs des Algorithmus ein Beweisplan als eine Sequenz von Taktiken, der nach Abschluß der Berechnungen an  $\Omega$ -MKRP zurückgegeben wird.

Für die Planerstellung selbst stehen dem Generator zwei Arten von Taktiken zur Verfügung: *einfache*, die die Anwendung einer einzelnen Hypothese im Beweis beschreiben (dies ist die Faktenebene [Hua94c]), und *komplexe*, die einem einzelnen Berechnungsschritt des Computeralgebra-Algorithmus entsprechen. Diese komplexen Taktiken sind Kombinationen von einfachen Taktiken mit Hilfe von Taktikalen (siehe Kapitel 3.3.2), also bezüglich

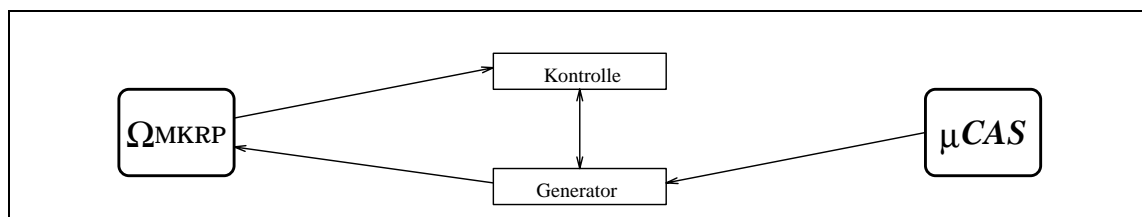


Abbildung 4.12: Datenflußdiagramm für den Taktikgenerator

bestimmter Bedingungen, die mit Hilfe der Kontrolle bestimmt werden. Damit erhalten die Beweispläne im Taktikgenerator eine hierarchische Form, bestehend aus zwei Ebenen: die der komplexen Taktiken und die der Faktenebene.

Zum besseren Verständnis dieses Punktes greifen wir wieder zurück auf die Polynomaddition (4.4) aus Abschnitt 4.2.2. In diesem Beispiel erhalten wir als Beweisplan auf der Ebene der komplexen Taktiken: (POLYNOMIAL-ADDITION, POP-FIRST, POP-SECOND, MONOMIAL-ADDITION, MONOMIAL-ADDITION). Dieser Plan expandiert zu dem in Abbildung 4.13 dargestellten Plan in einfachen Taktiken, also auf der Faktenebene. Hierbei entsprechen die einzelnen einfachen Taktiken der Anwendung der Polynomaddition (P+), der Kommutativität der Addition (C+), der Assoziativität der Addition (A+) und der Monomaddition (M+).

komplexe Taktik	Sequenz von einfachen Taktiken
(polynomial-addition)	P+
(pop-second)	A+, C+, A+
(pop-first)	A+
(monomial-addition)	A+, C+, A+, M+, A+
(monomial-addition)	M+

Abbildung 4.13: Expansion der komplexen Taktiken für die Berechnung von (4.4)

Darüber hinaus ist zu beachten, daß die einzelnen Taktiken keine Argumente benötigen, da etwaige Zusatzinformationen von der Kontrolle verwaltet werden. So wird im Beispiel die Termposition der Addition verwaltet, bezüglich welcher die entsprechende Taktik ausgeführt wird. Anhand des vorliegenden Terms wird dann das unterschiedliche Verhalten der MONOMIAL-ADDITION-Taktik bestimmt (zur formalen Definition dieser Taktik siehe Anhang B). Dies bedeutet für den Rechenschritt

$$(5x^3 + (3x^2 + ((2x + 5) \boxed{+} (x + 2))))),$$

daß der Taktikgenerator MONOMIAL-ADDITION bezüglich des eingerahmten + expandiert, was zu der Sequenz (A+, C+, A+, M+, A+) von einfachen Taktiken führt. Im Schritt

$$(5x^3 + (3x^2 + (3x + (5 \boxed{+} 2))))$$

dagegen wird MONOMIAL-ADDITION auf  $(5 + 2)$  angewendet und expandiert zu  $M+$ .

Bei einer Erweiterung von SAPPER um neue Algorithmen mit Verbose-Mode muß natürlich auch der Taktikgenerator expandiert werden können. Hierzu ist es möglich, neue einfache und komplexe Taktiken hinzuzufügen.

## 4.4 Arbeitsweise des Systems

Um die Arbeitsweise des gesamten SAPPER-Systems erklären zu können, ist es notwendig, das Zusammenspiel der einzelnen Module der Schnittstelle zu berücksichtigen. Aus diesem Grunde betrachten wir in diesem Abschnitt zunächst die Datenübertragung innerhalb der ganzen Schnittstelle anhand von Abbildung 4.14. Anschließend werden die verschiedenen Einsatzmöglichkeiten von SAPPER anhand einer Beispielsitzung in  $\Omega$ -MKRP exemplarisch vorgeführt.

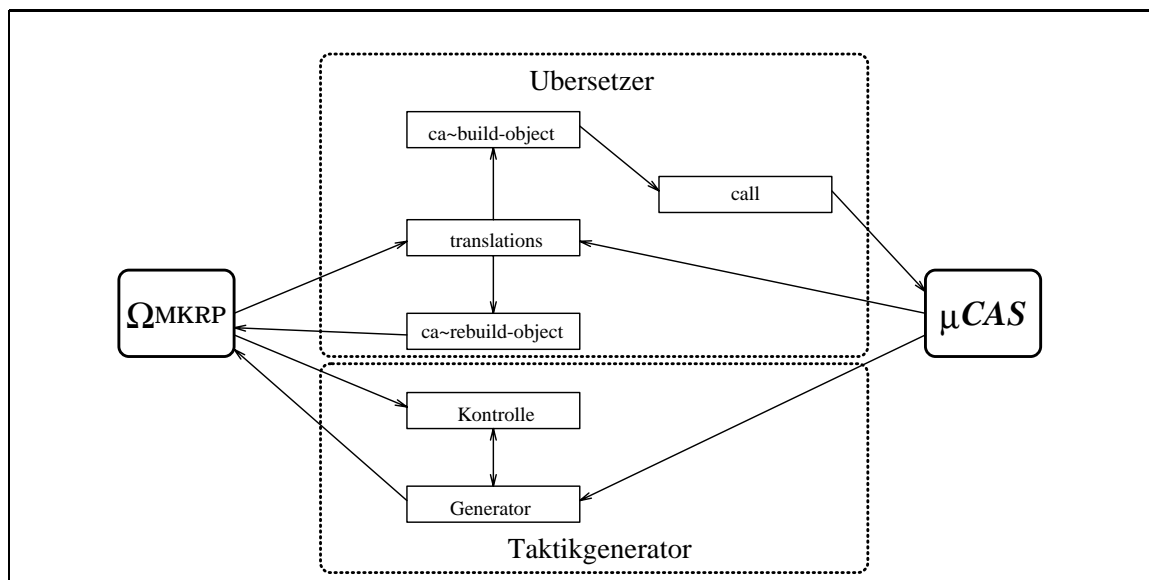


Abbildung 4.14: Datenflußdiagramm für das SAPPER-System

Das Datenflußdiagramm für die Schnittstelle in Abbildung 4.14 zeigt, daß nur im Übersetzungsteil Daten zwischen  $\Omega$ -MKRP und  $\mu$ CAS in beiden Richtungen fließen. Dagegen transportiert der Taktikgenerator nur Daten von  $\mu$ CAS nach  $\Omega$ -MKRP. Dies bedeutet, daß nur im Übersetzungsteil Wissen darüber vorhanden sein muß, mit welcher Syntax  $\mu$ CAS anzusprechen ist. Damit ist der Taktikgenerator unabhängig von dem jeweils benutzten CAS. Obendrein genügt die Funktionalität des Übersetzers, um ein CAS ausschließlich als Black-Box in SAPPER zu integrieren.

Zur Demonstration der beiden unterschiedlichen Verwendungsarten von  $\mu$ CAS in

SAPPER betrachten wir erneut den Beweis für die einfache Polynomaddition

$$\text{THM.} \quad \vdash (3x^2 + (2x + 5)) \oplus (5x^3 + (x + 2)) = (5x^3 + (3x^2 + (3x + 7))) \quad (\text{Open})$$

Die Abbildungen 4.15 und 4.16 zeigen Aufrufe von  $\mu\mathcal{CAS}$  in  $\Omega\text{-MKRP}$  für die Polynomaddition in unserem Theorem. Dabei werden jeweils allgemein mit `CALL-CAS` ein CAS-Aufruf bezüglich der Zeile `THM` und der Termposition (1) (dies ist die Position der Operation  $\oplus$  in der Formel von `THM`) spezifiziert. Als CAS benutzen wir jeweils  $\mu\mathcal{CAS}$ . Der einzige Unterschied zwischen den beiden Aufrufen ist die Benutzung des `Verbose-Modes`.

```
OMEGA: CALL-CAS
LINE (NDLINE) A line with term suitable for a CAS: THM
POSITION (INTEGER-LIST) The position of this term: (1)
SYSTEM (SYMBOL) The name of a CAS: MYCAS
FLAG (BOOLEAN) Switch on Verbose Mode: NO
```

Abbildung 4.15: Aufruf von  $\mu\mathcal{CAS}$  als Black-Box-System

```
OMEGA: CALL-CAS
LINE (NDLINE) A line with term suitable for a CAS: THM
POSITION (INTEGER-LIST) The position of this term: (1)
SYSTEM (SYMBOL) The name of a CAS: MYCAS
FLAG (BOOLEAN) Switch on Verbose Mode: YES
```

Abbildung 4.16: Aufruf von  $\mu\mathcal{CAS}$  als Beweisplaner

Beim ersten Aufruf wird auf diesen verzichtet, und wir erhalten einen einzeiligen Beweis für `THM`, dargestellt in Abbildung 4.17. Dabei wird ausschließlich das Resultat der Berechnungen von  $\mu\mathcal{CAS}$  und damit nur die Funktionalität der Übersetzungseinheit benutzt.

$$\begin{array}{ll} \text{L1.} & \vdash \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7))) \quad (=1) \\ \text{THM.} & \vdash (\lambda x_{\bullet}(3x^2 + (2x + 5)) \oplus \lambda x_{\bullet}(5x^3 + (x + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7))) \quad (\text{CAS L1}) \end{array}$$

Abbildung 4.17: Der Beweis von `THM` durch  $\mu\mathcal{CAS}$  als Black-Box-System

Der zweite Aufruf im `Verbose-Mode` dagegen erzeugt einen `ND-Beweis` für das Theorem mit der Ausgabe des Taktikgenerators, die dem Faktenebenenbeweis aus Abbildung 4.18 entspricht. Auf die Darstellung des ganzen `ND-Beweises`, der 47 Zeilen umfaßt, wurde hier verzichtet.

THM.	$\vdash (\lambda x_{\bullet}(3x^2 + (2x + 5)) \oplus \lambda x_{\bullet}(5x^3 + (x + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L3,L4)
L3.	$\vdash (\lambda x_{\bullet}(3x^2 + (2x + 5)) \oplus \lambda x_{\bullet}(5x^3 + (x + 2))) =$ $\lambda x_{\bullet}((3x^2 + (2x + 5)) + (5x^3 + (x + 2)))$	(P+)
L4.	$\vdash \lambda x_{\bullet}((3x^2 + (2x + 5)) + (5x^3 + (x + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L7,L8)
L7.	$\vdash (((3x^2 + (2x + 5)) + 5x^3) + (x + 2)) = ((3x^2 + (2x + 5)) + (5x^3 + (x + 2)))$	(A+)
L8.	$\vdash \lambda x_{\bullet}(((3x^2 + (2x + 5)) + 5x^3) + (x + 2)) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L10,L11)
L10.	$\vdash ((3x^2 + (2x + 5)) + 5x^3) = (5x^3 + (3x^2 + (2x + 5)))$	(C+)
L11.	$\vdash \lambda x_{\bullet}((5x^3 + (3x^2 + (2x + 5))) + (x + 2)) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L14,L15)
L14.	$\vdash ((5x^3 + (3x^2 + (2x + 5))) + (x + 2)) = (5x^3 + ((3x^2 + (2x + 5)) + (x + 2)))$	(A+)
L15.	$\vdash \lambda x_{\bullet}(5x^3 + ((3x^2 + (2x + 5)) + (x + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L18,L19)
L18.	$\vdash ((3x^2 + (2x + 5)) + (x + 2)) = (3x^2 + ((2x + 5) + (x + 2)))$	(A+)
L19.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + ((2x + 5) + (x + 2)))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L22,L23)
L22.	$\vdash (((2x + 5) + x) + 2) = ((2x + 5) + (x + 2))$	(A+)
L23.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + (((2x + 5) + x) + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L25,L26)
L25.	$\vdash ((2x + 5) + x) = (x + (2x + 5))$	(C+)
L26.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + ((x + (2x + 5)) + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L29,L30)
L29.	$\vdash ((x + 2x) + 5) = (x + (2x + 5))$	(A+)
L30.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + (((x + 2x) + 5) + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L35,L36)
L35.	$\vdash (x + 2x) = 3x$	(M+)
L36.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + ((3x + 5) + 2))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L39,L40)
L39.	$\vdash ((3x + 5) + 2) = (3x + (5 + 2))$	(A+)
L40.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + (5 + 2)))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=Subst L45,L46)
L45.	$\vdash (5 + 2) = 7$	(M+)
L46.	$\vdash \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7))) = \lambda x_{\bullet}(5x^3 + (3x^2 + (3x + 7)))$	(=I)

Abbildung 4.18: Der Beweis von THM auf der Faktenebene

## 4.5 Eine notwendige zukünftige Erweiterung

Wie in Abschnitt 4.3.2 angesprochen, gleicht der vom Taktikgenerator an  $\Omega$ -MKRP gelieferte Plan einer Sequenz von Taktiken auf der Faktenebene. Das bedeutet, daß jede Taktik der Anwendung einer bestimmten Hypothese entspricht. Dafür ist es notwendig, daß diese Hypothesen auch im jeweiligen Beweis vorhanden sind. Im gegenwärtigen Stadium der Implementation von  $\Omega$ -MKRP ist dies allerdings nicht a priori zu gewährleisten, da für jedes zu beweisende Theorem auch alle dessen Hypothesen neu angegeben werden müssen. Fehlt eine der vom Taktikgenerator geforderten Annahmen, führt dies unausweichlich zu einem Fehler in der Planausführung. Um dies zu vermeiden, muß SAPPER vor dem Hintergrund einer *mathematischen Wissensbasis* laufen. Dafür bietet sich die  $\Omega$ -MKRP-Datenbank an, die zukünftig aufgebaut werden soll. In ihr wird mathematisches Wissen in der Form von *Theorien* gespeichert, die hierarchisch aufeinander aufbauen können. Eine solche Theorie stellt Term- und Typdeklarationen, Hypothesen, Taktiken und Methoden zur Verfügung. Theoreme selbst stünden dann ebenfalls im Kontext einer Theorie, womit bekannt wäre, welche Hypothesen und Taktiken für ihren Beweis benutzt werden können.

Beispielweise wäre eine solche Theorie die additive Gruppe der Polynome über  $\mathbb{R}$ , die als eine Untertheorie die der reellen Zahlen hätte. In diesem Fall wären in dieser Theorie die Hypothesen  $M+$  und  $P+$  neu definiert, während sie  $A+$  und  $C+$  erbe. Ebenso können dann auch die Taktiken zugeordnet werden:  $POP$ -FIRST und  $POP$ -SECOND den reellen Zahlen;  $POLYNOMIAL$ -ADDITION und  $MONOMIAL$ -ADDITION der additiven Gruppe. Über dieser könnte dann der Ring der Polynome über  $\mathbb{R}$  als nächste Theorie aufgebaut werden.

Assoziierte man nun auch noch Algorithmen von CAS mit den Theorien der Datenbank, so daß zum einem die von Algorithmen ausgegebenen Taktiken nur denen in der Theorie bekannten entsprechen und zum anderen ein Algorithmus nur dann aufgerufen werden kann, wenn er auch der Theorie zugeordnet ist, in der  $\Omega$ -MKRP gerade arbeitet, könnten Fehler bei der Planausführung verhindert werden. Dies bedeutete im Beispiel, daß der Algorithmus zur Addition von reellen Polynomen unserer Theorie der additiven Gruppe angehörte. Eine solche Systemarchitektur ist in Abbildung 4.19 dargestellt.

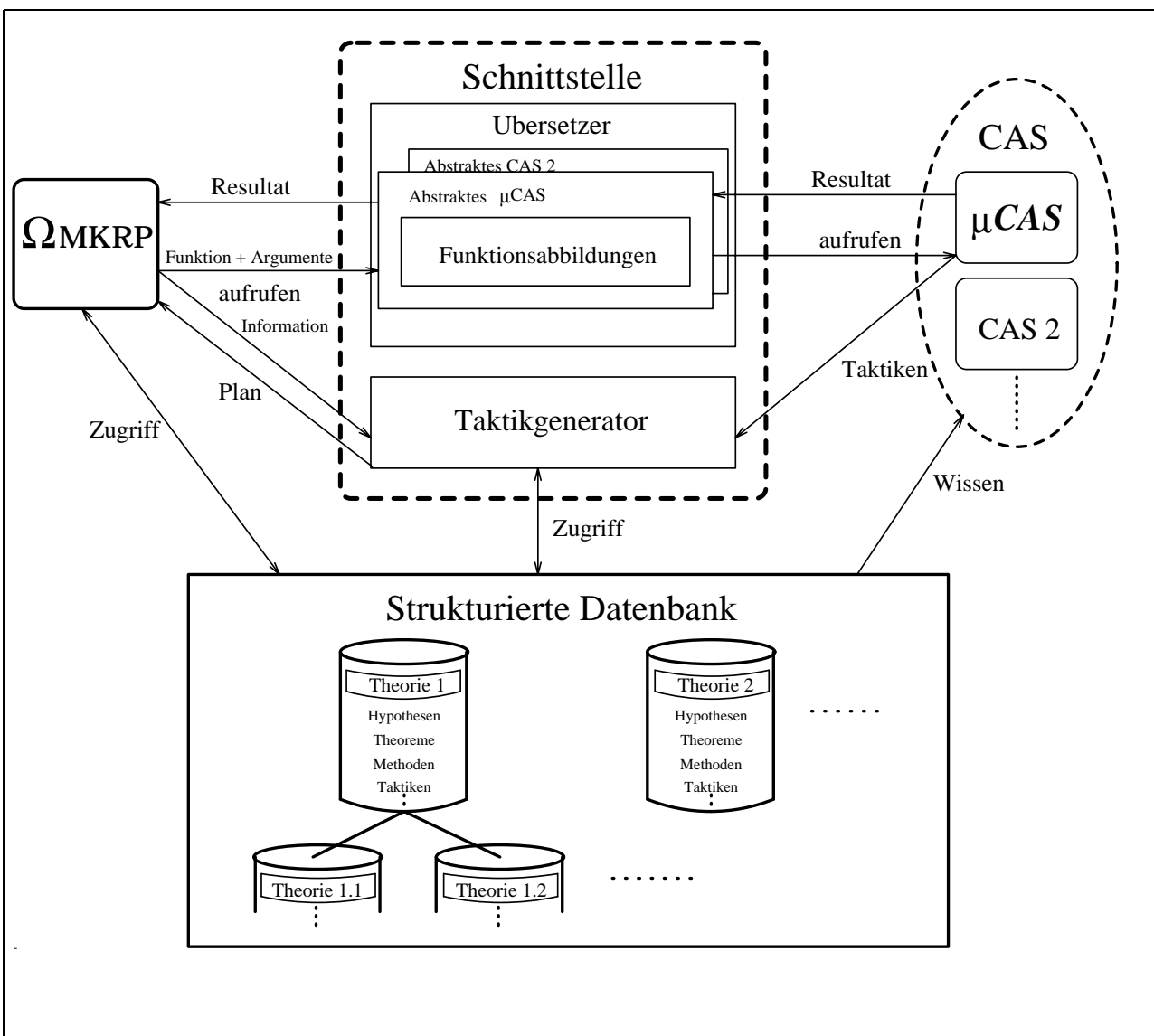


Abbildung 4.19: Die Schnittstelle zwischen  $\Omega$ -MKRP und Computerregelbrakomponenten

## Kapitel 5

# Erweiterung von Computeralgebraalgorithmen – Eine Untersuchung anhand der Polynommultiplikation

Wie bereits in den vorherigen Kapiteln erwähnt, müssen herkömmliche Computeralgebraalgorithmen erweitert werden, um aus ihren Berechnungen Beweispläne gewinnen zu können. Dafür ist es nicht nur interessant zu wissen, wie diese Erweiterung aussehen muß, sondern auch welche Algorithmen bzw. Algorithmenschemata überhaupt zur Planung in Frage kommen. In diesem Kapitel wird diese Frage anhand einer Fallstudie untersucht. Dafür werden drei Algorithmen zur Polynommultiplikation betrachtet, die jeweils auf verschiedenen Algorithmenschemata beruhen. Diese wurden teilweise [Zip93] entnommen oder im Falle des rekursiven Algorithmus in Abschnitt 5.2 vom Autor selbst entworfen.

In den folgenden drei Abschnitten werden jeweils die Algorithmen vorgestellt, analysiert und deren Implementation kurz erläutert. Der Hauptteil der jeweiligen Abschnitte ist dann den Erweiterungen im Rahmen des *Verbose-Modes* gewidmet bzw. einer Begründung, warum eine Erweiterung nicht möglich ist. Zunächst wollen wir aber die notwendigen Funktionen vorstellen, die allen drei Algorithmen gemeinsam sind, und auf die Mathematik eingehen, in deren Kontext sich die Beweise für die einzelnen Berechnungen befinden.

In  $\mu CAS$  werden Polynome durch CLOS-Objekte dargestellt (siehe Kapitel 4.2.1), deren Hauptkomponente eine Datenliste mit der Repräsentation der einzelnen Monome ist. Durch die Polynommultiplikation entsteht nun ein neues Polynom, dessen Datenliste dem Ergebnis der Multiplikation entspricht. Die Behandlung der Polynome und ihrer Datenlisten geschieht in getrennten Funktionen. Abbildung 5.1 zeigt den algorithmischen Vorspann für

die Polynommultiplikation. Dabei ist die `ca~polynomial-multiplication` die Funktion, die eigentlich vom Benutzer von  $\mu CAS$  bzw. in SAPPER von  $\Omega$ -MKRP aufgerufen wird. Ihr werden zwei Polynome vom Typ `ca+polynomial` und ein Schlüsselwort übergeben. Mit den Datenlisten der Eingabepolynome wird `ca=polynomial-multiplication` aufgerufen, welches nun bezüglich des Schlüsselwortes einen der drei in den nächsten Abschnitten beschriebenen Algorithmen aufruft. Diese arbeiten dann ausschließlich mit den Datenlisten der Polynome. Zusätzlich behandelt `ca=polynomial-multiplication` die Fälle, in denen mindestens eines der beiden Polynome das Nullpolynom ist.

```
(defgeneric ca~polynomial-multiplication (poly1 poly2 &key scheme)
  (:method ((poly1 ca+polynomial) (poly2 ca+polynomial) &key scheme)
    (ca=output-tactic 'polynomial-multiplication)
    (ca~create-polynomial
     (ca=polynomial-multiplication (ca=get-poly-data poly1)
                                   (ca=get-poly-data poly2)
                                   scheme)
     :field (ca~get-poly-field poly1)
     :var (ca=get-poly-var-number poly1))))

(defgeneric ca=polynomial-multiplication (data1 data2 scheme)
  (:method ((data1 (eql nil)) (data2 list) scheme)
    (ca=output-tactic 'multiply-zero-left)
    nil)
  (:method ((data1 list) (data2 (eql nil)) scheme)
    (ca=output-tactic 'multiply-zero-right)
    nil)
  (:method ((data1 list) (data2 list) (scheme (eql :rec)))
    (ca=polynomial-multiplication-rec1 data1 data2))
  (:method ((data1 list) (data2 list) (scheme (eql :it)))
    (ca=polynomial-multiplication-it data1 data2))
  (:method ((data1 list) (data2 list) (scheme (eql :dc)))
    (if (= 1 (length (ca=exps (car data1))))
        (ca=polynomial-multiplication-dc data1 data2)
        (error))))
```

Abbildung 5.1: Der algorithmische Rahmen der verschiedenen Polynommultiplikationen

Der Programmcode in Abbildung 5.1 enthält bereits einige Taktikausgaben mittels des `ca=output-tactic` Kommandos. Dabei handelt es sich um komplexe Taktiken (siehe hierzu auch Kapitel 4.3.2). Zu ihrem Verständnis ist es notwendig, zunächst die Hypothesen zu betrachten, auf denen diese Taktiken beruhen bzw. die den in ihnen aufgerufenen einfachen Taktiken entsprechen.

Die Hypothesen für die Polynommultiplikation sind aufgelistet in Abbildung 5.2. Dabei

P*.	$P^* \vdash \forall p \bullet \forall q \bullet (p \otimes q) = \lambda z \bullet (pz * qz)$	(Hyp)
M*.	$M^* \vdash \forall x \bullet \forall m \bullet \forall n \bullet \forall a \bullet \forall b \bullet (ax^m * bx^n) = (a * b)x^{(m+n)}$	(Hyp)
DL.	$DL \vdash \forall a \bullet \forall b \bullet \forall c \bullet (a * (b + c)) = ((a * b) + (a * c))$	(Hyp)
DR.	$DR \vdash \forall a \bullet \forall b \bullet \forall c \bullet ((a + b) * c) = ((a * c) + (b * c))$	(Hyp)
C*.	$C^* \vdash \forall a \bullet \forall b \bullet (a * b) = (b * a)$	(Hyp)
A+.	$A+ \vdash \forall a \bullet \forall b \bullet \forall c \bullet ((a + b) + c) = (a + (b + c))$	(Hyp)
0*.	$0^* \vdash \forall x \bullet (0 * x) = 0$	(Hyp)

Abbildung 5.2: Die notwendigen Hypothesen für die Polynommultiplikation

haben diese im einzelnen folgende Bedeutung:

- P\*** Multiplikation von zwei Polynomen
- M\*** Multiplikation von zwei Monomen
- DL** Distributivität der Multiplikation von links
- DR** Distributivität der Multiplikation von rechts
- C\*** Kommutativität der Multiplikation
- A+** Assoziativität der Addition
- 0\*** Multiplikation mit 0

Bei den beiden Hypothesen P\* und M\* ist zu beachten, daß sie sich in der hier dargestellten Form auf den Fall der Polynom- bzw. Monommultiplikation in  $\mathbb{R}[x]$  beziehen. In Polynomringen mit mehr als einer Variablen sind beide Hypothesen von entsprechend anderer Form, zum Beispiel für  $\mathbb{R}[x_1, x_2]$ :

$$P^*. \quad P^* \vdash \forall p \bullet \forall q \bullet (p \otimes q) = \lambda z_1 \bullet \lambda z_2 \bullet (pz_1 z_2 * qz_1 z_2) \quad (\text{Hyp})$$

$$M^*. \quad M^* \vdash \forall x_1 \bullet \forall x_2 \bullet \forall m_1 \bullet \forall m_2 \bullet \forall n_1 \bullet \forall n_2 \bullet \forall a \bullet \forall b \bullet (ax_1^{m_1} x_2^{m_2} * bx_1^{n_1} x_2^{n_2}) = (a * b)x_1^{(m_1+n_1)} x_2^{(m_2+n_2)} \quad (\text{Hyp})$$

Die für den jeweiligen Polynomring notwendige Hypothese wird automatisch von den Taktiken selektiert und angewendet.

Damit ist es jetzt möglich, die formalen Definitionen (siehe auch Kapitel 3.3.2) für die in den Funktionen `ca~polynomial-multiplication` und `ca=polynomial-multiplication` benutzten Taktiken anzugeben. (Für die Definitionen aller SAPPER bisher bekannten Taktiken siehe auch Anhang B) Diese entsprechen jeweils der Anwendung von einer oder im Falle der MULTIPLY-ZERO-RIGHT-Taktik zwei Hypothesen.

```
tactic POLYNOMIAL-MULTIPLICATION  APPLY(P*)
tactic MULTIPLY-ZERO-LEFT          APPLY(0*)
tactic MULTIPLY-ZERO-RIGHT         APPEND(APPLY(C*)
                                         APPLY(0*))
```

Bei den in den folgenden drei Abschnitten vorgestellten Algorithmen ist zu beachten, daß sie zwar das Ergebnis der eigentlichen Multiplikation liefern, dieses aber nicht unbedingt der Datenliste eines Polynoms in *normaler Ordnung* entsprechen muß. Sollte Sortieren notwendig sein, so geschieht dies nach Ablauf des jeweiligen Algorithmus in einer separaten

Funktion, mit der wir uns im Folgenden aber nicht beschäftigen wollen. Die von dieser Funktion benutzten Taktiken und die geforderten Hypothesen sind in Anhang B aufgeführt.

## 5.1 Iterativer Algorithmus

Der hier vorgestellte *iterative* Algorithmus entspricht dem Schema der Polynommultiplikation, das jeder Schüler im Mathematikunterricht erlernen muß, und ist wahrscheinlich die einfachste Form, dieses zu implementieren. Er basiert auf der simplen Tatsache, daß bei der Multiplikation zweier Polynome jedes Monom des ersten Polynoms mit jedem Monom des zweiten Polynoms genau einmal multipliziert wird:

Seien  $p = \sum_{i=1}^n \alpha_i x_1^{e_{1i}} \cdots x_r^{e_{ri}}$  und  $q = \sum_{j=1}^m \beta_j x_1^{e_{1j}} \cdots x_r^{e_{rj}}$  zwei Polynome in  $r$  Variablen, dann läßt sich das Produkt von  $p$  und  $q$  auffassen als

$$s := p \otimes q = \sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j x_1^{e_{1i}+e_{1j}} \cdots x_r^{e_{ri}+e_{rj}} \quad (5.1)$$

Damit ist  $s$  ein Polynom mit  $n \cdot m$  Monomen, und die Doppelsumme in (5.1) kann als Vorschrift für den folgenden Algorithmus interpretiert werden

*Nimm das erste Monom von  $p$*   
*Multipliziere dieses Monom mit den Monomen von  $q$*   
*Füge das Ergebnis dem Polynom  $s$  an*  
*Entferne das erste Monom aus  $p$*   
*Wiederhole den Algorithmus bis  $p$  leer ist*

Dieses Schema entspricht programmieretechnisch zwei ineinandergeschachtelter Schleifen: Eine, die alle Monome von  $p$  behandelt, und innerhalb dieser eine zweite, die über die Monome von  $q$  läuft und in der die eigentliche Multiplikation vorgenommen wird. Für die Analyse dieses Algorithmus kann die Anzahl der auszuführenden Monommultiplikationen einfach abgezählt werden. Bei der Monomanzahl unserer Polynome  $p$  und  $q$  wird die äußere Schleife  $n$ -mal und somit die innere Schleife  $n \cdot m$ -mal durchlaufen. Die Gesamtzahl der Monommultiplikationen entspricht also  $n \cdot m$ .

Die Implementation des Algorithmus in CLOS ist dargestellt in Abbildung 5.3. Dabei entsprechen die Argumente den Datenlisten der Polynome, und das Ergebnis der Funktion `ca=pmult-iterative` ist Datenliste des Ergebnispolynoms  $s$ . Die erste `mapcar`-Funktion ist die Schleife über den Monomen von  $p$  und die zweite die Schleife über  $q$ ; innerhalb der `cons`-Funktion werden die neuen Monome von  $s$  gebildet, indem die Koeffizienten der Monome von  $p$  und  $q$  multipliziert und die einzelnen Exponenten addiert werden.

```

(defun ca=pmult-iterative (p q)
  (apply #'append
    (mapcar #'(lambda (x)
      (ca=output-tactic 'distributivity-right)
      (mapcar #'(lambda (y)
        (ca=output-tactic 'monomial-multiplication)
        (cons (* (ca=coeff x) (ca=coeff y))
          (map 'list #'(lambda (exps) (ca=exps x) (ca=exps y))))
          q))
      (ca=output-tactic 'push-last)
    p)))

```

Abbildung 5.3: Iterative Polynommultiplikation

In diese Schleifen sind nun die Erweiterungen eingefügt, die notwendig sind, um einen ND-Beweis aus der Berechnung des Algorithmus zu konstruieren. Dies geschieht mittels der Funktion `ca=output-tactic`, die jeweils den Namen einer Taktik an SAPPERS Taktikgenerator (siehe auch Kapitel 4.3.2) übergibt. Insgesamt werden innerhalb des Algorithmus drei verschiedene Taktiken genutzt: zwei in der äußeren und eine in der inneren Schleife.

Diese Taktiken betrachten wir nun der Reihe nach, beginnend mit den Definitionen gemäß Kapitel 3.3.2. Dabei beziehen sich die Bedingungen der COND-Taktikale der folgenden Taktiken darauf, ob ein Term von der entsprechenden Form ist. Es seien  $a, b$  Monome und  $c, d$  beliebige Polynome:

<i>tactic</i> DISTRIBUTIVITY-RIGHT	COND((( $a + c$ ) * $d$ ) APPLY(DR))
<i>tactic</i> MONOMIAL-MULTIPLICATION	COND((( $a * (b + c)$ ) APPEND (APPLY(DL) APPLY(M*))) ( $a * b$ ) APPLY(M*)))
<i>tactic</i> PUSH-LAST	REPEAT-IF((( $a + c$ ) + $d$ ) APPLY(A+)))

Um die intuitive Bedeutung der Taktiken veranschaulichen zu können, betrachten wir jetzt die durch sie erzeugten partiellen Beweisebäume anhand eines Beispiels. Hierfür multiplizieren wir die bereits in Kapitel 4.2.2 benutzten Polynome  $p = (3x^2 + (2x + 5))$  und  $q = (5x^3 + (x + 2))$  in folgendem Theorem, wobei  $\mathcal{H}_1$  die Liste der in Abbildung 5.2 angegebenen Hypothesen ist:

$$\text{THM. } \mathcal{H}_1 \vdash (\lambda x.(3x^2 + (2x^1 + 5x^0)) \otimes \lambda x.(5x^3 + (1x^1 + 2x^0))) = \quad (\text{OPEN}) \\ \lambda x.(15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$$

Der vollständige ND-Beweis für THM umfaßt genau 201 Zeilen und wird hier weggelassen. Aber auch der Beweis auf der Faktenebene ist noch annähernd 100 Zeilen lang. Daher werden wir zur Erläuterung der einzelnen Taktiken nur den ersten Durchlauf der äße-

ren Schleife im Algorithmus aus Abbildung 5.3 und die dabei entstehenden partiellen Beweisbäume auf der Faktenebene betrachten. Der Beweisplan, der an den Taktikgenerator übergeben wird, ist bezüglich der einzelnen Rechenschritte dargestellt in Abbildung 5.4.

Rechenschritt	Taktikausgabe
$(3x^2 + (2x + 5)) * (5x^3 + (x + 2))$	(polynomial-multiplication)
$((3x^2 * (5x^3 + (x + 2))) + ((2x + 5) * (5x^3 + (x + 2))))$	(distributivity-right)
$((15x^5 + (3x^2 * (x + 2))) + ((2x + 5) * (5x^3 + (x + 2))))$	(monomial-multiplication)
$((15x^5 + (3x^3 + (3x^2 * 2))) + ((2x + 5) * (5x^3 + (x + 2))))$	(monomial-multiplication)
$((15x^5 + (3x^3 + 6x^2)) + ((2x + 5) * (5x^3 + (x + 2))))$	(monomial-multiplication)
$(15x^5 + (3x^3 + (6x^2 + ((2x + 5) * (5x^3 + (x + 2))))))$	(push-last)
⋮	⋮

Abbildung 5.4: Anfang der Taktik-Ausgabe zur Berechnung der Formel in THM

Wir vernachlässigen die erste Planzeile, da sie nicht während des Laufs des uns interessierenden Algorithmus entsteht und nur zur Vollständigkeit aufgeführt ist. Die erste von unserem Algorithmus zurückgegebene Taktik ist also DISTRIBUTIVITY-RIGHT. Sie entspricht bei diesem Schleifendurchlauf einer Anwendung des Distributivitätsgesetzes von rechts. Betrachten wir die formale Definition der Taktik, so erkennen wir, daß dieses nur dann angewendet wird, wenn der linke Multiplikand aus mindestens zwei Monomen besteht. Ist der Multiplikand nur ein einzelnes Monom, so hat die Taktik keinerlei Auswirkungen. Die drei nächsten Planzeilen sind jeweils MONOMIAL-MULTIPLICATION. Von diesen entsprechen die ersten beiden einer Anwendung der Distributivität von links und der Monommultiplikation, also dem ersten Fall des COND-Taktikals. Die dritte ist äquivalent zum zweiten Fall im COND-Taktikal und damit eine einfache Anwendung der Monommultiplikation. Die letzte in Abbildung 5.4 aufgeführte Planzeile ist PUSH-LAST. Deren Verhalten wird durch das REPEAT-IF-Taktikal bestimmt, was einer näheren Untersuchung bedarf. Betrachten wir im Beispiel den Term  $(15x^5 + (3x^3 + 6x^2))$  als das Polynom  $(a + c)$  der Bedingung des REPEAT-IF-Taktikals, dann wird  $((2x + 5) * (5x^3 + (x + 2)))$  als letzter Term diesem Polynom angefügt, und es entsteht  $(15x^5 + (3x^3 + (6x^2 + ((2x + 5) * (5x^3 + (x + 2))))))$ . Dies entspricht einer mehrfachen Anwendung der Assoziativität der Addition, und zwar solange der erste Term einem Polynom mit mindestens zwei Monomen entspricht. Da es sich bei  $(a + c)$  um ein Polynom, also immer um einen endlichen Term handelt, terminiert die Ausführung der PUSH-LAST-Taktik.

Damit ergibt sich als Expansion obigen Plans ein partieller Beweisplan auf Faktenebene, wie in Abbildung 5.5 dargestellt. Wird dieser in  $\Omega$ -MKRP auf das Theorem THM angewendet, so erhalten wir den in Abbildung 5.7 angegebenen Teilbeweis.

In der Kontrolle von SAPPERS Taktikgenerator werden während des Laufes dieses Algorithmus zwei Termpositionen verwaltet. (Für unser Beispiel ist dies in Abbildung 5.6

komplexe Taktik	Sequenz von einfachen Taktiken
(polynomial-multiplication)	P*
(distributivity-right)	DR
(monomial-multiplication)	DL, M*
(monomial-multiplication)	DL, M*
(monomial-multiplication)	M*
(push-last)	A+, A+

Abbildung 5.5: Expansion der komplexen Taktiken aus Abbildung 5.4

dargestellt, wobei die Terme an den verwalteten Positionen eingerahmt und die erste bzw. zweite Termposition mit tiefgestellten Nummerierung gekennzeichnet sind.) Zunächst wird beim Aufruf von  $\mu\mathcal{CAS}$  bezüglich der Polynommultiplikation in THM die Termposition der Funktion  $\otimes$  gespeichert. Bezüglich dieser arbeitet die POLYNOMIAL-MULTIPLICATION, deren Code dann die Kontrolle so verändert, daß zweimal die Termposition der Multiplikationsfunktion  $*$  gespeichert wird. Die erste der beiden benutzt DISTRIBUTIVITY-RIGHT, läßt sie unverändert und setzt die zweite auf die Position der ersten der beiden neuentstandenen Multiplikationsfunktionen  $*$ . Mit dieser zweiten Position arbeitet nun MONOMIAL-MULTIPLICATION bei jedem Durchlauf und setzt sie immer auf die neue Position von  $*$  weiter. Das Verhalten dieser Taktik wird dann durch Instantiierung des Ausdruckes in den Bedingungen des COND-Taktikals bestimmt. So entspricht im Beispiel bei der Anwendung der Taktik auf

$$\lambda x_{\bullet}((3x^2 \boxed{*}_2 (5x^3 + (x + 2))) \boxed{+}_1 ((2x + 5) * (5x^3 + (x + 2)))) = \dots$$

$(a * (b + c))$  dem Term  $(3x^2 * (5x^3 + (x + 2)))$ . Die Ausführung von PUSH-LAST geschieht nun wieder bezüglich der ersten der beiden Termpositionen in der Kontrolle. Danach werden beide Termpositionen auf die Position der verbliebenen Multiplikation  $*$  gesetzt.

$\lambda x_{\bullet}(3x^2 + (2x + 5)) \boxed{\otimes}(5x^3 + (x + 2)) = \dots$
$\lambda x_{\bullet}(3x^2 + (2x + 5)) \boxed{*}_{1,2}(5x^3 + (x + 2)) = \dots$
$\lambda x_{\bullet}((3x^2 \boxed{*}_2 (5x^3 + (x + 2))) \boxed{+}_1 ((2x + 5) * (5x^3 + (x + 2)))) = \dots$
$\lambda x_{\bullet}((15x^5 + (3x^2 \boxed{*}_2 (x + 2))) \boxed{+}_1 ((2x + 5) * (5x^3 + (x + 2)))) = \dots$
$\lambda x_{\bullet}((15x^5 + (3x^3 + (3x^2 \boxed{*}_2 2))) \boxed{+}_1 ((2x + 5) * (5x^3 + (x + 2)))) = \dots$
$\lambda x_{\bullet}((15x^5 + (3x^3 + 6x^2)) \boxed{+}_1 ((2x + 5) * (5x^3 + (x + 2)))) = \dots$
$\lambda x_{\bullet}(15x^5 + (3x^3 + (6x^2 + ((2x + 5) \boxed{*}_{1,2}(5x^3 + (x + 2))))) = \dots$

Abbildung 5.6: Von der Kontrolle verwalteten Termpositionen

THM.	$\vdash (\lambda x_{\bullet} (3x^2 + (2x + 5)) \otimes \lambda x_{\bullet} (5x^3 + (x + 2))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L3,L4)
L3.	$\vdash (\lambda x_{\bullet} (3x^2 + (2x + 5)) \otimes \lambda x_{\bullet} (5x^3 + (x + 2))) =$ $\lambda x_{\bullet} ((3x^2 + (2x + 5)) * (5x^3 + (x + 2)))$	(P*)
L4.	$\vdash \lambda x_{\bullet} ((3x^2 + (2x + 5)) * (5x^3 + (x + 2))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L7,L8)
L7.	$\vdash ((3x^2 + (2x + 5)) * (5x^3 + (x + 2))) =$ $((3x^2 * (5x^3 + (x + 2))) + ((2x + 5) * (5x^3 + (x + 2))))$	(DR)
L8.	$\vdash \lambda x_{\bullet} ((3x^2 * (5x^3 + (x + 2))) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L11,L12)
L11.	$\vdash (3x^2 * (5x^3 + (x + 2))) = ((3x^2 * 5x^3) + (3x^2 * (x + 2)))$	(DL)
L12.	$\vdash \lambda x_{\bullet} (((3x^2 * 5x^3) + (3x^2 * (x + 2))) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L18,L19)
L18.	$\vdash (3x^2 * 5x^3) = 15x^5$	(M*)
L19.	$\vdash \lambda x_{\bullet} ((15x^5 + (3x^2 * (x + 2))) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L22,L23)
L22.	$\vdash (3x^2 * (x + 2)) = ((3x^2 * x) + (3x^2 * 2))$	(DL)
L23.	$\vdash \lambda x_{\bullet} ((15x^5 + ((3x^2 * x) + (3x^2 * 2))) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L29,L30)
L29.	$\vdash (3x^2 * x) = 3x^3$	(M*)
L30.	$\vdash \lambda x_{\bullet} ((15x^5 + (3x^3 + (3x^2 * 2))) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L36,L37)
L36.	$\vdash (3x^2 * 2) = 6x^2$	(M*)
L37.	$\vdash \lambda x_{\bullet} ((15x^5 + (3x^3 + 6x^2)) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L40,L41)
L40.	$\vdash ((15x^5 + (3x^3 + 6x^2)) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $(15x^5 + ((3x^3 + 6x^2) + ((2x + 5) * (5x^3 + (x + 2)))))$	(A+)
L41.	$\vdash \lambda x_{\bullet} (15x^5 + ((3x^3 + 6x^2) + ((2x + 5) * (5x^3 + (x + 2))))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L44,L45)
L44.	$\vdash ((3x^3 + 6x^2) + ((2x + 5) * (5x^3 + (x + 2)))) =$ $(3x^3 + (6x^2 + ((2x + 5) * (5x^3 + (x + 2)))))$	(A+)
L45.	$\vdash \lambda x_{\bullet} (15x^5 + (3x^3 + (6x^2 + ((2x + 5) * (5x^3 + (x + 2)))))) =$ $\lambda x_{\bullet} (15x^5 + (10x^4 + (28x^3 + (8x^2 + (9x + 10))))))$	(=Subst L48,L49)

Abbildung 5.7: Der Faktenebenenbeweis für den in Abbildung 5.5 dargestellten Beweisplan

## 5.2 Rekursiver Algorithmus

Da LISP eine funktionale Programmiersprache ist, deren Hauptdatenstruktur Listen sind, läßt es sich in ihr besonders einfach und elegant rekursiv programmieren [WH87, May88]. Da Polynome in  $\mu CAS$  prinzipiell als Listen repräsentiert sind, sind auch die meisten Operationen auf diesen mittels *Rekursion* realisiert. So wird für die Polynommultiplikation ebenfalls ein *rekursiver* Algorithmus bereitgestellt. Dessen Schema entspricht im wesentlichen der in Abschnitt 5.1 vorgestellten Variante; insbesondere die Berechnung der Gleichung (5.1). Der eigentliche Unterschied ist der, daß nicht über die Monome der beiden Polynome iteriert wird, sondern daß die Datenlisten mittels zweier rekursiver Funktionen bearbeitet werden. Damit ist auch dieser Algorithmus schnell analysiert, entsprechen doch die beiden rekursiven Funktionen gerade den beiden Schleifen des iterativen Algorithmus aus Abschnitt 5.1. Das heißt für Polynome mit  $n$  und  $m$  Monomen werden ebenfalls  $n \cdot m$  Monommultiplikationen ausgeführt.

```
(defun ca=pmult-recursive1 (p q)
  (when p
    (ca=output-tactic 'distributivity-right)
    (append
      (ca=pmult-recursive2 (first p) q)
      (ca=pmult-recursive1 (rest p) q))
    (ca=output-tactic 'push-last)))

(defun ca=pmult-recursive2 (monomial q)
  (when q
    (ca=output-tactic 'monomial-multiplication)
    (append
      (list (cons (* (ca=coeff monomial) (ca=coeff (first q)))
                  (map 'list #' + (ca=exps monomial) (ca=exps (first q)))))
      (ca=pmult-recursive2 monomial (rest q)))))
```

Abbildung 5.8: Rekursive Polynommultiplikation

Der Code für beide Funktionen ist dargestellt in Abbildung 5.8. Dabei ist `ca=pmult-recursive1` eine Rekursion auf der Datenliste des Polynoms  $p$  und ruft für jedes Monom in  $p$  einmal die Funktion `ca=pmult-recursive2` auf. In dieser wiederum wird das übergebene Monom aus  $p$  mit den Monomen in  $q$  multipliziert. Dies geschieht in der bereits in Abschnitt 5.1 näher erläuterten `cons`-Funktion.

Die Erweiterung für die Beweisplanung ist auch beim rekursiven Algorithmus ohne weiteres möglich. Hierzu können exakt die gleichen Taktiken benutzt werden wie für den iterativen Algorithmus. Betrachten wir den Code in Abbildung 5.8, so erkennen wir, daß

die Ausgabe der Taktiken an den Punkten im Algorithmus geschieht, die denen im iterativen entsprechen. So werden Taktiken `DISTRIBUTIVITY-RIGHT` und `PUSH-LAST` in der ersten rekursiven Funktion ausgegeben; die erste vor, die zweite nach dem Aufruf von `ca=pmult-recursive2`. Von dieser Funktion wird dann `MONOMIAL-MULTIPLICATION` zurückgegeben. Da die beiden `mapcar`-Schleifen des Algorithmus in Abbildung 5.3 den beiden rekursiven Funktionen `ca=pmult-recursive1` und `ca=pmult-recursive2` entsprechen, ergeben sich die gleichen Berechnungen dieser und damit auch genau die gleichen Beweispläne als Ausgabe.

### 5.3 Devide-and-Conquer-Algorithmus

In diesem Abschnitt geben wir nun ein Beispiel für einen Algorithmus, dessen Erweiterung für die Beweisplanung nicht möglich ist und erläutern an anhand dessen die Probleme, die bei der Erweiterung von Algorithmen auftreten können. Hierfür betrachten wir eine Polynommultiplikation, die nach dem *Devide-and-Conquer*-Schema rechnet.

Devide-and-Conquer (eigentlich Devide-Conquer-Merge) ist ein allgemeines Designschema für Algorithmen, um die Anzahl der Rechenoperationen zum Lösen eines Problems zu verringern. Dabei wird das ursprüngliche Problem so lange in Unterprobleme zerlegt (devide), bis diese einfach (meist trivial) gelöst werden können (conquer). Diese einzelnen Lösungen werden dann zusammengefügt (merge), so daß eine Gesamtlösung für das Ausgangsproblem entsteht [Wir83, OW90, CLR92].

Auf diesem Schema beruht der Algorithmus von KARATSUBA und OFMAN zur Multiplikation *univariater* Polynome [KO63]. Dabei macht er sich folgende Gleichung zunutze:

Seien  $f(x) = \sum_{i=1}^k \alpha_i x^i$  und  $g(x) = \sum_{i=1}^l \beta_i x^i$  zwei univariate Polynome, dann können diese geschrieben werden als

$$\begin{aligned} f(x) &= f_0(x)x^m + f_1(x) \\ g(x) &= g_0(x)x^m + g_1(x) \end{aligned}$$

wobei  $f_0$  ein Polynom vom Grad  $k \Leftrightarrow m$ ,  $g_0$  ein Polynom vom Grad  $l \Leftrightarrow m$  und  $f_1, g_1$  Polynome vom Grad  $< m$  sind. Das Produkt von  $f$  und  $g$  kann nun folgendermaßen berechnet werden:

$$\begin{aligned} f(x)g(x) &= f_0g_0x^{2m} + (f_1g_0 + f_0g_1)x^m + f_1g_1 \\ &= f_0g_0x^{2m} + ((f_1 + g_1)(g_1 + g_0) \Leftrightarrow f_0g_0 \Leftrightarrow f_1g_1)x^m + f_1g_1 \end{aligned} \quad (5.2)$$

In Formel (5.2) treten die Produkte  $f_0g_0$  und  $f_1g_1$  jeweils doppelt auf, so daß für die Berechnung von  $f \cdot g$  nur drei Polynommultiplikationen benötigt werden, die erneut mit dem Devide-and-Conquer-Schema berechnet werden. Damit ist im Gegensatz zu rekursivem und

iterativem Algorithmus die Analyse etwas aufwendiger. Nehmen wir ohne Beschränkung der Allgemeinheit an, daß  $f$  mehr oder gleich viel Monome umfasse als  $g$ , und darüber hinaus die Anzahl der in  $f$  vorkommenden Monome  $n$  und die Anzahl der Monommultiplikationen  $M(n)$  sei, dann gilt

$$M(n) = 3M\left(\left\lceil \frac{n}{2} \right\rceil\right) = \underbrace{3 \cdot \dots \cdot 3}_{\lceil \log_2 n \rceil} M(1) = 3^{\lceil \log_2 n \rceil} = \left\lceil n^{\log_2 3} \right\rceil = O(n^{1.57})$$

Vergleichen wir dieses Ergebnis nun mit denen der Analysen in den Abschnitten 5.1 und 5.2, die jeweils  $n \cdot m$  Monommultiplikationen benötigten. Nehmen wir auch hier o.B.d.A. an, daß  $n \geq m$  gilt, so befinden sich die Algorithmen in der Klasse  $O(n^2)$ . Dies bedeutet also, daß der Devide-and-Conquer-Algorithmus bezüglich der Anzahl von Koeffizientenmultiplikationen effizienter ist als die beiden vorhergehenden.

Für eine effiziente Implementation des Algorithmus ist es außerdem wichtig, daß  $f_0$  und  $f_1$  möglichst gleich viele Monome enthalten. Das bedeutet, bei gerader Anzahl von Monomen in  $f$  enthält  $f_0$  gleich viele Monome wie  $f_1$ , bei ungerader Anzahl in  $f$  enthält  $f_0$  genau ein Monom mehr als  $f_1$ . Abbildung 5.9 zeigt die LISP-Implementation des Algorithmus. In diesem werden zunächst in der `let*`-Umgebung der Grad  $m$  berechnet und bezüglich dessen die Polynome geteilt. Der Hauptteil der Funktion besteht aus einer großen Fallunterscheidung mit Hilfe der `cond`-Funktion (hierbei handelt es sich um die COMMON LISP-Funktion `cond` und nicht um das in Kapitel 3.3.2 Definition 3.23 eingeführte Taktikal). Dabei werden vier Fälle unterschieden, die, in dieser Reihenfolge im Algorithmus, folgenden Berechnungen entsprechen (Diese sind in Abbildung 5.9 mit Kommentarzeilen gekennzeichnet.):

1.  $f$  besteht insgesamt aus einem Monom: Wir multiplizieren dieses Monom innerhalb des `mapcar`-Befehls mit dem Polynom  $g$ .
2. Polynom  $g_0$  existiert nicht: Wir berechnen die Formel

$$[(f_1 + f_0)g_1 \Leftrightarrow f_1g_1]x^m + f_1g_1 = f_0g_1x^m + f_1g_1$$

mittels rekursiver Aufrufe von `ca=pmult-devide-and-conquer` bezüglich  $(f_1 + f_0)$  und  $g_0$  bzw.  $f_1$  und  $g_1$ .

3. Polynom  $g_1$  ist leer: Wir berechnen

$$f_0g_0x^{2m} + [(f_1 + f_0)g_0 \Leftrightarrow f_0g_0]x^m = f_0g_0x^{2m} + f_1g_0x^m.$$

4. Alle Teilpolynome enthalten mindestens ein Monom: Wir berechnen die ursprüngliche Formel aus (5.2).

```

(defun ca=pmult-devide-and-conquer (f g)
  (let* ((m (ca=degree-of-split f))
         (f0 (ca=split-poly-at-exponent f m))
         (f1 (ca=trim-poly-list f (length f0)))
         (g0 (ca=split-poly-at-exponent g m))
         (g1 (ca=trim-poly-list g (length g0))))
    (cond ((null f1) (mapcar #'(lambda (x)
                                (cons
                                 (* (ca=coeff x) (ca=coeff (first f0)))
                                 (list (+ (first (ca=exps x)) m))))
                              g))
          ((null g0) (let ((y (ca=pmult-devide-and-conquer f1 g1))
                          (z (ca=pmult-devide-and-conquer f0 g1)))
                      ;; f0 g1 x^m + f1 g1
                      (ca=padd-recursive
                       (ca=add-exponent z m)
                       y)))
          ((null g1) (let ((x (ca=pmult-devide-and-conquer f0 g0))
                          (z (ca=pmult-devide-and-conquer f1 g0)))
                      ;; f0 g0 x^2m + f1 g0 x^m
                      (ca=padd-recursive
                       (ca=add-exponent x (* 2 m))
                       (ca=add-exponent z m))))
          (t (let ((x (ca=pmult-devide-and-conquer f0 g0))
                  (y (ca=pmult-devide-and-conquer f1 g1))
                  (z (ca=pmult-devide-and-conquer
                     (ca=padd-recursive f1 f0)
                     (ca=padd-recursive g1 g0))))
              ;; f0 g0 x^2m + [(f1 + f0)(g1 + g0) - f0 g0 - f1 g1] x^m + f1 g1
              (ca=padd-recursive
               (ca=padd-recursive
                (ca=add-exponent x (* 2 m))
                (ca=add-exponent
                 (ca=psubtr-recursive z (ca=padd-recursive x y))
                 m))
               y))))))

```

Abbildung 5.9: Polynommultiplikation nach KARATSUBA und OFMAN

Nach der ausführlichen Betrachtung des Algorithmus stellt sich nun die Frage, inwieweit es möglich ist, ihn durch Taktikausgaben zu erweitern und damit zur Beweisplanung einzusetzen. Hierzu können wir zunächst feststellen, daß die in Taktiken, die wir für die beiden vorangegangenen Algorithmen benutzt haben, für den Devide-and-Conquer-Algorithmus nicht brauchbar sind, da sie seinem Schema nicht angepaßt werden können. Folgten wir aber genau diesem Schema, erhielten wir einen Beweis, der den Berechnungen des Algorithmus entspricht. Dieser wäre dann allerdings wenig geeignet, die einzelnen Rechenschritte bei der Polynommultiplikation nachzuvollziehen oder Beweiserklärungen daraus zu gewinnen. Aufgabe dieser Diplomarbeit ist es schließlich, nicht nur vollständige, sondern auch darstellbare Beweise aus Algorithmen zu extrahieren und nicht die Korrektheit einzelner Algorithmen zu prüfen. Dies ist mit herkömmlichen Mechanismen zur Softwareverifikation sicher besser machbar.

Eine weitere Möglichkeit wäre die Rekonstruktion eines Beweises aus den Berechnungen des Algorithmus, der dem in Abbildung 5.7 ähnelt. Vom Autor wurden mehrere Versuche unternommen, dies zu realisieren. Diese scheiterten jedoch aus folgenden Gründen:

Werden im rekursiven und iterativen Algorithmus Monome ausschließlich durch Multiplikation und erst danach durch Addition berechnet, so erfolgt im Devide-and-Conquer-Algorithmus die Berechnung durch eine Mischung aus Addition, Subtraktion und Multiplikation. Dadurch daß die Anzahl der Koeffizientenmultiplikationen möglichst gering gehalten wird, wird ein Datum zum Errechnen mehrerer verschiedener Monome eingesetzt. Damit ist eine Taktikausgabe während der eigentlichen Multiplikation auch dann nicht möglich, wenn die vorher durchlaufenen Fallunterscheidungen innerhalb des Algorithmus registriert wurden. Diese Problematik wird noch dadurch verstärkt, daß für verschieden lange Polynome die gleichen Fälle durchlaufen. Betrachten wir zum Beispiel erneut die Polynommultiplikation aus dem Theorem THM in Abschnitt 5.1, so werden bei deren Berechnung dreimal der vierte Fall durchlaufen und neun Koeffizientenmultiplikationen durchgeführt. Addieren wir zum zweiten Polynom  $5x^2 + x + 2$  noch ein quadratisches Monom  $x^2$ , so werden bei dieser Polynommultiplikation genau die gleichen Fälle in gleicher Reihenfolge und gleicher Rekursionstiefe durchlaufen und nur zwei Koeffizienten mehr multipliziert.

Eine Möglichkeit zur Lösung des Problems könnte darin bestehen, die Taktikausgaben in die Algorithmen zur Addition und Subtraktion einzubauen. Dies würde aber bedeuten, daß einerseits die ursprünglichen Algorithmen hierfür nicht mehr benutzt werden könnten, da in diesen bereits andere Taktiken ausgegeben werden, andererseits, daß die eigentlichen Monommultiplikationen in diesen simuliert werden müßten. Es wird allerdings vom Autor bezweifelt, daß dies mit vertretbarem Aufwand gemacht werden kann. Darüber hinaus wirft es die Frage auf, ob die komplizierte Erweiterung elaborierter Algorithmen, wie dem vorliegenden überhaupt notwendig und wünschenswert ist. Statt dessen wäre es durchaus

möglich, schnelle und effiziente Algorithmen ohne Verbose-Mode zur Beweissuche zu benutzen und zum Nachweis der Korrektheit einer Berechnung Algorithmen mit iterativem oder rekursivem Charakter einzusetzen. Wie wir in diesem Kapitel gesehen haben, sind Taktiken für diese sehr ähnlich dem Rechenschema des Algorithmus und ihre Ausgabe mit wenig Aufwand zu erreichen.

# Kapitel 6

## Ein Beispiel

Nachdem in den vorangegangenen Kapiteln SAPPER vorgestellt und beschrieben worden ist, wird in diesem Kapitel seine Anwendung auf ein konkretes Beispiel betrachtet. Dabei handelt es sich um ein *Optimierungsproblem* aus der Ökonomie. Die konkrete Aufgabe ist die *Minimierung der Kostenfunktion* einer Maschine bei der Herstellung eines Produkts. Dieses Beispiel ist aus verschiedenen Gründen gut geeignet zur Demonstration des Zusammenspiels von Deduktion und Computeralgebra. Erstens handelt es sich bei solchen Optimierungsaufgaben um Probleme, die stets nach dem gleichen Schema gelöst werden können, was bedeutet, daß sie in einer kompakten Theorie ausgedrückt werden können. Zweitens ist die Mathematik, die sich hinter ihnen verbirgt, so überschaubar, daß deren Kodierung in  $\Omega$ -MKRP hier vollständig erläutert werden kann. Und drittens handelt es sich bei den benötigten Berechnungen fast ausschließlich um Polynommanipulationen, was den Einsatz von SAPPER zur Lösung derartiger Aufgaben besonders geeignet erscheinen läßt.

Im folgenden Abschnitt wird zunächst die Problemstellung für das Beispiel vorgestellt. Abschnitt 6.2 zeigt dessen Darstellung in  $\Omega$ -MKRP und auch die für eine Lösung des Problems mit Beweis notwendigen Voraussetzungen. Diese Lösung mit SAPPER und  $\Omega$ -MKRP wird dann im letzten Abschnitt dieses Kapitels erläutert.

### 6.1 Problemstellung

Die Aufgabe selbst ist [WiW], einer Sammlung von Examensaufgaben für Studenten der Wirtschaftswissenschaften, entnommen. Zur klareren Darstellung wurden vom Autor jedoch einige der numerischen Werte vereinfacht. Hier nun zunächst die Beschreibung des Problems:

*In einem Preßwerk sind mittels automatischer Pressen Kotflügel zu stanzen. Die Leistungsintensität  $d$  der Presse läßt sich zwischen  $1 \leq d \leq 7$  Kotflügel pro Minute verändern. Zum*

Betrieb des Aggregats werden die zwei Produktionsfaktoren „Kühlmittel“ und „Energie“ benötigt, deren Verbrauchsfunktion in Abhängigkeit von der Leistungsintensität lauten:

$$\bullet r_1 = (4d^2 \Leftrightarrow 18d + 7) \frac{l}{\text{Kotflügel}} \quad \bullet r_2 = (0.5d^2 \Leftrightarrow 1.5d + 0.5) \frac{kWh}{\text{Kotflügel}}$$

mit folgenden Preisen für Kühlmittel und Strom:

$$\bullet p_1 = 0.5 \frac{DM}{l} \quad \bullet p_2 = 2 \frac{DM}{kWh}$$

Ermittle die Leistungsintensität  $d$ , so daß die Gesamtkosten der Produktion minimal sind.

Um die Schreibweise der Benennungen im Folgenden zu vereinfachen, werden wir anstelle von Kotflügeln allgemein von dem Produkt *prod* sprechen.

Die Aufgabe des Prüflings ist es nun, die *Gesamtkostenfunktion* für die Maschine zu berechnen, deren Minimum zu bestimmen und festzustellen, ob dieses tatsächlich der kleinste Wert im möglichen Intervall für die Leistungsintensität ist. Die Gesamtkostenfunktion  $f$  selbst berechnet sich aus der Summe der Einzelkostenfunktionen, die sich wiederum aus dem Produkt der Verbrauchsfunktionen mit den entsprechenden Preisen ergeben. Das bedeutet in unserem Fall, daß  $f := p_1 \cdot r_1 + p_2 \cdot r_2 \frac{DM}{prod}$  ist. Für dieses  $f$  wird dann das *totale Minimum* im reellen Intervall  $I = [1, 7]$  berechnet.

Das Problem selbst kann also mit relativ einfachen analytischen Mitteln gelöst werden. Hier möchten wir uns aber nicht nur mit der Lösung beschäftigen, da für deren Berechnung ein CAS alleine oder auch nur Papier und Bleistift genügen. Vielmehr wollen wir auch die Korrektheit einer berechneten Lösung garantieren können. Hierfür muß die Aufgabe geeignet in  $\Omega$ -MKRP repräsentiert werden, wobei besonders darauf geachtet werden muß, daß die numerische Lösung des Problems auch berechnet wird. Es muß also konstruktiv gearbeitet werden, was bedeutet, daß wir uns nicht mit einem abstrakten Argument zufrieden geben wollen, das besagt, daß die gegebene Funktion  $f$  natürlich irgendeinen kleinsten Wert im Intervall  $I$  annimmt. Statt dessen soll dieser kleinste Wert auch konkret ausgegeben werden. Diese Vorgabe ist bei der Problemkodierung beachtet, die im folgenden Abschnitt beschrieben ist.

## 6.2 Problemkodierung in $\Omega$ -MKRP

Die Kodierung des Minimierungsproblems läßt sich im wesentlichen in drei Bereiche unterteilen. Zunächst definieren wir in Abschnitt 6.2.1 eine Theorie, in der Einheiten, Kostenfunktionen und Preise erfaßt und somit Optimierungsprobleme formuliert werden können. In Abschnitt 6.2.2 betrachten wir alle Hypothesen, die zur Axiomatisierung der analytischen

Funktionen nötig sind. Schließlich enthält Abschnitt 6.2.3 die Zusammenstellung aller Hypothesen, die für die während der Problemlösung benutzten Polynommanipulationen nötig sind.

### 6.2.1 Eine Theorie der Einheiten

Zur adäquaten Beschreibung des Optimierungsproblems und des Rechnens mit Preisen und Kostenfunktionen definieren wir zunächst eine *Theorie der Einheiten*. Mit deren Hilfe werden wir in der Lage sein, innerhalb  $\Omega$ -MKRP Funktionen handzuhaben, die aus einem numerischen Anteil und einer Benennung bestehen. Zu diesem Zweck erweitern wir die Menge  $\mathcal{T}_B$  der Basistypen unserer Logik  $\mathcal{HOL}$  (siehe hierzu Kapitel 3.1) um zwei neue Typen und definieren bezüglich dieser einige neue Konstanten.

Seien  $\kappa, \xi \in \mathcal{T}_B$ , dann bezeichne  $\xi$  den Typ der Einheiten und  $\kappa$  den Typ der Kosten. Damit können wir nun den benutzten Einheiten für Geld, Liter, Energie und Produkt den Typ  $\xi$  zuordnen, also  $DM, l, kWh, prod \in \Sigma_\xi$ . Kosten mit Typ  $\kappa$  betrachten wir als Paare von reellen Zahlen und Einheiten und Kostenfunktionen mit Typ  $\kappa \rightarrow \kappa$  als reelle Funktionen mit zwei zugeordneten Einheiten. Sei also  $cf \in \Sigma_{\kappa \rightarrow \kappa}$  die Kostenfunktion, dann hat  $cf$  die Form  $cf(f, u, v)$  mit  $f \in \Sigma_{\nu \rightarrow \nu}$  und  $u, v \in \Sigma_\xi$ . Das bedeutet also wir können  $r_1$  und  $r_2$  unseres Problems schreiben als

$$\bullet cf(\lambda d. 4d^2 \Leftrightarrow 18d + 7, l, prod) \quad \bullet cf(\lambda d. 0.5d^2 \Leftrightarrow 1.5d + 0.5, kWh, prod)$$

Als nächstes definieren wir Rechenoperationen für die Kostenfunktionen und axiomatisieren diese so, daß sie sich korrekt beim Rechnen mit Benennungen verhalten. Es seien  $\boxplus, \boxtimes \in \Sigma_{((\kappa \rightarrow \kappa), (\kappa \rightarrow \kappa)) \rightarrow (\kappa \rightarrow \kappa)}$  Addition und Multiplikation auf Kostenfunktionen, dann gelten für diese Operationen die folgenden beiden Axiome:

$$\begin{array}{ll} \text{CF1.} & \text{CF1} \vdash \forall f. \forall g. \forall u. \forall v. (cf(f, u, v) \boxplus cf(g, u, v)) = cf((f \boxplus g), u, v) & (\text{Hyp}) \\ \text{CF2.} & \text{CF2} \vdash \forall f. \forall g. \forall u. \forall v. \forall w. (cf(f, u, v) \boxtimes cf(g, v, w)) = cf((f \boxtimes g), u, w) & (\text{Hyp}) \end{array}$$

In der Problemstellung aus Abschnitt 6.1 sind neben den Kostenfunktionen auch die Preise mit Maßeinheiten versehen. Um diese allerdings in unserer Theorie der Einheiten zu kodieren, müssen wir zunächst betrachten, wie diese in den  $\Omega$ -MKRP-Beweis eingeführt werden. Der Preis eines Produktionsfaktors soll unabhängig von Kosten- oder Verbrauchsfunktionen abgerufen werden können, was bedeutet, daß er eine Voraussetzung darstellt bzw. als Hypothese eingeführt wird. Damit entspricht jeder Preis einer einzelnen Aussage und muß daher eine Funktion mit dem Bildtyp  $o$  sein. Wir definieren also ein Prädikat  $price \in \Sigma_{(\kappa \rightarrow \kappa) \rightarrow o}$ , dessen Argumente eine reelle Funktion und zwei Einheiten sind. Die reelle Funktion könnte durchaus auch durch eine einfache reelle Konstante ausgedrückt

werden, aber zur Demonstration der bisher vorgestellten Algorithmen (insbesondere deren in Kapitel 5) wählen wir hier bewußt die Darstellung eines Preises durch ein konstantes Polynom. Mit Hilfe des Prädikats *price* können nun die Preise  $p_1$  und  $p_2$  der Problemstellung als die folgenden Hypothesen angegeben werden:

$$\begin{array}{lll} \text{P-kWh.} & \text{P-kWh} & \vdash \text{price}(\lambda d. 2, DM, kWh) & (\text{Hyp}) \\ \text{P-l.} & \text{P-l} & \vdash \text{price}(\lambda d. 0.5, DM, l) & (\text{Hyp}) \end{array}$$

Um nun aus Preisen und Kostenfunktionen auch die eigentliche Verbrauchsfunktion berechnen zu können, benutzen wir die in Axiom P definierte Implikation.

$$\text{P.} \quad \text{P} \quad \vdash \forall f. \forall g. \forall u. \forall v. \forall w. [\text{price}(f, u, v) \Rightarrow \text{cf}(g, v, w) = \text{cf}((f \otimes g), u, w)] \quad (\text{Hyp})$$

Mit all diesen Voraussetzungen in unserer Theorie der Einheiten ist es uns jetzt möglich, eine formale Definition des Optimierungsproblems anzugeben. Dabei wird die Optimierung mit einem Prädikat *Opt* mit zwei Argumenten, einer Kostenfunktion und einem Intervall, formalisiert und durch die Hypothese O axiomatisiert.

$$\text{O.} \quad \text{O} \quad \vdash \forall f. \forall I. [\text{Opt}(\text{cf}(f, DM, \text{prod}), I) \Leftrightarrow \exists x. \text{TotMin}(x, f, I)] \quad (\text{Hyp})$$

Diese Hypothese besagt also, daß das Optimierungsproblem bezüglich der Kostenfunktion  $\text{cf}(f, DM, \text{prod})$  und dem Intervall  $I$  genau dann erfüllt werden kann, wenn ein totales Minimum von  $f$  in  $I$  ist<sup>1</sup>. Die Definitionen und Hypothesen für totales Minimum, Intervall, usw. betrachten wir im nächsten Abschnitt.

## 6.2.2 Analytische Theoreme

Hier beginnen wir mit der Definition des totalen Minimums einer Funktion. Dazu machen wir es uns zunutze, daß die Kostenfunktion  $f$  ein Polynom vom Grad  $\leq 2$  ist, und benutzen das folgende Corollar zum Theorem von Rolle ([Heu91] Seite 279 oder [BS91] Seite 269):

*Sei  $f$  ein Polynom vom Grad 2, dann hat  $f$  sein totales Minimum in  $x \in [a, b]$ , genau dann wenn  $f$  in  $x$  ein Minimum hat, und es gilt  $f(a) \geq f(x) \leq f(b)$ .*

Damit erhalten wir in der Syntax unserer Logik die folgende Äquivalenz als Hypothese:

$$\begin{array}{lll} \text{TotMin.} & \text{TotMin} & \vdash \forall f. \forall a. \forall b. \forall x. (\text{TotMin}(x, f, [a, b]) \Leftrightarrow \\ & & (x \in [a, b] \wedge (\text{Min}(f, x) \wedge ((f(x) \leq f(a)) \wedge (f(x) \leq f(b)))))) \end{array} \quad (\text{Hyp})$$

<sup>1</sup>Tatsächlich muß der Prüfling in seiner Examensarbeit das Minimum der Kostenfunktion ausrechnen und überprüfen, ob dieses im gegebenen Intervall für die Leistungsintensität liegt. Ist dies nicht der Fall, so gilt die Aufgabe als nicht lösbar.

In Zeile TotMin benutzen wir mehrere Prädikate, die bislang noch nicht definiert wurden. Diese betrachten wir nun in der Reihenfolge ihres Auftretens auf der rechten Seite der Äquivalenz:  $x \in [a, b]$ ,  $Min(f, x)$  und  $(f(x) \leq f(a))$ . Das erste der Prädikate entspricht in  $\Omega$ -MKRP Syntax  $Interv(a, b, x)$ , dessen drei Argumente jeweils Zahlen vom Typ  $\nu$  sind und das genau dann wahr liefert, wenn  $x$  ein Element des Intervalls  $[a, b]$  ist.  $Min$  ist vom Typ  $((\nu \rightarrow \nu), \nu) \rightarrow o$ , benötigt also als Argumente eine Funktion und eine Zahl und ist genau dann wahr, wenn die Zahl das Minimum der Funktion ist. Das letzte Prädikat,  $\leq$ , beschreibt die übliche *reflexive Halbordnung* auf den reellen Zahlen und ist daher vom Typ  $(\nu, \nu) \rightarrow o$ .

Intv.	Intv	$\vdash \forall a. \forall b. \forall x. [x \in [a, b] \Leftrightarrow [(a \leq x) \wedge (x \leq b)]]$	(Hyp)
$\leq$ -Ax.	$\leq$ -Ax	$\vdash \forall x. \forall y. [(x \leq y) \Leftrightarrow \exists z. [(x + z) = y \wedge (0 \leq z)]]$	(Hyp)
Min.	Min	$\vdash \forall f. \forall x. [((\partial_p(f, 1)x) = 0 \wedge (0 < (\partial_p(\partial_p(f, 1), 1)x))] \Rightarrow Min(f, x))]$	(Hyp)

Abbildung 6.1: Analytische Theoreme

Die Hypothesen für die Axiomatisierung aller drei Prädikate sind angegeben in Abbildung 6.1. Dabei ist in der Zeile Min zu beachten, daß die Schreibweise  $(\partial_p(f, 1)x)$  bedeutet, daß die Ableitung der Funktion  $f$  – eine  $\lambda$ -Abstraktion – angewandt wird auf das Argument  $x$ . Dies geschieht mit der in Kapitel 3.1 definierten  $\beta$ -Reduktion.

### 6.2.3 Voraussetzungen für die Polynombehandlung

Aus der Beschreibung der Problemlösung in Abschnitt 6.1 ist zu ersehen, daß es sich bei Polynommanipulationen um Addition, Multiplikation und Differentiation handelt. Im allgemeinen Fall der Behandlung von univariaten Polynomen benötigen wir für diese Operationen 15 Hypothesen, dargestellt in Abbildung 6.2. Die meisten von diesen wurden bereits in vorangegangenen Kapiteln erläutert (siehe Kapitel 4 und 5). An dieser Stelle sollen nur kurz die drei Hypothesen vorgestellt werden, die für die Polynomdifferentiation notwendig sind: die Zeilen  $\partial$ -const-1,  $\partial$ -mon-1 und  $\partial$ -pol. Dabei besagt  $\partial$ -pol, daß die Ableitung der Summe zweier Polynome bezüglich der Variable an der Stelle  $c$  (siehe hierzu Kapitel 4.1.2) die Summe der einzelnen Ableitungen ist. Die beiden anderen Hypothesen beschreiben die Ableitung eines Monoms bzw. einer Konstanten jeweils bezüglich der Variablen an erster Stelle.

Bei den in Abbildung 6.2 angegebenen Hypothesen handelt es sich um alle, die von den aufgerufenen Algorithmen in SAPPER benutzt werden können. Die Berechnungen im konkreten Fall unseres Beispiels benötigen allerdings nur die folgenden:  $0+$ ,  $C+$ ,  $A+$ ,  $DL$ ,  $M+$ ,  $P+$ ,  $M^*$ ,  $P^*$ ,  $\partial$ -const-1,  $\partial$ -mon-1 und  $\partial$ -pol.

0+.	o+	$\vdash \forall x_{\bullet}(0 + x) = x$	(Hyp)
C+.	c+	$\vdash \forall a_{\bullet} \forall b_{\bullet}(a + b) = (b + a)$	(Hyp)
A+.	A+	$\vdash \forall a_{\bullet} \forall b_{\bullet} \forall c_{\bullet}((a + b) + c) = (a + (b + c))$	(Hyp)
M+.	M+	$\vdash \forall x_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(ax^n + bx^n) = (a + b)x^n$	(Hyp)
P+.	P+	$\vdash \forall p_{\bullet} \forall q_{\bullet}(p \oplus q) = \lambda z_{\bullet}(p(z) + q(z))$	(Hyp)
0*.	o*	$\vdash \forall x_{\bullet}(0 * x) = 0$	(Hyp)
C*.	c*	$\vdash \forall a_{\bullet} \forall b_{\bullet}(a * b) = (b * a)$	(Hyp)
A*.	A*	$\vdash \forall a_{\bullet} \forall b_{\bullet} \forall c_{\bullet}((a * b) * c) = (a * (b * c))$	(Hyp)
M*.	M*	$\vdash \forall x_{\bullet} \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(ax^m * bx^n) = (a * b)x^{(m+n)}$	(Hyp)
P*.	P*	$\vdash \forall p_{\bullet} \forall q_{\bullet}(p \otimes q) = \lambda z_{\bullet}(p(z) * q(z))$	(Hyp)
DL.	DL	$\vdash \forall a_{\bullet} \forall b_{\bullet} \forall c_{\bullet}(a * (b + c)) = ((a * b) + (a * c))$	(Hyp)
DR.	DR	$\vdash \forall a_{\bullet} \forall b_{\bullet} \forall c_{\bullet}((a + b) * c) = ((a * c) + (b * c))$	(Hyp)
$\partial$ -const-1.	$\partial$ -const-1	$\vdash \forall a_{\bullet} \partial_p(\lambda x_{\bullet} ax^0, 1) = \lambda x_{\bullet} 0$	(Hyp)
$\partial$ -mon-1.	$\partial$ -mon-1	$\vdash \forall a_{\bullet} \forall b_{\bullet} \partial_p(\lambda x_{\bullet} ax^b, 1) = \lambda x_{\bullet}(a * b)x^{(b-1)}$	(Hyp)
$\partial$ -pol.	$\partial$ -pol	$\vdash \forall p_{\bullet} \forall q_{\bullet} \forall c_{\bullet} \partial_p((p \oplus q), c) = (\partial_p(p, c) \oplus \partial_p(q, c))$	(Hyp)

Abbildung 6.2: Die notwendigen Hypothesen für die Polynommanipulationen

### 6.3 Problemlösung mit SAPPER

Nach der Vorarbeit in Abschnitt 6.2 ist es uns nun möglich, das Optimierungsproblem in ein Theorem zu fassen und dieses in  $\Omega$ -MKRP unter Zuhilfenahme von SAPPER zu zeigen.

$$\text{THM.} \quad \kappa_1 \quad \vdash \text{Opt}((cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7)), l, prod) \boxplus (Open) \\ cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), kWh, prod)), [1, 7])$$

An dieser Stelle soll der Beweis nur grob skizziert werden. Die im Folgenden erklärten Schritte sind aufgeführt in Abbildung 6.3. Dabei entspricht die Anordnung der Zeilen der Reihenfolge der Inferenzschritte, mit denen das Theorem aus den Hypothesen abgeleitet wird. Während der Konstruktion des Beweises wird aber vom Theorem zu den Hypothesen geschlossen, was zu der angegebenen Zeilennummerierung führt. (Die ausführliche Fassung des Beweises auf ND- und auf Faktenebene befindet sich in Anhang A.)

Zunächst wird die Gesamtkostenfunktion  $f$  innerhalb des Optimierungsprädikats berechnet. Hierzu multiplizieren wir den Preis für Liter mit der entsprechenden Verbrauchsfunktion. Um dies zu erreichen, müssen wir allerdings zunächst die Hypothese P auf die Zeile P-1 anwenden. Dies geschieht in den ersten sechs Inferenzschritten.

In der so erzeugten Zeile L7 wird dann die Polynommultiplikation durch das Computeralgebrasystem vereinfacht. Nach Abschluß der Berechnungen von  $\mu\text{CAS}$  erhalten wir Zeile L40 als neue geplante Zeile. In dieser multiplizieren wir diesmal, auf die gleiche Weise wie oben, den Preis für Strom mit der zweiten Verbrauchsfunktion und vereinfachen erneut mit dem Computeralgebrasystem. Auf die beiden Kostenfunktionen in L80 kann

		⋮	
L193.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p(\partial_p(\lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5))), 1), 1)2))$	(=Subst L247,L249)
L192.	$\mathcal{H}_{14}$	$\vdash (\partial_p(\lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5))), 1)2) = 0$	(=Subst L196,L198)
		⋮	
L158.	$\leq\text{-Ax}$	$\vdash (\lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5)))2 \leq \lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5))7)$	( $\lambda I$ L174)
L157.	$\leq\text{-Ax}$	$\vdash (\lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5)))2 \leq \lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5))1)$	( $\lambda I$ L159)
L155.	$\mathcal{H}_{13}$	$\vdash \text{Min}(\lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5)), 2)$	( $\Rightarrow E$ L191,L190)
L153.	$\mathcal{H}_{18}$	$\vdash 2 \in [1, 7]$	( $\Rightarrow E$ L328,L327)
		⋮	
L144.	$\mathcal{H}_{10}$	$\vdash \text{TotMin}(2, \lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5)), [1, 7])$	( $\Rightarrow E$ L152,L151)
L143.	$\mathcal{H}_{10}$	$\vdash \exists x \text{TotMin}(x, \lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5)), [1, 7])$	( $\exists I$ L144)
		⋮	
L137.	$\mathcal{H}_9$	$\vdash \text{Opt}(cf(\lambda d_{\bullet}(3d^2 + (\Leftrightarrow 12d^1 + 4.5))), DM, prod), [1, 7])$	( $\Rightarrow E$ L143,L142)
		⋮	
L85.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf((\lambda d_{\bullet}(2d^2 + (\Leftrightarrow 9d^1 + 3.5)) \oplus \lambda d_{\bullet}(1d^2 + (\Leftrightarrow 3d^1 + 1))), DM, prod), [1, 7])$	(=Subst L88,L89)
		⋮	
L80.	$\mathcal{H}_6$	$\vdash \text{Opt}((cf(\lambda d_{\bullet}(2d^2 + (\Leftrightarrow 9d^1 + 3.5))), DM, prod) \boxplus cf(\lambda d_{\bullet}(1d^2 + (\Leftrightarrow 3d^1 + 1))), DM, prod), [1, 7])$	(=Subst L84,L85)
		⋮	
L40.	$\mathcal{H}_2$	$\vdash \text{Opt}((cf(\lambda d_{\bullet}(2d^2 + (\Leftrightarrow 9d^1 + 3.5))), DM, prod) \boxplus cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod), [1, 7])$	(=Subst L46,L47)
		⋮	
L7.	$\mathcal{H}_2$	$\vdash \text{Opt}((cf((\lambda d_{\bullet}0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod) \boxplus cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod), [1, 7])$	(=Subst L10,L11)
L6.	$P, P\text{-1}$	$\vdash cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7)), l, prod) = cf((\lambda d_{\bullet}0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod)$	( $\Rightarrow E$ P-1,L5)
L5.	$P$	$\vdash [price(\lambda d_{\bullet}0.5, DM, l) \Rightarrow cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7)), l, prod) = cf((\lambda d_{\bullet}0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod)]$	( $\forall E$ L4)
		⋮	
L1.	$P$	$\vdash \forall g. \forall u. \forall v. \forall w. [price(\lambda d_{\bullet}0.5, u, v) \Rightarrow cf(g, v, w) = cf((\lambda d_{\bullet}0.5 \otimes g), u, w)]$	( $\forall E$ P)
THM.	$\mathcal{H}_1$	$\vdash \text{Opt}((cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d^1 + 7))), l, prod) \boxplus cf(\lambda d_{\bullet}(1/2d^2 + (\Leftrightarrow 3/2d^1 + 1/2))), kWh, prod), [1, 7])$	(=Subst L6,L7)

Abbildung 6.3: Beweisskizze des Optimierungsproblems

jetzt die Hypothese CF1 angewendet werden und auf die daraus resultierende Zeile L85 wieder  $\mu\mathcal{CAS}$ . Die Ausführung dieser Polynomaddition berechnet die Gesamtkostenfunktion innerhalb des Optimierungsprädikats in L137. Nach diesem Rechenschritt ist es nun möglich, die Äquivalenz in der Definition der Optimierung direkt auszunutzen, was Zeile L143 ergibt.

Hier ist nun eine entscheidende Stelle im Beweis. Zum einen werden von da an im Beweis die Hypothesen aus der Theorie der Einheiten nicht mehr benötigt und nur noch von den analytischen Theoremen und den Hypothesen zur Polynombehandlung Gebrauch gemacht. Zum anderen muß hier das tatsächliche Minimum des Polynoms  $\lambda z \cdot (3z^2 + (\Leftrightarrow 12z^1 + 4.5))$  eingeführt werden. Das bedeutet also, daß hier die eingangs geforderte Konstruktivität des Beweises tatsächlich gewährleistet ist. Der numerische Wert für das Minimum wird mittels einer rückwärts angewendeten  $\exists I$  (siehe Kapitel 3.2) in L143 in den Beweis eingeführt. Hierfür kann er interaktiv im  $\mu\mathcal{CAS}$ -Kommandointerpreter innerhalb der  $\Omega$ -MKRP-Umgebung berechnet werden<sup>2</sup>.

Im Rest des Beweises zeigen wir, daß die eingeführte Zahl tatsächlich im gegebenen Intervall liegt, die Kostenfunktion an dieser Stelle ein Minimum hat und den kleinsten Funktionswert annimmt. Auf die Zeile L144 kann nun die Definition des totalen Minimums aus der Hypothese TotMin angewendet werden. Damit erhalten wir drei neue geplante Zeilen: L153, L155, L157 und L158. Von diesen werden L157 und L158 nach einer  $\beta$ -Reduktion mit Hilfe des  $\leq$ -Axioms und L153 durch Anwendung der Definition für Intervalle bewiesen.

Nachdem die Aussagen dieser drei Zeilen – ohne Verwendung von  $\mu\mathcal{CAS}$  – gezeigt wurden, bleibt L155 die einzige noch offene Zeile unseres Beweises. Um nun zu bestätigen, daß 2 tatsächlich ein Minimum unserer Funktion ist, benutzen wir die Hypothese Min. Dies liefert als neue zu beweisende Ziele L192 und L193. Zur Überprüfung der Aussagen dieser Zeilen, nämlich daß die erste Ableitung der Funktion  $\lambda z \cdot (3z^2 + (\Leftrightarrow 12z^1 + 4.5))$  an der Stelle  $d = 2$  gleich 0 bzw. die zweite Ableitung größer als 0 ist, benutzen wir erneut unser Computeralgebrasystem im Verbose-Mode.

Nach diesen letzten Schritten ergeben sich keine neuen offenen Zeilen, und der Beweis des Optimierungsproblems ist abgeschlossen. Hierfür wurde  $\mu\mathcal{CAS}$  nicht nur zur Erzeugung von Teilbeweisen im Verbose-Mode, sondern auch zum Berechnen des Minimums der Kostenfunktion selbst eingesetzt. Als Resultat erhalten wir aber einen Beweis, der rein im ND-Kalkül ist und damit völlig unabhängig vom Computeralgebrasystem und seinen Berechnungen. Benutzt man  $\mu\mathcal{CAS}$  während des Beweises nicht in seinem Verbose-Mode, so ergibt sich ein kürzerer Beweis von 92 Schritten. Da  $\Omega$ -MKRP eine interaktive Beweisumge-

---

<sup>2</sup>Eine Automatisierung einer solchen Berechnung könnte durch die Einbindung von Aufrufen an das Computeralgebrasystem in  $\Omega$ -MKRPs Planungskomponente geschehen (siehe hierzu Kapitel 7).

---

bung ist, entspricht diese Anzahl der Beweisschritte in etwa auch der Zahl der Benutzerinteraktionen.

## Kapitel 7

# Zusammenfassung und Ausblick

In dieser Arbeit haben wir mit SAPPER ein System vorgestellt, das es ermöglicht, aus Computeralgebraalgorithmen Beweispläne zu extrahieren, mit deren Hilfe ND-Beweise für Berechnungen von der Algorithmen erstellt werden können. Damit unterscheidet sich dieser Ansatz zur Integration von Deduktion und Computeralgebra von denen in Kapitel 2.2 vorgestellten insoweit, als er es nicht nur erlaubt, Berechnungen eines CAS während eines Beweises zu benutzen, sondern gleichzeitig auch noch die Korrektheit dieser garantiert. Mit SAPPER ist es möglich, Rechnungen in Beweisen automatisch ausführen zu lassen, ohne dabei die Forderung zu verletzen, daß der Beweis ausschließlich in den Regeln eines festvorgegebenen ND-Kalküls zu führen ist, um ihn einfach verifizieren zu können. Dies bedeutet, daß die so erstellten Beweise einfach kommunizierbar sind, also in jeder anderen Beweisumgebung, die mit dem ND-Kalkül arbeitet nachvollziehbar sind.

Bei der Konzeption von SAPPER wurden mehrere wichtige Designkriterien berücksichtigt:

- Wie wir in Kapitel 4.3 gesehen haben, ist der Aufbau des Systems generisch. Zum einen ist es unabhängig von den im einzelnen verwendeten Systemkomponenten, also dem Deduktions- und dem Computeralgebrasystem, zum anderen kann SAPPER leicht für neu anzuschließende CAS und neu zu integrierende Computeralgebraalgorithmen erweitert werden.
- SAPPER stellt die Voraussetzungen bereit, um CAS auf zwei verschiedene Arten einzusetzen. Neben dem eigentlichen Zweck von SAPPER, CAS als Beweisplaner zu nutzen, können auch das CAS als Black-Box aufgerufen werden und damit Berechnungen innerhalb eines Beweises in einem Schritt ausgeführt werden (Kapitel 4.3.1).
- Darüber hinaus wurde der SAPPER zugrunde liegende Taktikmechanismus so realisiert, daß die Beweispläne eine hierarchische Struktur erhalten und somit auf verschiedenen

---

Ebenen der Abstraktion mit Erklärungen versehen werden können (Kapitel 4.3.2).

Wir haben in Kapitel 5 gezeigt, daß die Extraktion der Beweispläne aus Computeralgebraalgorithmen bei einfachen Algorithmenschemata relativ problemlos erfolgen kann, ohne das Rechenverhalten des Algorithmus zu verändern. Allerdings war es nicht möglich, elaborierte Algorithmen wie die Polynommultiplikation von KARATSUBA und OFMAN durch geeignete Taktikausgaben zu erweitern. Dies läßt es naheliegend erscheinen, einfache iterative und rekursive Algorithmen mit einem Verbose-Mode zu versehen und zur Beweisplanung zu benutzen.

Das System wurde erfolgreich zum Lösen eines Optimierungsproblems in Kapitel 6 eingesetzt. Der dafür erstellte Beweis umfaßt mehr als 350 Zeilen und ist damit weder durch einen im ND-Kalkül arbeitenden automatischen Beweiser noch mit vertretbarem Aufwand interaktiv durch einen Benutzer von  $\Omega$ -MKRP erstellbar. Dies zeigt die Nützlichkeit von SAPPER bei der Lösung derartiger Probleme.

Als allgemeines Fazit aus den Betrachtungen der vorangegangenen Kapiteln läßt sich wohl sagen, daß Computeralgebrasysteme im Rahmen eines mathematischen Assistenzsystems wie  $\Omega$ -MKRP durchaus einen größeren Beitrag leisten können, als das bloße Liefern von Ergebnissen für symbolische Rechnungen. Zu diesem Zwecke müßten sie jedoch im Einklang mit den im Deduktionssystem bekannten Taktiken erweitert werden. Um nun die Rechenleistung des CAS nicht einzuschränken, könnte ein solches System zweigeteilten Aufbau haben. Zum einen sollte unabhängiges Rechnen mit effizienten Algorithmen und Datenstrukturen möglich sein, und zum anderen sollte es erlauben, aus einfacheren Algorithmen mittels des Verbose-Modes Erklärungen für Zwischenschritte von Berechnungen zu erhalten.

Aus dem vorgestellten System SAPPER und den Ergebnissen dieser Arbeit ergeben sich Ausblicke, Fragen und Probleme, die einer zukünftigen Bearbeitung bedürfen.

Zunächst wäre noch einiges an Verbesserungen am vorgestellten System wünschenswert. So müßte für größere Anwendungen die Anzahl der zur Beweisplanung verfügbaren Computeralgebraalgorithmen erweitert werden oder aber ein bereits existierendes großes Computeralgebrasystem mit einem Verbose-Mode versehen und in SAPPER eingepaßt werden. Außerdem sind einige Erweiterungen und Veränderungen an  $\Omega$ -MKRP selbst notwendig. Eine davon wurde bereits in Kapitel 4.5 angesprochen, der Aufbau einer nach Theorien geordneten mathematischen Wissensbasis. Eine weitere ist eine neue Datenstruktur für Pläne und Beweise, so daß es möglich wird, die von SAPPER erstellten hierarchischen Beweispläne auch direkt in  $\Omega$ -MKRP zu importieren und darzustellen.

Eine weitere mögliche Interaktion zwischen  $\Omega$ -MKRP und einem CAS wäre die Integration von CAS-Aufrufen in der Planungskomponente. Dabei könnten innerhalb der Methoden,

mit denen  $\Omega$ -MKRP Beweispläne konstruiert, Ergebnisse von CAS-Berechnungen genutzt werden, um z.B. damit existenzquantifizierte Variablen zu ersetzen. In unserem Optimierungsbeispiel könnte der Planer dann mit Hilfe des CAS das entsprechende Minimum der Kostenfunktion berechnen lassen. Darüber hinaus sollten vom CAS erstellte Teilpläne ebenfalls in die von  $\Omega$ -MKRP geplanten eingepaßt werden.

Um dem System einen größeren Vorrat an Algorithmen zur Beweisplanung zu verschaffen, müssen weitere mit einem Verbose-Mode versehen und entsprechende Taktiken implementiert werden. Um dies nicht von Hand tun zu müssen, wäre ein theoretischer Rahmen wünschenswert, der Regeln bereitstellt, nach welchen Algorithmen erweitert und in Taktiken übersetzt werden können. Außerdem wäre es auch denkbar, verschiedene Modi für die Extraktion solcher Informationen zur Verfügung zu stellen, die jeweils verschieden genau den Vorgang in einem Algorithmus protokollieren.

$\Omega$ -MKRP ist eine interaktive Beweisumgebung, in der der Benutzer darüber entscheidet, wann und wie ein CAS eingesetzt werden soll. Für ein Verbinden von SAPPER mit stärker automatisierten, taktischen Theorembeweisern, müssen Kriterien gefunden werden, wann ein CAS aufgerufen werden kann und wie die erstellten Beweispläne eingesetzt werden können.

SAPPER wurde im Rahmen dieser Arbeit auf noch relativ einfache Optimierungsprobleme angewendet. Dies kann als Ausgangspunkt dienen, um andere wirtschaftswissenschaftliche Problemstellungen zu lösen und das System zu einem KI-Werkzeug für dieses Gebiet mit nachprüfbarer Korrektheit weiterzuentwickeln. Eine andere Klasse von Problemen für die SAPPER eingesetzt werden kann, wären Beweise für die Anzahl geschlossener Terme – also Terme ohne freien Variablen (siehe Kapitel 3.1) – in getypten  $\lambda$ -Kalkülen mit einer abzählbaren Menge von Basistypen. Diese Anzahl wird im wesentlichen mit einem Algorithmus ermittelt, der den Fixpunkt eines Gleichungssystems berechnet, das aus Polynomen mit Koeffizienten in den natürlichen Zahlen besteht [Zai95].

# Anhang A

## Beweis des Optimierungsproblems

In diesem Anhang befindet sich der vollständige Beweis für das Optimierungsproblem aus Kapitel 6, auf den beiden in  $\Omega$ -MKRP verfügbaren Darstellungsebenen: der Fakten- und der ND-Kalkülebene. Dabei ist zu beachten, daß in beiden Darstellungen die in Kapitel 4.1.1 eingeführten reellen Zahlen benutzt werden. Dies hat zur Folge, daß der ND-Beweis eigentlich noch nicht vollständig auf der Kalkülebene ist, sondern in ihm immer noch die Taktik SIMPLIFY-NUM benutzt wird.

In den folgenden beiden Abschnitten werden zunächst der Faktenebenenbeweis und dann der ND-Beweis angegeben. Diesen werden jeweils die Erläuterungen von bemerkenswerten Stellen in den Beweisen vorangestellt. (Für die Beschreibung der verwendeten Hypothesen vergleiche Kapitel 6.2.)

### A.1 Beweis auf der Faktenebene

Bei den Beweisen auf der Faktenebene, die bisher in dieser Arbeit vorgestellt wurden, waren die Formeln der einzelnen Zeilen jeweils aus einer leeren Hypothesenliste abgeleitet (siehe hierzu auch Kapitel 3.3.1). Im vorliegenden Beweis ist dies nicht der Fall; hier ist unser Theorem THM abhängig von den beiden Hypothesen, die den Preis der beiden Kostenfaktoren beinhalten. Diese werden im Gegensatz zu allen anderen Hypothesen während des Beweises nicht direkt auf das Theorem angewendet, sondern dienen als Voraussetzung für die Benutzung des Preisaxioms P in den Zeilen L6 und L46. Dies wird auch ausgedrückt durch die Begründung dieser beiden Zeilen (P P-l) und (P P-kWh), was bedeutet: P angewendet auf den Preis für Kühlmittel bzw. Strom.

Eine Begründung der Form  $(R L_1, \dots, L_n)$  besagt allgemein auf der Faktenebene, daß die begründete Zeile durch die Anwendung der Definition oder des Axioms R auf die

Zeilen  $L_1, \dots, L_n$  entsteht. So wird zum Beispiel die Formel in der Zeile

$$L144. \quad \vdash \text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7]) \quad (\text{TotMin } L153, L155, L157, L158)$$

mit der Anwendung der Definition in der Hypothese TotMin auf die Zeilen L153, L155, L157 und L158 begründet.

Einige weitere Hypothesen kommen innerhalb des Beweises in den Zeilen L171, L186, L321, L340 und L352 vor. Sie dienen jeweils, wie zum Beispiel

$$L171. \quad L171 \quad \vdash (0 \leq 3) \quad (\text{Hyp})$$

zur Begründung der Aussage, daß 3 größer oder gleich 0 ist.

P-kWh.	P-kWh	$\vdash \text{price}(\lambda d_{\bullet} 2, DM, kWh)$	(Hyp)
P-1.	P-1	$\vdash \text{price}(\lambda d_{\bullet} 0.5, DM, l)$	(Hyp)
THM.	P-kWh, P-1	$\vdash \text{Opt}((cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), l, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L6,L7)
L6.	P-1	$\vdash cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), l, prod) =$ $cf((\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod)$	(P P-1)
L7.	P-kWh	$\vdash \text{Opt}((cf((\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L10,L11)
L10.		$\vdash (\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))) = \lambda z_{\bullet}(0.5 * (4z^2 + (\Leftrightarrow 18z + 7)))$	(P*)
L11.	P-kWh	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(0.5 * (4z^2 + (\Leftrightarrow 18z + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L14,L15)
L14.		$\vdash (0.5 * (4z^2 + (\Leftrightarrow 18z + 7))) = ((0.5 * 4z^2) + (0.5 * (\Leftrightarrow 18z + 7)))$	(DL)
L15.	P-kWh	$\vdash \text{Opt}((cf(\lambda z_{\bullet}((0.5 * 4z^2) + (0.5 * (\Leftrightarrow 18z + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L21,L22)
L21.		$\vdash (0.5 * 4z^2) = 2z^2$	(M*)
L22.	P-kWh	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (0.5 * (\Leftrightarrow 18z + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L25,L26)
L25.		$\vdash (0.5 * (\Leftrightarrow 18z + 7)) = ((0.5 * \Leftrightarrow 18z) + (0.5 * 7))$	(DL)
L26.	P-kWh	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + ((0.5 * \Leftrightarrow 18z) + (0.5 * 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L32,L33)
L32.		$\vdash (0.5 * \Leftrightarrow 18z) = \Leftrightarrow 9z$	(M*)
L33.	P-kWh	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + (0.5 * 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L39,L40)
L39.		$\vdash (0.5 * 7) = 3.5$	(M*)
L40.	P-kWh	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L46,L47)
L46.	P-kWh	$\vdash cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod) =$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, prod)$	(P P-kWh)
L47.		$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, prod)), [1, 7])$	(=Subst L50,L51)
L50.		$\vdash (\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))) =$ $\lambda z_{\bullet}(2 * (0.5z^2 + (\Leftrightarrow 1.5z + 0.5)))$	(P*)
L51.		$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(2 * (0.5z^2 + (\Leftrightarrow 1.5z + 0.5))), DM, prod)), [1, 7])$	(=Subst L54,L55)
L54.		$\vdash (2 * (0.5z^2 + (\Leftrightarrow 1.5z + 0.5))) = ((2 * 0.5z^2) + (2 * (\Leftrightarrow 1.5z + 0.5)))$	(DL)
L55.		$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}((2 * 0.5z^2) + (2 * (\Leftrightarrow 1.5z + 0.5))), DM, prod)), [1, 7])$	(=Subst L61,L62)

L61.	$\vdash (2 * 0.5z^2) = z^2$	(M*)
L62.	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (2 * (\Leftrightarrow 1.5z + 0.5))), DM, prod)), [1, 7])$	(=Subst L65,L66)
L65.	$\vdash (2 * (\Leftrightarrow 1.5z + 0.5)) = ((2 * \Leftrightarrow 1.5z) + (2 * 0.5))$	(DL)
L66.	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + ((2 * \Leftrightarrow 1.5z) + (2 * 0.5))), DM, prod)), [1, 7])$	(=Subst L72,L73)
L72.	$\vdash (2 * \Leftrightarrow 1.5z) = \Leftrightarrow 3z$	(M*)
L73.	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + (2 * 0.5))), DM, prod)), [1, 7])$	(=Subst L79,L80)
L79.	$\vdash (2 * 0.5) = 1$	(M*)
L80.	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod)), [1, 7])$	(=Subst L84,L85)
L84.	$\vdash (cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod) =$ $cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod)$	(CF1)
L85.	$\vdash Opt(cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus$ $\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod), [1, 7])$	(=Subst L88,L89)
L88.	$\vdash (\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))) =$ $\lambda z_{\bullet}((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + (\Leftrightarrow 3z + 1)))$	(P+)
L89.	$\vdash Opt(cf(\lambda z_{\bullet}((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + (\Leftrightarrow 3z + 1))), DM, prod),$ $[1, 7])$	(=Subst L92,L93)
L92.	$\vdash (((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) + (\Leftrightarrow 3z + 1)) =$ $((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + (\Leftrightarrow 3z + 1)))$	(A+)
L93.	$\vdash Opt(cf(\lambda z_{\bullet}(((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) + (\Leftrightarrow 3z + 1))), DM, prod),$ $[1, 7])$	(=Subst L95,L96)
L95.	$\vdash ((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) = (z^2 + (2z^2 + (\Leftrightarrow 9z + 3.5)))$	(C+)
L96.	$\vdash Opt(cf(\lambda z_{\bullet}((z^2 + (2z^2 + (\Leftrightarrow 9z + 3.5))) + (\Leftrightarrow 3z + 1))), DM, prod),$ $[1, 7])$	(=Subst L99,L100)
L99.	$\vdash ((z^2 + 2z^2) + (\Leftrightarrow 9z + 3.5)) = (z^2 + (2z^2 + (\Leftrightarrow 9z + 3.5)))$	(A+)
L100.	$\vdash Opt(cf(\lambda z_{\bullet}(((z^2 + 2z^2) + (\Leftrightarrow 9z + 3.5)) + (\Leftrightarrow 3z + 1))), DM, prod),$ $[1, 7])$	(=Subst L105,L106)
L105.	$\vdash (z^2 + 2z^2) = 3z^2$	(M+)
L106.	$\vdash Opt(cf(\lambda z_{\bullet}((3z^2 + (\Leftrightarrow 9z + 3.5)) + (\Leftrightarrow 3z + 1))), DM, prod), [1, 7])$	(=Subst L109,L110)
L109.	$\vdash ((3z^2 + (\Leftrightarrow 9z + 3.5)) + (\Leftrightarrow 3z + 1)) =$ $(3z^2 + ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + 1)))$	(A+)
L110.	$\vdash Opt(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + 1))), DM, prod), [1, 7])$	(=Subst L113,L114)
L113.	$\vdash (((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) + 1) = ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + 1))$	(A+)
L114.	$\vdash Opt(cf(\lambda z_{\bullet}(3z^2 + (((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) + 1))), DM, prod), [1, 7])$	(=Subst L116,L117)
L116.	$\vdash ((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) = (\Leftrightarrow 3z + (\Leftrightarrow 9z + 3.5))$	(C+)
L117.	$\vdash Opt(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 3z + (\Leftrightarrow 9z + 3.5)) + 1))), DM, prod), [1, 7])$	(=Subst L120,L121)
L120.	$\vdash ((\Leftrightarrow 3z + \Leftrightarrow 9z) + 3.5) = (\Leftrightarrow 3z + (\Leftrightarrow 9z + 3.5))$	(A+)
L121.	$\vdash Opt(cf(\lambda z_{\bullet}(3z^2 + (((\Leftrightarrow 3z + \Leftrightarrow 9z) + 3.5) + 1))), DM, prod), [1, 7])$	(=Subst L126,L127)
L126.	$\vdash (\Leftrightarrow 3z + \Leftrightarrow 9z) = \Leftrightarrow 12z$	(M+)
L127.	$\vdash Opt(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 12z + 3.5) + 1))), DM, prod), [1, 7])$	(=Subst L130,L131)
L130.	$\vdash ((\Leftrightarrow 12z + 3.5) + 1) = (\Leftrightarrow 12z + (3.5 + 1))$	(A+)
L131.	$\vdash Opt(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + (3.5 + 1))), DM, prod), [1, 7])$	(=Subst L136,L137)
L136.	$\vdash (3.5 + 1) = 4.5$	(M+)

L137.		$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), DM, prod), [1, 7])$	(O L144)
L144.		$\vdash \text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])$	(TotMin L153,L155, L157,L158)
L153.		$\vdash 2 \in [1, 7]$	(Intv L329,L330)
L155.		$\vdash \text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2)$	(Min L192,L193)
L157.		$\vdash (\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1$	( $\lambda I$ L159)
L158.		$\vdash (\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7$	( $\lambda I$ L174)
L159.		$\vdash (((3 * 2^2) + ((\Leftrightarrow 12 * 2^1) + 4.5)) \leq ((3 * 1^2) + ((\Leftrightarrow 12 * 1^1) + 4.5)))$	(Simplify-Num L160)
L160.		$\vdash (\Leftrightarrow 7.5 \leq ((3 * 1^2) + ((\Leftrightarrow 12 * 1^1) + 4.5)))$	(Simplify-Num L161)
L161.		$\vdash (\Leftrightarrow 7.5 \leq \Leftrightarrow 4.5)$	( $\leq$ -AX L167)
L167.		$\vdash \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = \Leftrightarrow 4.5 \wedge (0 \leq z)]$	( $\exists I$ L168)
L168.		$\vdash [(\Leftrightarrow 7.5 + 3) = \Leftrightarrow 4.5 \wedge (0 \leq 3)]$	( $\wedge I$ L169,L171)
L169.		$\vdash (\Leftrightarrow 7.5 + 3) = \Leftrightarrow 4.5$	(Simplify-Num L173)
L171.	L171	$\vdash (0 \leq 3)$	(Hyp)
L173.		$\vdash \Leftrightarrow 4.5 = \Leftrightarrow 4.5$	(=I)
L174.		$\vdash (((3 * 2^2) + ((\Leftrightarrow 12 * 2^1) + 4.5)) \leq ((3 * 7^2) + ((\Leftrightarrow 12 * 7^1) + 4.5)))$	(Simplify-Num L175)
L175.		$\vdash (\Leftrightarrow 7.5 \leq ((3 * 7^2) + ((\Leftrightarrow 12 * 7^1) + 4.5)))$	(Simplify-Num L176)
L176.		$\vdash (\Leftrightarrow 7.5 \leq 67.5)$	( $\leq$ -AX L182)
L182.		$\vdash \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = 67.5 \wedge (0 \leq z)]$	( $\exists I$ L183)
L183.		$\vdash [(\Leftrightarrow 7.5 + 75) = 67.5 \wedge (0 \leq 75)]$	( $\wedge I$ L184,L186)
L184.		$\vdash (\Leftrightarrow 7.5 + 75) = 67.5$	(Simplify-Num L188)
L186.	L186	$\vdash (0 \leq 75)$	(Hyp)
L188.		$\vdash 67.5 = 67.5$	(=I)
L192.		$\vdash (\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1)2) = 0$	(=Subst L196,L198)
L193.		$\vdash (0 < (\partial_p(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1), 1)2))$	(=Subst L247,L249)
L196.		$\vdash (\lambda z_{\bullet}3z^2 \oplus \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)) = \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))$	(P+)
L198.		$\vdash (\partial_p((\lambda z_{\bullet}3z^2 \oplus \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)), 1)2) = 0$	(=Subst L203,L204)
L203.		$\vdash (\partial_p((\lambda z_{\bullet}3z^2 \oplus \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)), 1)2) =$ $((\partial_p(\lambda z_{\bullet}3z^2, 1) \oplus \partial_p(\lambda z_{\bullet}(\Leftrightarrow 12z + 4.5), 1))2)$	( $\partial$ -pol)
L204.		$\vdash ((\partial_p(\lambda z_{\bullet}3z^2, 1) \oplus \partial_p(\lambda z_{\bullet}(\Leftrightarrow 12z + 4.5), 1))2) = 0$	(=Subst L208,L209)
L208.		$\vdash \partial_p(\lambda z_{\bullet}3z^2, 1) = \lambda x_{\bullet}6x$	( $\partial$ -mon-1)
L209.		$\vdash ((\lambda x_{\bullet}6x \oplus \partial_p(\lambda z_{\bullet}(\Leftrightarrow 12z + 4.5), 1))2) = 0$	(=Subst L212,L214)
L212.		$\vdash (\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet}4.5) = \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)$	(P+)
L214.		$\vdash ((\lambda x_{\bullet}6x \oplus \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet}4.5), 1))2) = 0$	(=Subst L217,L218)
L217.		$\vdash \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet}4.5), 1) = (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet}4.5, 1))$	( $\partial$ -pol)
L218.		$\vdash ((\lambda x_{\bullet}6x \oplus (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet}4.5, 1)))2) = 0$	(=Subst L222,L223)
L222.		$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) = \lambda x_{\bullet} \Leftrightarrow 12$	( $\partial$ -mon-1)
L223.		$\vdash ((\lambda x_{\bullet}6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \partial_p(\lambda z_{\bullet}4.5, 1)))2) = 0$	(=Subst L225,L226)
L225.		$\vdash \partial_p(\lambda z_{\bullet}4.5, 1) = \lambda x_{\bullet}0$	( $\partial$ -const-1)
L226.		$\vdash ((\lambda x_{\bullet}6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet}0))2) = 0$	(=Subst L229,L230)
L229.		$\vdash (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet}0) = \lambda z_{\bullet}(\Leftrightarrow 12 + 0)$	(P+)
L230.		$\vdash ((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet}(\Leftrightarrow 12 + 0))2) = 0$	(=Subst L232,L233)
L232.		$\vdash (\Leftrightarrow 12 + 0) = (0 + \Leftrightarrow 12)$	(C+)
L233.		$\vdash ((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet}(0 + \Leftrightarrow 12))2) = 0$	(=Subst L234,L235)
L234.		$\vdash (0 + \Leftrightarrow 12) = \Leftrightarrow 12$	(0+)
L235.		$\vdash ((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet} \Leftrightarrow 12)2) = 0$	(=Subst L240,L241)
L240.		$\vdash ((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet} \Leftrightarrow 12)2) = \lambda z_{\bullet}(6z + \Leftrightarrow 12)2$	(P+)

L241.	$\vdash \lambda z_{\bullet}(6z + \Leftrightarrow 12)2 = 0$	( $\lambda I$ L242)
L242.	$\vdash ((6 * 2^1) + \Leftrightarrow 12) = 0$	(Simplify-Num L244)
L244.	$\vdash 0 = 0$	(=I)
L247.	$\vdash (\lambda z_{\bullet}3z^2 \oplus \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)) = \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))$	(P+)
L249.	$\vdash (0 < (\partial_p(\partial_p((\lambda z_{\bullet}3z^2 \oplus \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)), 1), 1)2))$	(=Subst L252,L253)
L252.	$\vdash \partial_p((\lambda z_{\bullet}3z^2 \oplus \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)), 1) =$ $(\partial_p(\lambda z_{\bullet}3z^2, 1) \oplus \partial_p(\lambda z_{\bullet}(\Leftrightarrow 12z + 4.5), 1))$	( $\partial$ -pol)
L253.	$\vdash (0 < (\partial_p((\partial_p(\lambda z_{\bullet}3z^2, 1) \oplus \partial_p(\lambda z_{\bullet}(\Leftrightarrow 12z + 4.5), 1)), 1)2))$	(=Subst L257,L258)
L257.	$\vdash \partial_p(\lambda z_{\bullet}3z^2, 1) = \lambda x_{\bullet}6x$	( $\partial$ -mon-1)
L258.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus \partial_p(\lambda z_{\bullet}(\Leftrightarrow 12z + 4.5), 1)), 1)2))$	(=Subst L261,L263)
L261.	$\vdash (\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet}4.5) = \lambda z_{\bullet}(\Leftrightarrow 12z + 4.5)$	(P+)
L263.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet}4.5), 1)), 1)2))$	(=Subst L266,L267)
L266.	$\vdash \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet}4.5), 1) = (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet}4.5, 1))$	( $\partial$ -pol)
L267.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet}4.5, 1))), 1)2))$	(=Subst L271,L272)
L271.	$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) = \lambda x_{\bullet} \Leftrightarrow 12$	( $\partial$ -mon-1)
L272.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \partial_p(\lambda z_{\bullet}4.5, 1))), 1)2))$	(=Subst L274,L275)
L274.	$\vdash \partial_p(\lambda z_{\bullet}4.5, 1) = \lambda x_{\bullet}0$	( $\partial$ -const-1)
L275.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet}0)), 1)2))$	(=Subst L278,L279)
L278.	$\vdash (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet}0) = \lambda z_{\bullet}(\Leftrightarrow 12 + 0)$	(P+)
L279.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet}(\Leftrightarrow 12 + 0)), 1)2))$	(=Subst L281,L282)
L281.	$\vdash (\Leftrightarrow 12 + 0) = (0 + \Leftrightarrow 12)$	(C+)
L282.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet}(0 + \Leftrightarrow 12)), 1)2))$	(=Subst L283,L284)
L283.	$\vdash (0 + \Leftrightarrow 12) = \Leftrightarrow 12$	(0+)
L284.	$\vdash (0 < (\partial_p((\lambda x_{\bullet}6x \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)2))$	(=Subst L287,L288)
L287.	$\vdash (\lambda x_{\bullet}6x \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet}(6z + \Leftrightarrow 12)$	(P+)
L288.	$\vdash (0 < (\partial_p(\lambda z_{\bullet}(6z + \Leftrightarrow 12), 1)2))$	(=Subst L291,L293)
L291.	$\vdash (\lambda z_{\bullet}6z \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet}(6z + \Leftrightarrow 12)$	(P+)
L293.	$\vdash (0 < (\partial_p((\lambda z_{\bullet}6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)2))$	(=Subst L298,L299)
L298.	$\vdash (\partial_p((\lambda z_{\bullet}6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)2) =$ $((\partial_p(\lambda z_{\bullet}6z, 1) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2)$	( $\partial$ -pol)
L299.	$\vdash (0 < ((\partial_p(\lambda z_{\bullet}6z, 1) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2))$	(=Subst L303,L304)
L303.	$\vdash \partial_p(\lambda z_{\bullet}6z, 1) = \lambda x_{\bullet}6$	( $\partial$ -mon-1)
L304.	$\vdash (0 < ((\lambda x_{\bullet}6 \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2))$	(=Subst L306,L307)
L306.	$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1) = \lambda x_{\bullet}0$	( $\partial$ -const-1)
L307.	$\vdash (0 < ((\lambda x_{\bullet}6 \oplus \lambda x_{\bullet}0)2))$	(=Subst L312,L313)
L312.	$\vdash ((\lambda x_{\bullet}6 \oplus \lambda x_{\bullet}0)2) = \lambda z_{\bullet}(6 + 0)2$	(P+)
L313.	$\vdash (0 < \lambda z_{\bullet}(6 + 0)2)$	(=Subst L315,L316)
L315.	$\vdash (6 + 0) = (0 + 6)$	(C+)
L316.	$\vdash (0 < \lambda z_{\bullet}(0 + 6)2)$	(=Subst L317,L318)
L317.	$\vdash (0 + 6) = 6$	(0+)
L318.	$\vdash (0 < \lambda z_{\bullet}(6)2)$	( $\lambda I$ L321)
L321.	L321 $\vdash (0 < 6)$	(Hyp)
L329.	$\vdash (1 \leq 2)$	( $\leq$ -AX L336)
L330.	$\vdash (2 \leq 7)$	( $\leq$ -AX L348)
L336.	$\vdash \exists z_{\bullet}[(1 + z) = 2 \wedge (0 \leq z)]$	( $\exists I$ L337)
L337.	$\vdash [(1 + 1) = 2 \wedge (0 \leq 1)]$	( $\lambda I$ L338,L340)
L338.	$\vdash (1 + 1) = 2$	(Simplify-Num L342)

L340.	L340	$\vdash (0 \leq 1)$	(Hyp)
L342.		$\vdash 2 = 2$	(=I)
L348.		$\vdash \exists z_{\bullet}[(2 + z) = 7 \wedge (0 \leq z)]$	( $\exists I$ L349)
L349.		$\vdash [(2 + 5) = 7 \wedge (0 \leq 5)]$	( $\wedge I$ L350, L352)
L350.		$\vdash (2 + 5) = 7$	(Simplify-Num L354)
L352.	L352	$\vdash (0 \leq 5)$	(Hyp)
L354.		$\vdash 7 = 7$	(=I)

## A.2 ND-Beweis

In dem hier aufgeführten ND-Beweis machen wir neben den in Kapitel 3.2 definierten Inferenzregeln des ND-Kalküls auch Gebrauch von der in Definition 3.19 eingeführten Extensionalität und den  $\lambda$ -Reduktionen aus Definition 3.9. Die benutzten Regeln sind im einzelnen

- AB für  $\alpha$ -Reduktion,
- $\lambda E$ ,  $\lambda I$  für  $\beta\eta$ -Reduktion vorwärts bzw. rückwärts,
- Ext-I für Extensionalitätseinführung.

Zur Verdeutlichung dieser Regeln betrachten wir ihre Anwendung im folgenden Beweisteil:

L304.	$\mathcal{H}_{16}$	$\vdash (0 < ((\lambda x_{\bullet} 6 \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2))$	(=Subst L306, L307)
L305.	$\partial\text{-const-1}$	$\vdash \partial_p(\lambda x_{\bullet} \Leftrightarrow 12, 1) = \lambda x_{\bullet} 0$	( $\forall E$ $\partial\text{-const-1}$ )
L306.	$\partial\text{-const-1}$	$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1) = \lambda x_{\bullet} 0$	(AB L305)
L307.	$\mathcal{H}_{17}$	$\vdash (0 < ((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)2))$	(=Subst L312, L313)
L308.	P+	$\vdash \forall q_{\bullet}(\lambda x_{\bullet} 6 \oplus q) = \lambda z_{\bullet}(\lambda x_{\bullet}(6)z + q(z))$	( $\forall E$ P+)
L309.	P+	$\vdash (\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet}(\lambda x_{\bullet}(6)z + \lambda x_{\bullet}(0)z)$	( $\forall E$ L308)
L310.	P+	$\vdash (\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet}(6 + 0)$	( $\lambda E$ L309)
L311.	P+	$\vdash \forall x_{\bullet}((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)x) = \lambda z_{\bullet}(6 + 0)x$	(Ext-I L310)
L312.	P+	$\vdash ((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)2) = \lambda z_{\bullet}(6 + 0)2$	( $\forall E$ L311)
L313.	$0+, C+$	$\vdash (0 < \lambda z_{\bullet}(6 + 0)2)$	(=Subst L315, L316)

Zeile L306 entsteht aus L305 durch Umbenennung der Variable  $x$  im Term  $\lambda x_{\bullet} \Leftrightarrow 12$  in  $z$ . Damit können wir diesen in Zeile L304 durch eine =Subst ersetzen. Die Zeile L310 entsteht aus L309 durch  $\beta$ -Reduktion in den Termen  $\lambda x_{\bullet}(6)z$  und  $\lambda x_{\bullet}(0)z$ . Um auch den Term  $((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)2)$  in L307 durch  $\lambda z_{\bullet}(6 + 0)2$  ersetzen zu können, benötigen wir die Extensionalität der  $\oplus$ -Funktion in Zeile L311. Warum dies so sein muß, wird allerdings erst in der Präfixschreibweise von  $\Omega\text{-MKRP}$  deutlich, denn hier lautet der zu ersetzende Term

( $\oplus \lambda x.6 \lambda x.0 \ 2$ ) und die Gleichung in Zeile L310 ( $\oplus \lambda x.6 \lambda x.0$ ) =  $\lambda z. (+ 6 \ 0)$ . Da es aber nicht möglich ist, eine Gleichheitssubstitution nur bezüglich einer Funktion und einem Teil ihrer Argumente durchzuführen, müssen wir zunächst mit Hilfe der Extensionalität die 2 als weiteres Argument einführen.

Eine weitere Besonderheit bei der Darstellung des folgenden ND-Beweises ist, daß die meisten Hypothesenlisten in den einzelnen Zeilen platzsparend durch  $\mathcal{H}_1, \dots, \mathcal{H}_{18}$  abgekürzt wurden. Die tatsächliche Bedeutung dieser Listen ist am Ende des Beweises in einer Art Legende angegeben.

A+.	A+	$\vdash \forall a. \forall b. \forall c. ((a + b) + c) = (a + (b + c))$	(Hyp)
0+.	0+	$\vdash \forall x. (0 + x) = x$	(Hyp)
C+.	C+	$\vdash \forall a. \forall b. (a + b) = (b + a)$	(Hyp)
CF1.	CF1	$\vdash \forall f. \forall g. \forall u. \forall v. (cf(f, u, v) \boxplus cf(g, u, v)) = cf((f \oplus g), u, v)$	(Hyp)
$\partial$ -const-1.	$\partial$ -const-1	$\vdash \forall a. \partial_p(\lambda x. a, 1) = \lambda x. 0$	(Hyp)
DL.	DL	$\vdash \forall a. \forall b. \forall c. (a * (b + c)) = ((a * b) + (a * c))$	(Hyp)
Intv.	Intv	$\vdash \forall a. \forall b. \forall x. [x \in [a, b] \Leftrightarrow [(a \leq x) \wedge (x \leq b)]]$	(Hyp)
$\leq$ -AX.	$\leq$ -AX	$\vdash \forall x. \forall y. [(x \leq y) \Leftrightarrow \exists z. [(x + z) = y \wedge (0 \leq z)]]$	(Hyp)
Min.	Min	$\vdash \forall f. \forall x. [(\partial_p(f, 1)x) = 0 \wedge (0 < (\partial_p(\partial_p(f, 1), 1)x))] \Rightarrow \text{Min}(f, x)$	(Hyp)
M+.	M+	$\vdash \forall x. \forall n. \forall a. \forall b. (ax^n + bx^n) = (a + b)x^n$	(Hyp)
$\partial$ -mon-1.	$\partial$ -mon-1	$\vdash \forall a. \forall b. \partial_p(\lambda x. ax^b, 1) = \lambda x. (a * b)x^{(b-1)}$	(Hyp)
M*.	M*	$\vdash \forall x. \forall m. \forall n. \forall a. \forall b. (ax^m * bx^n) = (a * b)x^{(m+n)}$	(Hyp)
O.	O	$\vdash \forall f. \forall I. [\text{Opt}(cf(f, DM, prod), I) \Leftrightarrow \exists x. \text{TotMin}(x, f, I)]$	(Hyp)
P.	P	$\vdash \forall f. \forall g. \forall u. \forall v. \forall w. [\text{price}(f, u, v) \Rightarrow cf(g, v, w) = cf((f \otimes g), u, w)]$	(Hyp)
P+.	P+	$\vdash \forall p. \forall q. (p \oplus q) = \lambda z. (p(z) + q(z))$	(Hyp)
$\partial$ -pol.	$\partial$ -pol	$\vdash \forall p. \forall q. \forall c. \partial_p((p \oplus q), c) = (\partial_p(p, c) \oplus \partial_p(q, c))$	(Hyp)
P-kWh.	P-kWh	$\vdash \text{price}(\lambda d. 2, DM, kWh)$	(Hyp)
P-l.	P-l	$\vdash \text{price}(\lambda d. 0.5, DM, l)$	(Hyp)
P*.	P*	$\vdash \forall p. \forall q. (p \otimes q) = \lambda z. (p(z) * q(z))$	(Hyp)
TotMin.	TotMin	$\vdash \forall f. \forall a. \forall b. \forall x. [\text{TotMin}(x, f, [a, b]) \Leftrightarrow [x \in [a, b] \wedge [\text{Min}(f, x) \wedge [(f(x) \leq f(a)) \wedge (f(x) \leq f(b))]]]]]$	(Hyp)
THM.	$\mathcal{H}_1$	$\vdash \text{Opt}((cf(\lambda d. (4d^2 + (\Leftrightarrow 18d + 7))), l, prod) \boxplus cf(\lambda d. (0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), kWh, prod)), [1, 7])$	(=Subst L6,L7)
L1.	P	$\vdash \forall g. \forall u. \forall v. \forall w. [\text{price}(\lambda d. 0.5, u, v) \Rightarrow cf(g, v, w) = cf((\lambda d. 0.5 \otimes g), u, w)]$	( $\forall E$ P)
L2.	P	$\vdash \forall u. \forall v. \forall w. [\text{price}(\lambda d. 0.5, u, v) \Rightarrow cf(\lambda d. (4d^2 + (\Leftrightarrow 18d + 7)), v, w) = cf((\lambda d. 0.5 \otimes \lambda d. (4d^2 + (\Leftrightarrow 18d + 7))), u, w)]$	( $\forall E$ L1)
L3.	P	$\vdash \forall v. \forall w. [\text{price}(\lambda d. 0.5, DM, v) \Rightarrow cf(\lambda d. (4d^2 + (\Leftrightarrow 18d + 7)), v, w) = cf((\lambda d. 0.5 \otimes \lambda d. (4d^2 + (\Leftrightarrow 18d + 7))), DM, w)]$	( $\forall E$ L2)
L4.	P	$\vdash \forall w. [\text{price}(\lambda d. 0.5, DM, l) \Rightarrow cf(\lambda d. (4d^2 + (\Leftrightarrow 18d + 7)), l, w) = cf((\lambda d. 0.5 \otimes \lambda d. (4d^2 + (\Leftrightarrow 18d + 7))), DM, w)]$	( $\forall E$ L3)

L5.	P	$\vdash [price(\lambda d_{\bullet} 0.5, DM, l) \Rightarrow$ $cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7)), l, prod) =$ $cf((\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod)]$	( $\forall E$ L4)
L6.	P, P-1	$\vdash cf(\lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7)), l, prod) =$ $cf((\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod)$	( $\Rightarrow E$ P-1, L5)
L7.	$\mathcal{H}_2$	$\vdash Opt((cf((\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L10, L11)
L8.	P*	$\vdash \forall q_{\bullet}(\lambda d_{\bullet} 0.5 \otimes q) = \lambda z_{\bullet}(\lambda d_{\bullet}(0.5)z * q(z))$	( $\forall E$ P*)
L9.	P*	$\vdash (\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))) =$ $\lambda z_{\bullet}(\lambda d_{\bullet}(0.5)z * \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))z)$	( $\forall E$ L8)
L10.	P*	$\vdash (\lambda d_{\bullet} 0.5 \otimes \lambda d_{\bullet}(4d^2 + (\Leftrightarrow 18d + 7))) =$ $\lambda z_{\bullet}(0.5 * (4z^2 + (\Leftrightarrow 18z + 7)))$	( $\lambda E$ L9)
L11.	$\mathcal{H}_2$	$\vdash Opt((cf(\lambda z_{\bullet}(0.5 * (4z^2 + (\Leftrightarrow 18z + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L14, L15)
L12.	DL	$\vdash \forall b_{\bullet} \forall c_{\bullet}(0.5 * (b + c)) = ((0.5 * b) + (0.5 * c))$	( $\forall E$ DL)
L13.	DL	$\vdash \forall c_{\bullet}(0.5 * (4z^2 + c)) = ((0.5 * 4z^2) + (0.5 * c))$	( $\forall E$ L12)
L14.	DL	$\vdash (0.5 * (4z^2 + (\Leftrightarrow 18z + 7))) = ((0.5 * 4z^2) + (0.5 * (\Leftrightarrow 18z + 7)))$	( $\forall E$ L13)
L15.	$\mathcal{H}_2$	$\vdash Opt((cf(\lambda z_{\bullet}((0.5 * 4z^2) + (0.5 * (\Leftrightarrow 18z + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L21, L22)
L16.	M*	$\vdash \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^m * bz^n) = (a * b)z^{(m+n)}$	( $\forall E$ M*)
L17.	M*	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^0 * bz^n) = (a * b)z^{(0+n)}$	( $\forall E$ L16)
L18.	M*	$\vdash \forall a_{\bullet} \forall b_{\bullet}(az^0 * bz^2) = (a * b)z^{(0+2)}$	( $\forall E$ L17)
L19.	M*	$\vdash \forall b_{\bullet}(0.5z^0 * bz^2) = (0.5 * b)z^{(0+2)}$	( $\forall E$ L18)
L20.	M*	$\vdash (0.5z^0 * 4z^2) = (0.5 * 4)z^{(0+2)}$	( $\forall E$ L19)
L21.	M*	$\vdash (0.5 * 4z^2) = 2z^2$	(Simplify-Num L20)
L22.	$\mathcal{H}_2$	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (0.5 * (\Leftrightarrow 18z + 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L25, L26)
L23.	DL	$\vdash \forall b_{\bullet} \forall c_{\bullet}(0.5 * (b + c)) = ((0.5 * b) + (0.5 * c))$	( $\forall E$ DL)
L24.	DL	$\vdash \forall c_{\bullet}(0.5 * (\Leftrightarrow 18z + c)) = ((0.5 * \Leftrightarrow 18z) + (0.5 * c))$	( $\forall E$ L23)
L25.	DL	$\vdash (0.5 * (\Leftrightarrow 18z + 7)) = ((0.5 * \Leftrightarrow 18z) + (0.5 * 7))$	( $\forall E$ L24)
L26.	$\mathcal{H}_2$	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + ((0.5 * \Leftrightarrow 18z) + (0.5 * 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L32, L33)
L27.	M*	$\vdash \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^m * bz^n) = (a * b)z^{(m+n)}$	( $\forall E$ M*)
L28.	M*	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^0 * bz^n) = (a * b)z^{(0+n)}$	( $\forall E$ L27)
L29.	M*	$\vdash \forall a_{\bullet} \forall b_{\bullet}(az^0 * bz) = (a * b)z^{(0+1)}$	( $\forall E$ L28)
L30.	M*	$\vdash \forall b_{\bullet}(0.5z^0 * bz) = (0.5 * b)z^{(0+1)}$	( $\forall E$ L29)
L31.	M*	$\vdash (0.5z^0 * \Leftrightarrow 18z) = (0.5 * \Leftrightarrow 18)z^{(0+1)}$	( $\forall E$ L30)
L32.	M*	$\vdash (0.5 * \Leftrightarrow 18z) = \Leftrightarrow 9z$	(Simplify-Num L31)
L33.	$\mathcal{H}_2$	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + (0.5 * 7))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L39, L40)
L34.	M*	$\vdash \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^m * bz^n) = (a * b)z^{(m+n)}$	( $\forall E$ M*)
L35.	M*	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^0 * bz^n) = (a * b)z^{(0+n)}$	( $\forall E$ L34)
L36.	M*	$\vdash \forall a_{\bullet} \forall b_{\bullet}(az^0 * bz^0) = (a * b)z^{(0+0)}$	( $\forall E$ L35)
L37.	M*	$\vdash \forall b_{\bullet}(0.5z^0 * bz^0) = (0.5 * b)z^{(0+0)}$	( $\forall E$ L36)
L38.	M*	$\vdash (0.5z^0 * 7z^0) = (0.5 * 7)z^{(0+0)}$	( $\forall E$ L37)
L39.	M*	$\vdash (0.5 * 7) = 3.5$	(Simplify-Num L38)

L40.	$\mathcal{H}_2$	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), kWh, prod)), [1, 7])$	(=Subst L46,L47)
L41.	P	$\vdash \forall g_{\bullet} \forall u_{\bullet} \forall v_{\bullet} \forall w_{\bullet} [price(\lambda d_{\bullet} 2, u, v) \Rightarrow$ $cf(g, v, w) = cf((\lambda d_{\bullet} 2 \otimes g), u, w)]$	( $\forall E$ P)
L42.	P	$\vdash \forall u_{\bullet} \forall v_{\bullet} \forall w_{\bullet} [price(\lambda d_{\bullet} 2, u, v) \Rightarrow$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), v, w) =$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), u, w)]$	( $\forall E$ L41)
L43.	P	$\vdash \forall v_{\bullet} \forall w_{\bullet} [price(\lambda d_{\bullet} 2, DM, v) \Rightarrow$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), v, w) =$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, w)]$	( $\forall E$ L42)
L44.	P	$\vdash \forall w_{\bullet} [price(\lambda d_{\bullet} 2, DM, kWh) \Rightarrow$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), kWh, w) =$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, w)]$	( $\forall E$ L43)
L45.	P	$\vdash [price(\lambda d_{\bullet} 2, DM, kWh) \Rightarrow$ $cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), kWh, prod) =$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, prod)]$	( $\forall E$ L44)
L46.	P, P-kWh	$\vdash cf(\lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)), kWh, prod) =$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, prod)$	( $\Rightarrow E$ P-kWh,L45)
L47.	$\mathcal{H}_3$	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf((\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))), DM, prod)), [1, 7])$	(=Subst L50,L51)
L48.	P*	$\vdash \forall q_{\bullet} (\lambda d_{\bullet} 2 \otimes q) = \lambda z_{\bullet} (\lambda d_{\bullet} (2) z * q(z))$	( $\forall E$ P*)
L49.	P*	$\vdash (\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))) =$ $\lambda z_{\bullet} (\lambda d_{\bullet} (2) z * \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5)) z)$	( $\forall E$ L48)
L50.	P*	$\vdash (\lambda d_{\bullet} 2 \otimes \lambda d_{\bullet}(0.5d^2 + (\Leftrightarrow 1.5d + 0.5))) =$ $\lambda z_{\bullet} (2 * (0.5z^2 + (\Leftrightarrow 1.5z + 0.5)))$	( $\lambda E$ L49)
L51.	$\mathcal{H}_4$	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(2 * (0.5z^2 + (\Leftrightarrow 1.5z + 0.5))), DM, prod)), [1, 7])$	(=Subst L54,L55)
L52.	DL	$\vdash \forall b_{\bullet} \forall c_{\bullet} (2 * (b + c)) = ((2 * b) + (2 * c))$	( $\forall E$ DL)
L53.	DL	$\vdash \forall c_{\bullet} (2 * (0.5z^2 + c)) = ((2 * 0.5z^2) + (2 * c))$	( $\forall E$ L52)
L54.	DL	$\vdash (2 * (0.5z^2 + (\Leftrightarrow 1.5z + 0.5))) = ((2 * 0.5z^2) + (2 * (\Leftrightarrow 1.5z +$ $0.5)))$	( $\forall E$ L53)
L55.	$\mathcal{H}_4$	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}((2 * 0.5z^2) + (2 * (\Leftrightarrow 1.5z + 0.5))), DM, prod)), [1, 7])$	(=Subst L61,L62)
L56.	M*	$\vdash \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet} (az^m * bz^n) = (a * b)z^{(m+n)}$	( $\forall E$ M*)
L57.	M*	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet} (az^0 * bz^n) = (a * b)z^{(0+n)}$	( $\forall E$ L56)
L58.	M*	$\vdash \forall a_{\bullet} \forall b_{\bullet} (az^0 * bz^2) = (a * b)z^{(0+2)}$	( $\forall E$ L57)
L59.	M*	$\vdash \forall b_{\bullet} (2z^0 * bz^2) = (2 * b)z^{(0+2)}$	( $\forall E$ L58)
L60.	M*	$\vdash (2z^0 * 0.5z^2) = (2 * 0.5)z^{(0+2)}$	( $\forall E$ L59)
L61.	M*	$\vdash (2 * 0.5z^2) = z^2$	(Simplify-Num L60)
L62.	$\mathcal{H}_4$	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (2 * (\Leftrightarrow 1.5z + 0.5))), DM, prod)), [1, 7])$	(=Subst L65,L66)
L63.	DL	$\vdash \forall b_{\bullet} \forall c_{\bullet} (2 * (b + c)) = ((2 * b) + (2 * c))$	( $\forall E$ DL)
L64.	DL	$\vdash \forall c_{\bullet} (2 * (\Leftrightarrow 1.5z + c)) = ((2 * \Leftrightarrow 1.5z) + (2 * c))$	( $\forall E$ L63)
L65.	DL	$\vdash (2 * (\Leftrightarrow 1.5z + 0.5)) = ((2 * \Leftrightarrow 1.5z) + (2 * 0.5))$	( $\forall E$ L64)
L66.	$\mathcal{H}_5$	$\vdash \text{Opt}((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + ((2 * \Leftrightarrow 1.5z) + (2 * 0.5))), DM, prod)), [1, 7])$	(=Subst L72,L73)
L67.	M*	$\vdash \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet} (az^m * bz^n) = (a * b)z^{(m+n)}$	( $\forall E$ M*)

L68.	M*	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet} (az^0 * bz^n) = (a * b)z^{(0+n)}$	( $\forall E$ L67)
L69.	M*	$\vdash \forall a_{\bullet} \forall b_{\bullet} (az^0 * bz) = (a * b)z^{(0+1)}$	( $\forall E$ L68)
L70.	M*	$\vdash \forall b_{\bullet} (2z^0 * bz) = (2 * b)z^{(0+1)}$	( $\forall E$ L69)
L71.	M*	$\vdash (2z^0 * \Leftrightarrow 1.5z) = (2 * \Leftrightarrow 1.5)z^{(0+1)}$	( $\forall E$ L70)
L72.	M*	$\vdash (2 * \Leftrightarrow 1.5z) = \Leftrightarrow 3z$	(Simplify-Num L71)
L73.	$\mathcal{H}_5$	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + (2 * 0.5))), DM, prod)), [1, 7])$	(=Subst L79,L80)
L74.	M*	$\vdash \forall m_{\bullet} \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet} (az^m * bz^n) = (a * b)z^{(m+n)}$	( $\forall E$ M*)
L75.	M*	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet} (az^0 * bz^n) = (a * b)z^{(0+n)}$	( $\forall E$ L74)
L76.	M*	$\vdash \forall a_{\bullet} \forall b_{\bullet} (az^0 * bz^0) = (a * b)z^{(0+0)}$	( $\forall E$ L75)
L77.	M*	$\vdash \forall b_{\bullet} (2z^0 * bz^0) = (2 * b)z^{(0+0)}$	( $\forall E$ L76)
L78.	M*	$\vdash (2z^0 * 0.5z^0) = (2 * 0.5)z^{(0+0)}$	( $\forall E$ L77)
L79.	M*	$\vdash (2 * 0.5) = 1$	(Simplify-Num L78)
L80.	$\mathcal{H}_6$	$\vdash Opt((cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod)), [1, 7])$	(=Subst L84,L85)
L81.	CF1	$\vdash \forall g_{\bullet} \forall u_{\bullet} \forall v_{\bullet} (cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), u, v) \boxplus cf(g, u, v) =$ $cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus g), u, v)$	( $\forall E$ CF1)
L82.	CF1	$\vdash \forall u_{\bullet} \forall v_{\bullet} (cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), u, v) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), u, v) =$ $cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), u, v)$	( $\forall E$ L81)
L83.	CF1	$\vdash \forall v_{\bullet} (cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, v) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, v) =$ $cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, v)$	( $\forall E$ L82)
L84.	CF1	$\vdash (cf(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))), DM, prod) \boxplus$ $cf(\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod) =$ $cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod)$	( $\forall E$ L83)
L85.	$\mathcal{H}_7$	$\vdash Opt(cf((\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus$ $\lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))), DM, prod), [1, 7])$	(=Subst L88,L89)
L86.	P+	$\vdash \forall q_{\bullet} (\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus q) =$ $\lambda z_{\bullet}(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))z + q(z))$	( $\forall E$ P+)
L87.	P+	$\vdash (\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))) =$ $\lambda z_{\bullet}(\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5))z + \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))z)$	( $\forall E$ L86)
L88.	P+	$\vdash (\lambda z_{\bullet}(2z^2 + (\Leftrightarrow 9z + 3.5)) \oplus \lambda z_{\bullet}(z^2 + (\Leftrightarrow 3z + 1))) =$ $\lambda z_{\bullet}((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + (\Leftrightarrow 3z + 1)))$	( $\lambda E$ L87)
L89.	$\mathcal{H}_7$	$\vdash Opt(cf(\lambda z_{\bullet}((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + (\Leftrightarrow 3z + 1))),$ $DM, prod), [1, 7])$	(=Subst L92,L93)
L90.	A+	$\vdash \forall b_{\bullet} \forall c_{\bullet} (((2z^2 + (\Leftrightarrow 9z + 3.5)) + b) + c) =$ $((2z^2 + (\Leftrightarrow 9z + 3.5)) + (b + c))$	( $\forall E$ A+)
L91.	A+	$\vdash \forall c_{\bullet} (((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) + c) =$ $((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + c))$	( $\forall E$ L90)
L92.	A+	$\vdash (((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) + (\Leftrightarrow 3z + 1)) =$ $((2z^2 + (\Leftrightarrow 9z + 3.5)) + (z^2 + (\Leftrightarrow 3z + 1)))$	( $\forall E$ L91)
L93.	$\mathcal{H}_7$	$\vdash Opt(cf(\lambda z_{\bullet}(((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) + (\Leftrightarrow 3z + 1))),$ $DM, prod), [1, 7])$	(=Subst L95,L96)
L94.	C+	$\vdash \forall b_{\bullet} ((2z^2 + (\Leftrightarrow 9z + 3.5)) + b) = (b + (2z^2 + (\Leftrightarrow 9z + 3.5)))$	( $\forall E$ C+)
L95.	C+	$\vdash ((2z^2 + (\Leftrightarrow 9z + 3.5)) + z^2) = (z^2 + (2z^2 + (\Leftrightarrow 9z + 3.5)))$	( $\forall E$ L94)

L96.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}((z^2 + (2z^2 + (\Leftrightarrow 9z + 3.5))) + (\Leftrightarrow 3z + 1)),$ $DM, prod), [1, 7])$	(=Subst L99,L100)
L97.	A+	$\vdash \forall b_{\bullet} \forall c_{\bullet}((z^2 + b) + c) = (z^2 + (b + c))$	( $\forall E$ A+)
L98.	A+	$\vdash \forall c_{\bullet}((z^2 + 2z^2) + c) = (z^2 + (2z^2 + c))$	( $\forall E$ L97)
L99.	A+	$\vdash ((z^2 + 2z^2) + (\Leftrightarrow 9z + 3.5)) = (z^2 + (2z^2 + (\Leftrightarrow 9z + 3.5)))$	( $\forall E$ L98)
L100.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(((z^2 + 2z^2) + (\Leftrightarrow 9z + 3.5)) + (\Leftrightarrow 3z + 1)),$ $DM, prod), [1, 7])$	(=Subst L105,L106)
L101.	M+	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^n + bz^n) = (a + b)z^n$	( $\forall E$ M+)
L102.	M+	$\vdash \forall a_{\bullet} \forall b_{\bullet}(az^2 + bz^2) = (a + b)z^2$	( $\forall E$ L101)
L103.	M+	$\vdash \forall b_{\bullet}(z^2 + bz^2) = (1 + b)z^2$	( $\forall E$ L102)
L104.	M+	$\vdash (z^2 + 2z^2) = (1 + 2)z^2$	( $\forall E$ L103)
L105.	M+	$\vdash (z^2 + 2z^2) = 3z^2$	(Simplify-Num L104)
L106.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}((3z^2 + (\Leftrightarrow 9z + 3.5)) + (\Leftrightarrow 3z + 1)), DM, prod),$ $[1, 7])$	(=Subst L109,L110)
L107.	A+	$\vdash \forall b_{\bullet} \forall c_{\bullet}((3z^2 + b) + c) = (3z^2 + (b + c))$	( $\forall E$ A+)
L108.	A+	$\vdash \forall c_{\bullet}((3z^2 + (\Leftrightarrow 9z + 3.5)) + c) = (3z^2 + ((\Leftrightarrow 9z + 3.5) + c))$	( $\forall E$ L107)
L109.	A+	$\vdash ((3z^2 + (\Leftrightarrow 9z + 3.5)) + (\Leftrightarrow 3z + 1)) =$ $(3z^2 + ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + 1)))$	( $\forall E$ L108)
L110.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + 1))), DM, prod),$ $[1, 7])$	(=Subst L113,L114)
L111.	A+	$\vdash \forall b_{\bullet} \forall c_{\bullet}(((\Leftrightarrow 9z + 3.5) + b) + c) = ((\Leftrightarrow 9z + 3.5) + (b + c))$	( $\forall E$ A+)
L112.	A+	$\vdash \forall c_{\bullet}(((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) + c) = ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + c))$	( $\forall E$ L111)
L113.	A+	$\vdash (((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) + 1) = ((\Leftrightarrow 9z + 3.5) + (\Leftrightarrow 3z + 1))$	( $\forall E$ L112)
L114.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) + 1)), DM, prod),$ $[1, 7])$	(=Subst L116,L117)
L115.	C+	$\vdash \forall b_{\bullet}((\Leftrightarrow 9z + 3.5) + b) = (b + (\Leftrightarrow 9z + 3.5))$	( $\forall E$ C+)
L116.	C+	$\vdash ((\Leftrightarrow 9z + 3.5) + \Leftrightarrow 3z) = (\Leftrightarrow 3z + (\Leftrightarrow 9z + 3.5))$	( $\forall E$ L115)
L117.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 3z + (\Leftrightarrow 9z + 3.5)) + 1)), DM, prod),$ $[1, 7])$	(=Subst L120,L121)
L118.	A+	$\vdash \forall b_{\bullet} \forall c_{\bullet}((\Leftrightarrow 3z + b) + c) = (\Leftrightarrow 3z + (b + c))$	( $\forall E$ A+)
L119.	A+	$\vdash \forall c_{\bullet}((\Leftrightarrow 3z + \Leftrightarrow 9z) + c) = (\Leftrightarrow 3z + (\Leftrightarrow 9z + c))$	( $\forall E$ L118)
L120.	A+	$\vdash ((\Leftrightarrow 3z + \Leftrightarrow 9z) + 3.5) = (\Leftrightarrow 3z + (\Leftrightarrow 9z + 3.5))$	( $\forall E$ L119)
L121.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (((\Leftrightarrow 3z + \Leftrightarrow 9z) + 3.5) + 1)), DM, prod),$ $[1, 7])$	(=Subst L126,L127)
L122.	M+	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^n + bz^n) = (a + b)z^n$	( $\forall E$ M+)
L123.	M+	$\vdash \forall a_{\bullet} \forall b_{\bullet}(az + bz) = (a + b)z$	( $\forall E$ L122)
L124.	M+	$\vdash \forall b_{\bullet}(\Leftrightarrow 3z + bz) = (\Leftrightarrow 3 + b)z$	( $\forall E$ L123)
L125.	M+	$\vdash (\Leftrightarrow 3z + \Leftrightarrow 9z) = (\Leftrightarrow 3 + \Leftrightarrow 9)z$	( $\forall E$ L124)
L126.	M+	$\vdash (\Leftrightarrow 3z + \Leftrightarrow 9z) = \Leftrightarrow 12z$	(Simplify-Num L125)
L127.	$\mathcal{H}_7$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 12z + 3.5) + 1)), DM, prod), [1, 7])$	(=Subst L130,L131)
L128.	A+	$\vdash \forall b_{\bullet} \forall c_{\bullet}((\Leftrightarrow 12z + b) + c) = (\Leftrightarrow 12z + (b + c))$	( $\forall E$ A+)
L129.	A+	$\vdash \forall c_{\bullet}((\Leftrightarrow 12z + 3.5) + c) = (\Leftrightarrow 12z + (3.5 + c))$	( $\forall E$ L128)
L130.	A+	$\vdash ((\Leftrightarrow 12z + 3.5) + 1) = (\Leftrightarrow 12z + (3.5 + 1))$	( $\forall E$ L129)
L131.	$\mathcal{H}_8$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + ((\Leftrightarrow 12z + (3.5 + 1))))), DM, prod), [1, 7])$	(=Subst L136,L137)
L132.	M+	$\vdash \forall n_{\bullet} \forall a_{\bullet} \forall b_{\bullet}(az^n + bz^n) = (a + b)z^n$	( $\forall E$ M+)
L133.	M+	$\vdash \forall a_{\bullet} \forall b_{\bullet}(a + b) = (a + b)z^0$	( $\forall E$ L132)
L134.	M+	$\vdash \forall b_{\bullet}(3.5 + b) = (3.5 + b)z^0$	( $\forall E$ L133)

L135.	M+	$\vdash (3.5 + 1) = (3.5 + 1)z^0$	( $\forall E$ L134)
L136.	M+	$\vdash (3.5 + 1) = 4.5$	(Simplify-Num L135)
L137.	$\mathcal{H}_9$	$\vdash \text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))), DM, prod), [1, 7]$	( $\Rightarrow E$ L143, L142)
L138.	o	$\vdash \forall I_{\bullet}[\text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))), DM, prod), I] \Leftrightarrow$ $\exists x_{\bullet} \text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), I)]$	( $\forall E$ O)
L139.	o	$\vdash [\text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))), DM, prod), [1, 7]] \Leftrightarrow$ $\exists x_{\bullet} \text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])]$	( $\forall E$ L138)
L140.	o	$\vdash [[\text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))), DM, prod), [1, 7]] \Rightarrow$ $\exists x_{\bullet} \text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])] \wedge$ $[\exists x_{\bullet} \text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7]] \Rightarrow$ $\text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))), DM, prod), [1, 7])]$	( $\Leftrightarrow E$ L139)
L142.	o	$\vdash [\exists x_{\bullet} \text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7]] \Rightarrow$ $\text{Opt}(cf(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))), DM, prod), [1, 7])]$	( $\wedge E$ L140)
L143.	$\mathcal{H}_{10}$	$\vdash \exists x_{\bullet} \text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])$	( $\exists I$ L144)
L144.	$\mathcal{H}_{10}$	$\vdash \text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])$	( $\Rightarrow E$ L152, L151)
L145.	TotMin	$\vdash \forall a_{\bullet} \forall b_{\bullet} \forall x_{\bullet} [\text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [a, b]) \Leftrightarrow$ $[x \in [a, b] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), x) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))x \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))a] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))x \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))b]]]]]$	( $\forall E$ TotMin)
L146.	TotMin	$\vdash \forall b_{\bullet} \forall x_{\bullet} [\text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, b]) \Leftrightarrow$ $[x \in [1, b] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), x) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))x \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))x \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))b]]]]]$	( $\forall E$ L145)
L147.	TotMin	$\vdash \forall x_{\bullet} [\text{TotMin}(x, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7]) \Leftrightarrow$ $[x \in [1, 7] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), x) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))x \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))x \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]]]]$	( $\forall E$ L146)
L148.	TotMin	$\vdash [\text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7]) \Leftrightarrow$ $[2 \in [1, 7] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]]]]$	( $\forall E$ L147)
L149.	TotMin	$\vdash [[\text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7]) \Rightarrow$ $[2 \in [1, 7] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]]]] \wedge$ $[[2 \in [1, 7] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]]]] \Rightarrow$ $\text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])]$	( $\Leftrightarrow E$ L148)
L151.	TotMin	$\vdash [[2 \in [1, 7] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]]]] \Rightarrow$ $\text{TotMin}(2, \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), [1, 7])]$	( $\wedge E$ L149)
L152.	$\mathcal{H}_{11}$	$\vdash [2 \in [1, 7] \wedge [\text{Min}(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1] \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]]]]$	( $\wedge I$ L153, L154)
L153.	$\mathcal{H}_{18}$	$\vdash 2 \in [1, 7]$	( $\Rightarrow E$ L328, L327)

L154.	$\mathcal{H}_{12}$	$\vdash [Min(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2) \wedge$ $[(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1) \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]$	( $\wedge I$ L155,L156)
L155.	$\mathcal{H}_{13}$	$\vdash Min(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2)$	( $\Rightarrow E$ L191,L190)
L156.	$\leq\text{-AX}$	$\vdash [(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1) \wedge$ $(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7]]$	( $\wedge I$ L157,L158)
L157.	$\leq\text{-AX}$	$\vdash (\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))1)$	( $\wedge I$ L159)
L158.	$\leq\text{-AX}$	$\vdash (\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)))2 \leq \lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5))7)$	( $\wedge I$ L174)
L159.	$\leq\text{-AX}$	$\vdash (((3 * 2^2) + ((\Leftrightarrow 12 * 2) + 4.5)) \leq ((3 * 1^2) + ((\Leftrightarrow 12 * 1) + 4.5)))$	(Simplify-Num L160)
L160.	$\leq\text{-AX}$	$\vdash (\Leftrightarrow 7.5 \leq ((3 * 1^2) + ((\Leftrightarrow 12 * 1) + 4.5)))$	(Simplify-Num L161)
L161.	$\leq\text{-AX}$	$\vdash (\Leftrightarrow 7.5 \leq \Leftrightarrow 4.5)$	( $\Rightarrow E$ L167,L166)
L162.	$\leq\text{-AX}$	$\vdash \forall y_{\bullet}[(\Leftrightarrow 7.5 \leq y) \Leftrightarrow \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = y \wedge (0 \leq z)]]$	( $\forall E$ $\leq\text{-AX}$ )
L163.	$\leq\text{-AX}$	$\vdash [(\Leftrightarrow 7.5 \leq \Leftrightarrow 4.5) \Leftrightarrow \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = \Leftrightarrow 4.5 \wedge (0 \leq z)]]$	( $\forall E$ L162)
L164.	$\leq\text{-AX}$	$\vdash [((\Leftrightarrow 7.5 \leq \Leftrightarrow 4.5) \Rightarrow \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = \Leftrightarrow 4.5 \wedge (0 \leq z)]) \wedge$ $[\exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = \Leftrightarrow 4.5 \wedge (0 \leq z)] \Rightarrow$ $(\Leftrightarrow 7.5 \leq \Leftrightarrow 4.5)]]$	( $\Leftrightarrow E$ L163)
L166.	$\leq\text{-AX}$	$\vdash [\exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = \Leftrightarrow 4.5 \wedge (0 \leq z)] \Rightarrow (\Leftrightarrow 7.5 \leq \Leftrightarrow 4.5)]$	( $\wedge E$ L164)
L167.		$\vdash \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = \Leftrightarrow 4.5 \wedge (0 \leq z)]$	( $\exists I$ L168)
L168.		$\vdash [(\Leftrightarrow 7.5 + 3) = \Leftrightarrow 4.5 \wedge (0 \leq 3)]$	( $\wedge I$ L169,L171)
L169.		$\vdash (\Leftrightarrow 7.5 + 3) = \Leftrightarrow 4.5$	(Simplify-Num L173)
L171.	L171	$\vdash (0 \leq 3)$	(Hyp)
L173.		$\vdash \Leftrightarrow 4.5 = \Leftrightarrow 4.5$	(=I)
L174.	$\leq\text{-AX}$	$\vdash (((3 * 2^2) + ((\Leftrightarrow 12 * 2) + 4.5)) \leq ((3 * 7^2) + ((\Leftrightarrow 12 * 7) + 4.5)))$	(Simplify-Num L175)
L175.	$\leq\text{-AX}$	$\vdash (\Leftrightarrow 7.5 \leq ((3 * 7^2) + ((\Leftrightarrow 12 * 7) + 4.5)))$	(Simplify-Num L176)
L176.	$\leq\text{-AX}$	$\vdash (\Leftrightarrow 7.5 \leq 67.5)$	( $\Rightarrow E$ L182,L181)
L177.	$\leq\text{-AX}$	$\vdash \forall y_{\bullet}[(\Leftrightarrow 7.5 \leq y) \Leftrightarrow \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = y \wedge (0 \leq z)]]$	( $\forall E$ $\leq\text{-AX}$ )
L178.	$\leq\text{-AX}$	$\vdash [(\Leftrightarrow 7.5 \leq 67.5) \Leftrightarrow \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = 67.5 \wedge (0 \leq z)]]$	( $\forall E$ L177)
L179.	$\leq\text{-AX}$	$\vdash [((\Leftrightarrow 7.5 \leq 67.5) \Rightarrow \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = 67.5 \wedge (0 \leq z)]) \wedge$ $[\exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = 67.5 \wedge (0 \leq z)] \Rightarrow (\Leftrightarrow 7.5 \leq 67.5)]]$	( $\Leftrightarrow E$ L178)
L181.	$\leq\text{-AX}$	$\vdash [\exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = 67.5 \wedge (0 \leq z)] \Rightarrow (\Leftrightarrow 7.5 \leq 67.5)]$	( $\wedge E$ L179)
L182.		$\vdash \exists z_{\bullet}[(\Leftrightarrow 7.5 + z) = 67.5 \wedge (0 \leq z)]$	( $\exists I$ L183)
L183.		$\vdash [(\Leftrightarrow 7.5 + 75) = 67.5 \wedge (0 \leq 75)]$	( $\wedge I$ L184,L186)
L184.		$\vdash (\Leftrightarrow 7.5 + 75) = 67.5$	(Simplify-Num L188)
L186.	L186	$\vdash (0 \leq 75)$	(Hyp)
L188.		$\vdash 67.5 = 67.5$	(=I)
L189.	Min	$\vdash \forall x_{\bullet}[(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1)x) = 0 \wedge$ $(0 < (\partial_p(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1), 1)x))] \Rightarrow$ $Min(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), x)]$	( $\forall E$ Min)
L190.	Min	$\vdash [((\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1)2) = 0 \wedge$ $(0 < (\partial_p(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1), 1)2))] \Rightarrow$ $Min(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 2)]$	( $\forall E$ L189)
L191.	$\mathcal{H}_{14}$	$\vdash [(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1)2) = 0 \wedge$ $(0 < (\partial_p(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1), 1)2))]$	( $\wedge I$ L192,L193)
L192.	$\mathcal{H}_{14}$	$\vdash (\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1)2) = 0$	(=Subst L196,L198)
L193.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p(\partial_p(\lambda z_{\bullet}(3z^2 + (\Leftrightarrow 12z + 4.5)), 1), 1)2))]$	(=Subst L247,L249)
L194.	P+	$\vdash \forall q_{\bullet}(\lambda z_{\bullet}3z^2 \oplus q) = \lambda z_{\bullet}(\lambda z_{\bullet}(3z^2)z + q(z))$	( $\forall E$ P+)

L195.	P+	$\vdash (\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)) =$ $\lambda z_{\bullet} (\lambda z_{\bullet} (3z^2)z + \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)z)$	( $\forall E$ L194)
L196.	P+	$\vdash (\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)) = \lambda z_{\bullet} (3z^2 + (\Leftrightarrow 12z + 4.5))$	( $\lambda E$ L195)
L198.	$\mathcal{H}_{14}$	$\vdash (\partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), 1)2) = 0$	(=Subst L203,L204)
L199.	$\partial$ -pol	$\vdash \forall q_{\bullet} \forall c_{\bullet} \partial_p((\lambda z_{\bullet} 3z^2 \oplus q), c) = (\partial_p(\lambda z_{\bullet} 3z^2, c) \oplus \partial_p(q, c))$	( $\forall E$ $\partial$ -pol)
L200.	$\partial$ -pol	$\vdash \forall c_{\bullet} \partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), c) =$ $(\partial_p(\lambda z_{\bullet} 3z^2, c) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), c))$	( $\forall E$ L199)
L201.	$\partial$ -pol	$\vdash \partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), 1) =$ $(\partial_p(\lambda z_{\bullet} 3z^2, 1) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1))$	( $\forall E$ L200)
L202.	$\partial$ -pol	$\vdash \forall x_{\bullet} (\partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), 1)x) =$ $((\partial_p(\lambda z_{\bullet} 3z^2, 1) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1))x)$	(Ext-I L201)
L203.	$\partial$ -pol	$\vdash (\partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), 1)2) =$ $((\partial_p(\lambda z_{\bullet} 3z^2, 1) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1))2)$	( $\forall E$ L202)
L204.	$\mathcal{H}_{14}$	$\vdash ((\partial_p(\lambda z_{\bullet} 3z^2, 1) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1))2) = 0$	(=Subst L208,L209)
L205.	$\partial$ -mon-1	$\vdash \forall b_{\bullet} \partial_p(\lambda x_{\bullet} 3x^b, 1) = \lambda x_{\bullet} (3 * b)x^{(b-1)}$	( $\forall E$ $\partial$ -mon-1)
L206.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} 3x^2, 1) = \lambda x_{\bullet} (3 * 2)x^{(2-1)}$	( $\forall E$ L205)
L207.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} 3x^2, 1) = \lambda x_{\bullet} 6x$	(Simplify-Num L206)
L208.	$\partial$ -mon-1	$\vdash \partial_p(\lambda z_{\bullet} 3z^2, 1) = \lambda x_{\bullet} 6x$	(AB L207)
L209.	$\mathcal{H}_{14}$	$\vdash ((\lambda x_{\bullet} 6x \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1))2) = 0$	(=Subst L212,L214)
L210.	P+	$\vdash \forall q_{\bullet} (\lambda z_{\bullet} \Leftrightarrow 12z \oplus q) = \lambda z_{\bullet} (\lambda z_{\bullet} (\Leftrightarrow 12z)z + q(z))$	( $\forall E$ P+)
L211.	P+	$\vdash (\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5) = \lambda z_{\bullet} (\lambda z_{\bullet} (\Leftrightarrow 12z)z + \lambda z_{\bullet} (4.5)z)$	( $\forall E$ L210)
L212.	P+	$\vdash (\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5) = \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)$	( $\lambda E$ L211)
L214.	$\mathcal{H}_{14}$	$\vdash ((\lambda x_{\bullet} 6x \oplus \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5), 1))2) = 0$	(=Subst L217,L218)
L215.	$\partial$ -pol	$\vdash \forall q_{\bullet} \forall c_{\bullet} \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus q), c) = (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, c) \oplus \partial_p(q, c))$	( $\forall E$ $\partial$ -pol)
L216.	$\partial$ -pol	$\vdash \forall c_{\bullet} \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5), c) =$ $(\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, c) \oplus \partial_p(\lambda z_{\bullet} 4.5, c))$	( $\forall E$ L215)
L217.	$\partial$ -pol	$\vdash \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5), 1) =$ $(\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet} 4.5, 1))$	( $\forall E$ L216)
L218.	$\mathcal{H}_{15}$	$\vdash ((\lambda x_{\bullet} 6x \oplus (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet} 4.5, 1)))2) = 0$	(=Subst L222,L223)
L219.	$\partial$ -mon-1	$\vdash \forall b_{\bullet} \partial_p(\lambda x_{\bullet} \Leftrightarrow 12x^b, 1) = \lambda x_{\bullet} (\Leftrightarrow 12 * b)x^{(b-1)}$	( $\forall E$ $\partial$ -mon-1)
L220.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} \Leftrightarrow 12x, 1) = \lambda x_{\bullet} (\Leftrightarrow 12 * 1)x^{(1-1)}$	( $\forall E$ L219)
L221.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} \Leftrightarrow 12x, 1) = \lambda x_{\bullet} \Leftrightarrow 12$	(Simplify-Num L220)
L222.	$\partial$ -mon-1	$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) = \lambda x_{\bullet} \Leftrightarrow 12$	(AB L221)
L223.	$\mathcal{H}_{16}$	$\vdash ((\lambda x_{\bullet} 6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \partial_p(\lambda z_{\bullet} 4.5, 1)))2) = 0$	(=Subst L225,L226)
L224.	$\partial$ -const-1	$\vdash \partial_p(\lambda x_{\bullet} 4.5, 1) = \lambda x_{\bullet} 0$	( $\forall E$ $\partial$ -const-1)
L225.	$\partial$ -const-1	$\vdash \partial_p(\lambda z_{\bullet} 4.5, 1) = \lambda x_{\bullet} 0$	(AB L224)
L226.	$\mathcal{H}_{17}$	$\vdash ((\lambda x_{\bullet} 6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet} 0))2) = 0$	(=Subst L229,L230)
L227.	P+	$\vdash \forall q_{\bullet} (\lambda x_{\bullet} \Leftrightarrow 12 \oplus q) = \lambda z_{\bullet} (\lambda x_{\bullet} (\Leftrightarrow 12)z + q(z))$	( $\forall E$ P+)
L228.	P+	$\vdash (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet} (\lambda x_{\bullet} (\Leftrightarrow 12)z + \lambda x_{\bullet} (0)z)$	( $\forall E$ L227)
L229.	P+	$\vdash (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet} (\Leftrightarrow 12 + 0)$	( $\lambda E$ L228)
L230.	$\mathcal{H}_{17}$	$\vdash ((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} (\Leftrightarrow 12 + 0))2) = 0$	(=Subst L232,L233)
L231.	C+	$\vdash \forall b_{\bullet} (\Leftrightarrow 12 + b) = (b + \Leftrightarrow 12)$	( $\forall E$ C+)
L232.	C+	$\vdash (\Leftrightarrow 12 + 0) = (0 + \Leftrightarrow 12)$	( $\forall E$ L231)
L233.	0+, P+	$\vdash ((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} (0 + \Leftrightarrow 12))2) = 0$	(=Subst L234,L235)
L234.	0+	$\vdash (0 + \Leftrightarrow 12) = \Leftrightarrow 12$	( $\forall E$ 0+)
L235.	P+	$\vdash ((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12)2) = 0$	(=Subst L240,L241)
L236.	P+	$\vdash \forall q_{\bullet} (\lambda x_{\bullet} 6x \oplus q) = \lambda z_{\bullet} (\lambda x_{\bullet} (6x)z + q(z))$	( $\forall E$ P+)

L237.	P+	$\vdash (\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet} (\lambda x_{\bullet} (6x)z + \lambda z_{\bullet} (\Leftrightarrow 12)z)$	( $\forall E$ L236)
L238.	P+	$\vdash (\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet} (6z + \Leftrightarrow 12)$	( $\lambda E$ L237)
L239.	P+	$\vdash \forall x_{\bullet} ((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12)x) = \lambda z_{\bullet} (6z + \Leftrightarrow 12)x$	(Ext-I L238)
L240.	P+	$\vdash ((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12)2) = \lambda z_{\bullet} (6z + \Leftrightarrow 12)2$	( $\forall E$ L239)
L241.		$\vdash \lambda z_{\bullet} (6z + \Leftrightarrow 12)2 = 0$	( $\lambda I$ L242)
L242.		$\vdash ((6 * 2^1) + \Leftrightarrow 12) = 0$	(Simplify-Num L244)
L244.		$\vdash 0 = 0$	(=I)
L245.	P+	$\vdash \forall q_{\bullet} (\lambda z_{\bullet} 3z^2 \oplus q) = \lambda z_{\bullet} (\lambda z_{\bullet} (3z^2)z + q(z))$	( $\forall E$ P+)
L246.	P+	$\vdash (\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)) =$ $\lambda z_{\bullet} (\lambda z_{\bullet} (3z^2)z + \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)z)$	( $\forall E$ L245)
L247.	P+	$\vdash (\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)) = \lambda z_{\bullet} (3z^2 + (\Leftrightarrow 12z + 4.5))$	( $\lambda E$ L246)
L249.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p(\partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), 1), 1)2))$	(=Subst L252,L253)
L250.	$\partial$ -pol	$\vdash \forall q_{\bullet} \forall c_{\bullet} \partial_p((\lambda z_{\bullet} 3z^2 \oplus q), c) = (\partial_p(\lambda z_{\bullet} 3z^2, c) \oplus \partial_p(q, c))$	( $\forall E$ $\partial$ -pol)
L251.	$\partial$ -pol	$\vdash \forall c_{\bullet} \partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), c) =$ $(\partial_p(\lambda z_{\bullet} 3z^2, c) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), c))$	( $\forall E$ L250)
L252.	$\partial$ -pol	$\vdash \partial_p((\lambda z_{\bullet} 3z^2 \oplus \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)), 1) =$ $(\partial_p(\lambda z_{\bullet} 3z^2, 1) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1))$	( $\forall E$ L251)
L253.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\partial_p(\lambda z_{\bullet} 3z^2, 1) \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1)), 1)2))$	(=Subst L257,L258)
L254.	$\partial$ -mon-1	$\vdash \forall b_{\bullet} \partial_p(\lambda x_{\bullet} 3x^b, 1) = \lambda x_{\bullet} (3 * b)x^{(b-1)}$	( $\forall E$ $\partial$ -mon-1)
L255.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} 3x^2, 1) = \lambda x_{\bullet} (3 * 2)x^{(2-1)}$	( $\forall E$ L254)
L256.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} 3x^2, 1) = \lambda x_{\bullet} 6x$	(Simplify-Num L255)
L257.	$\partial$ -mon-1	$\vdash \partial_p(\lambda z_{\bullet} 3z^2, 1) = \lambda x_{\bullet} 6x$	(AB L256)
L258.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus \partial_p(\lambda z_{\bullet} (\Leftrightarrow 12z + 4.5), 1)), 1)2))$	(=Subst L261,L263)
L259.	P+	$\vdash \forall q_{\bullet} (\lambda z_{\bullet} \Leftrightarrow 12z \oplus q) = \lambda z_{\bullet} (\lambda z_{\bullet} (\Leftrightarrow 12z)z + q(z))$	( $\forall E$ P+)
L260.	P+	$\vdash (\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5) = \lambda z_{\bullet} (\lambda z_{\bullet} (\Leftrightarrow 12z)z + \lambda z_{\bullet} (4.5)z)$	( $\forall E$ L259)
L261.	P+	$\vdash (\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5) = \lambda z_{\bullet} (\Leftrightarrow 12z + 4.5)$	( $\lambda E$ L260)
L263.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5), 1)), 1)2))$	(=Subst L266,L267)
L264.	$\partial$ -pol	$\vdash \forall q_{\bullet} \forall c_{\bullet} \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus q), c) = (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, c) \oplus \partial_p(q, c))$	( $\forall E$ $\partial$ -pol)
L265.	$\partial$ -pol	$\vdash \forall c_{\bullet} \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5), c) =$ $(\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, c) \oplus \partial_p(\lambda z_{\bullet} 4.5, c))$	( $\forall E$ L264)
L266.	$\partial$ -pol	$\vdash \partial_p((\lambda z_{\bullet} \Leftrightarrow 12z \oplus \lambda z_{\bullet} 4.5), 1) =$ $(\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet} 4.5, 1))$	( $\forall E$ L265)
L267.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus (\partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) \oplus \partial_p(\lambda z_{\bullet} 4.5, 1))), 1)2))$	(=Subst L271,L272)
L268.	$\partial$ -mon-1	$\vdash \forall b_{\bullet} \partial_p(\lambda x_{\bullet} \Leftrightarrow 12x^b, 1) = \lambda x_{\bullet} (\Leftrightarrow 12 * b)x^{(b-1)}$	( $\forall E$ $\partial$ -mon-1)
L269.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} \Leftrightarrow 12x, 1) = \lambda x_{\bullet} (\Leftrightarrow 12 * 1)x^{(1-1)}$	( $\forall E$ L268)
L270.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} \Leftrightarrow 12x, 1) = \lambda x_{\bullet} \Leftrightarrow 12$	(Simplify-Num L269)
L271.	$\partial$ -mon-1	$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12z, 1) = \lambda x_{\bullet} \Leftrightarrow 12$	(AB L270)
L272.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \partial_p(\lambda z_{\bullet} 4.5, 1))), 1)2))$	(=Subst L274,L275)
L273.	$\partial$ -const-1	$\vdash \partial_p(\lambda x_{\bullet} 4.5, 1) = \lambda x_{\bullet} 0$	( $\forall E$ $\partial$ -const-1)
L274.	$\partial$ -const-1	$\vdash \partial_p(\lambda z_{\bullet} 4.5, 1) = \lambda x_{\bullet} 0$	(AB L273)
L275.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet} 0)), 1)2))$	(=Subst L278,L279)
L276.	P+	$\vdash \forall q_{\bullet} (\lambda x_{\bullet} \Leftrightarrow 12 \oplus q) = \lambda z_{\bullet} (\lambda x_{\bullet} (\Leftrightarrow 12)z + q(z))$	( $\forall E$ P+)
L277.	P+	$\vdash (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet} (\lambda x_{\bullet} (\Leftrightarrow 12)z + \lambda x_{\bullet} (0)z)$	( $\forall E$ L276)
L278.	P+	$\vdash (\lambda x_{\bullet} \Leftrightarrow 12 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet} (\Leftrightarrow 12 + 0)$	( $\lambda E$ L277)
L279.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} (\Leftrightarrow 12 + 0)), 1)2))$	(=Subst L281,L282)
L280.	C+	$\vdash \forall b_{\bullet} (\Leftrightarrow 12 + b) = (b + \Leftrightarrow 12)$	( $\forall E$ C+)
L281.	C+	$\vdash (\Leftrightarrow 12 + 0) = (0 + \Leftrightarrow 12)$	( $\forall E$ L280)

L282.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet}(0 + \Leftrightarrow 12)), 1)2))$	(=Subst L283,L284)
L283.	0+	$\vdash (0 + \Leftrightarrow 12) = \Leftrightarrow 12$	( $\forall E$ 0+)
L284.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)2))$	(=Subst L287,L288)
L285.	P+	$\vdash \forall q_{\bullet}(\lambda x_{\bullet} 6x \oplus q) = \lambda z_{\bullet}(\lambda x_{\bullet}(6x)z + q(z))$	( $\forall E$ P+)
L286.	P+	$\vdash (\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet}(\lambda x_{\bullet}(6x)z + \lambda z_{\bullet}(\Leftrightarrow 12)z)$	( $\forall E$ L285)
L287.	P+	$\vdash (\lambda x_{\bullet} 6x \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet}(6z + \Leftrightarrow 12)$	( $\lambda E$ L286)
L288.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p(\lambda z_{\bullet}(6z + \Leftrightarrow 12), 1)2))$	(=Subst L291,L293)
L289.	P+	$\vdash \forall q_{\bullet}(\lambda z_{\bullet} 6z \oplus q) = \lambda z_{\bullet}(\lambda z_{\bullet}(6z)z + q(z))$	( $\forall E$ P+)
L290.	P+	$\vdash (\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet}(\lambda z_{\bullet}(6z)z + \lambda z_{\bullet}(\Leftrightarrow 12)z)$	( $\forall E$ L289)
L291.	P+	$\vdash (\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12) = \lambda z_{\bullet}(6z + \Leftrightarrow 12)$	( $\lambda E$ L290)
L293.	$\mathcal{H}_{14}$	$\vdash (0 < (\partial_p((\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)2))$	(=Subst L298,L299)
L294.	$\partial$ -pol	$\vdash \forall c_{\bullet} \forall q_{\bullet} \partial_p((\lambda z_{\bullet} 6z \oplus q), c) = (\partial_p(\lambda z_{\bullet} 6z, c) \oplus \partial_p(q, c))$	( $\forall E$ $\partial$ -pol)
L295.	$\partial$ -pol	$\vdash \forall c_{\bullet} \partial_p((\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), c) = (\partial_p(\lambda z_{\bullet} 6z, c) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, c))$	( $\forall E$ L294)
L296.	$\partial$ -pol	$\vdash \partial_p((\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1) = (\partial_p(\lambda z_{\bullet} 6z, 1) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))$	( $\forall E$ L295)
L297.	$\partial$ -pol	$\vdash \forall x_{\bullet}(\partial_p((\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)x) = ((\partial_p(\lambda z_{\bullet} 6z, 1) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))x)$	(Ext-I L296)
L298.	$\partial$ -pol	$\vdash (\partial_p((\lambda z_{\bullet} 6z \oplus \lambda z_{\bullet} \Leftrightarrow 12), 1)2) = ((\partial_p(\lambda z_{\bullet} 6z, 1) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2)$	( $\forall E$ L297)
L299.	$\mathcal{H}_{15}$	$\vdash (0 < ((\partial_p(\lambda z_{\bullet} 6z, 1) \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2))$	(=Subst L303,L304)
L300.	$\partial$ -mon-1	$\vdash \forall b_{\bullet} \partial_p(\lambda x_{\bullet} 6x^b, 1) = \lambda x_{\bullet}(6 * b)x^{(b-1)}$	( $\forall E$ $\partial$ -mon-1)
L301.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} 6x, 1) = \lambda x_{\bullet}(6 * 1)x^{(1-1)}$	( $\forall E$ L300)
L302.	$\partial$ -mon-1	$\vdash \partial_p(\lambda x_{\bullet} 6x, 1) = \lambda x_{\bullet} 6$	(Simplify-Num L301)
L303.	$\partial$ -mon-1	$\vdash \partial_p(\lambda z_{\bullet} 6z, 1) = \lambda x_{\bullet} 6$	(AB L302)
L304.	$\mathcal{H}_{16}$	$\vdash (0 < ((\lambda x_{\bullet} 6 \oplus \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1))2))$	(=Subst L306,L307)
L305.	$\partial$ -const-1	$\vdash \partial_p(\lambda x_{\bullet} \Leftrightarrow 12, 1) = \lambda x_{\bullet} 0$	( $\forall E$ $\partial$ -const-1)
L306.	$\partial$ -const-1	$\vdash \partial_p(\lambda z_{\bullet} \Leftrightarrow 12, 1) = \lambda x_{\bullet} 0$	(AB L305)
L307.	$\mathcal{H}_{17}$	$\vdash (0 < ((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)2))$	(=Subst L312,L313)
L308.	P+	$\vdash \forall q_{\bullet}(\lambda x_{\bullet} 6 \oplus q) = \lambda z_{\bullet}(\lambda x_{\bullet}(6)z + q(z))$	( $\forall E$ P+)
L309.	P+	$\vdash (\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet}(\lambda x_{\bullet}(6)z + \lambda x_{\bullet}(0)z)$	( $\forall E$ L308)
L310.	P+	$\vdash (\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0) = \lambda z_{\bullet}(6 + 0)$	( $\lambda E$ L309)
L311.	P+	$\vdash \forall x_{\bullet}((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)x) = \lambda z_{\bullet}(6 + 0)x$	(Ext-I L310)
L312.	P+	$\vdash ((\lambda x_{\bullet} 6 \oplus \lambda x_{\bullet} 0)2) = \lambda z_{\bullet}(6 + 0)2$	( $\forall E$ L311)
L313.	0+, C+	$\vdash (0 < \lambda z_{\bullet}(6 + 0)2)$	(=Subst L315,L316)
L314.	C+	$\vdash \forall b_{\bullet}(6 + b) = (b + 6)$	( $\forall E$ C+)
L315.	C+	$\vdash (6 + 0) = (0 + 6)$	( $\forall E$ L314)
L316.	0+	$\vdash (0 < \lambda z_{\bullet}(0 + 6)2)$	(=Subst L317,L318)
L317.	0+	$\vdash (0 + 6) = 6$	( $\forall E$ 0+)
L318.		$\vdash (0 < \lambda z_{\bullet}(6)2)$	( $\lambda I$ L319)
L319.		$\vdash (0 < 6)$	(Simplify-Num L321)
L321.	L321	$\vdash (0 < 6)$	(Hyp)
L322.	Intv	$\vdash \forall b_{\bullet} \forall x_{\bullet}[x \in [1, b] \Leftrightarrow [(1 \leq x) \wedge (x \leq b)]]$	( $\forall E$ Intv)
L323.	Intv	$\vdash \forall x_{\bullet}[x \in [1, 7] \Leftrightarrow [(1 \leq x) \wedge (x \leq 7)]]$	( $\forall E$ L322)
L324.	Intv	$\vdash [2 \in [1, 7] \Leftrightarrow [(1 \leq 2) \wedge (2 \leq 7)]]$	( $\forall E$ L323)
L325.	Intv	$\vdash [[2 \in [1, 7] \Rightarrow [(1 \leq 2) \wedge (2 \leq 7)]] \wedge [(1 \leq 2) \wedge (2 \leq 7)] \Rightarrow 2 \in [1, 7]]$	( $\Leftrightarrow E$ L324)
L327.	Intv	$\vdash [[(1 \leq 2) \wedge (2 \leq 7)] \Rightarrow 2 \in [1, 7]]$	( $\wedge E$ L325)

L328.	$\leq$ -AX	$\vdash [(1 \leq 2) \wedge (2 \leq 7)]$	( $\wedge I$ L329,L330)
L329.	$\leq$ -AX	$\vdash (1 \leq 2)$	( $\Rightarrow E$ L336,L335)
L330.	$\leq$ -AX	$\vdash (2 \leq 7)$	( $\Rightarrow E$ L348,L347)
L331.	$\leq$ -AX	$\vdash \forall y_{\bullet} [(1 \leq y) \Leftrightarrow \exists z_{\bullet} [(1+z) = y \wedge (0 \leq z)]]$	( $\forall E \leq$ -AX)
L332.	$\leq$ -AX	$\vdash [(1 \leq 2) \Leftrightarrow \exists z_{\bullet} [(1+z) = 2 \wedge (0 \leq z)]]$	( $\forall E$ L331)
L333.	$\leq$ -AX	$\vdash [[(1 \leq 2) \Rightarrow \exists z_{\bullet} [(1+z) = 2 \wedge (0 \leq z)]] \wedge$ $\quad \exists z_{\bullet} [(1+z) = 2 \wedge (0 \leq z)] \Rightarrow (1 \leq 2)]$	( $\Rightarrow E$ L332)
L335.	$\leq$ -AX	$\vdash [\exists z_{\bullet} [(1+z) = 2 \wedge (0 \leq z)] \Rightarrow (1 \leq 2)]$	( $\wedge E$ L333)
L336.		$\vdash \exists z_{\bullet} [(1+z) = 2 \wedge (0 \leq z)]$	( $\exists I$ L337)
L337.		$\vdash [(1+1) = 2 \wedge (0 \leq 1)]$	( $\wedge I$ L338,L340)
L338.		$\vdash (1+1) = 2$	(Simplify-Num L342)
L340.	L340	$\vdash (0 \leq 1)$	(Hyp)
L342.		$\vdash 2 = 2$	(=I)
L343.	$\leq$ -AX	$\vdash \forall y_{\bullet} [(2 \leq y) \Leftrightarrow \exists z_{\bullet} [(2+z) = y \wedge (0 \leq z)]]$	( $\forall E \leq$ -AX)
L344.	$\leq$ -AX	$\vdash [(2 \leq 7) \Leftrightarrow \exists z_{\bullet} [(2+z) = 7 \wedge (0 \leq z)]]$	( $\forall E$ L343)
L345.	$\leq$ -AX	$\vdash [[(2 \leq 7) \Rightarrow \exists z_{\bullet} [(2+z) = 7 \wedge (0 \leq z)]] \wedge$ $\quad \exists z_{\bullet} [(2+z) = 7 \wedge (0 \leq z)] \Rightarrow (2 \leq 7)]$	( $\Rightarrow E$ L344)
L347.	$\leq$ -AX	$\vdash [\exists z_{\bullet} [(2+z) = 7 \wedge (0 \leq z)] \Rightarrow (2 \leq 7)]$	( $\wedge E$ L345)
L348.		$\vdash \exists z_{\bullet} [(2+z) = 7 \wedge (0 \leq z)]$	( $\exists I$ L349)
L349.		$\vdash [(2+5) = 7 \wedge (0 \leq 5)]$	( $\wedge I$ L350,L352)
L350.		$\vdash (2+5) = 7$	(Simplify-Num L354)
L352.	L352	$\vdash (0 \leq 5)$	(Hyp)
L354.		$\vdash 7 = 7$	(=I)

$\mathcal{H}_1 = A+, 0+, C+, CF1, \partial$ -const-1, DL, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, M\*, O, P, P+,  $\partial$ -pol, P-kWh, P-l, P\*, TotMin

$\mathcal{H}_2 = A+, 0+, C+, CF1, \partial$ -const-1, DL, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, M\*, O, P, P+,  $\partial$ -pol, P-kWh, P\*, TotMin

$\mathcal{H}_3 = A+, 0+, C+, CF1, \partial$ -const-1, DL, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, M\*, O, P+,  $\partial$ -pol, P\*, TotMin

$\mathcal{H}_4 = A+, 0+, C+, CF1, \partial$ -const-1, DL, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, M\*, O, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_5 = A+, 0+, C+, CF1, \partial$ -const-1, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, M\*, O, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_6 = A+, 0+, C+, CF1, \partial$ -const-1, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, O, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_7 = A+, 0+, C+, \partial$ -const-1, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, O, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_8 = 0+, C+, \partial$ -const-1, Intv,  $\leq$ -AX, Min, M+,  $\partial$ -mon-1, O, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_9 = 0+, C+, \partial$ -const-1, Intv,  $\leq$ -AX, Min,  $\partial$ -mon-1, O, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_{10} = 0+, C+, \partial$ -const-1, Intv,  $\leq$ -AX, Min,  $\partial$ -mon-1, P+,  $\partial$ -pol, TotMin

$\mathcal{H}_{11} = 0+, C+, \partial$ -const-1, Intv,  $\leq$ -AX, Min,  $\partial$ -mon-1, P+,  $\partial$ -pol

$\mathcal{H}_{12} = 0+, C+, \partial$ -const-1,  $\leq$ -AX, Min,  $\partial$ -mon-1, P+,  $\partial$ -pol

$\mathcal{H}_{13} = 0+, C+, \partial$ -const-1, Min,  $\partial$ -mon-1, P+,  $\partial$ -pol

$\mathcal{H}_{14} = 0+, C+, \partial$ -const-1,  $\partial$ -mon-1, P+,  $\partial$ -pol

$\mathcal{H}_{15} = 0+, C+, \partial$ -const-1,  $\partial$ -mon-1, P+

$\mathcal{H}_{16} = 0+, C+, \partial$ -const-1, P+

$\mathcal{H}_{17} = 0+, C+, P+$

$\mathcal{H}_{18} = \text{Intv}, \leq$ -Ax

## Anhang B

# Verzeichnis der komplexen Taktiken von SAPPER

In diesem Anhang sind die im Rahmen dieser Arbeit benutzten komplexen Taktiken zusammengestellt. Hierfür wird jeweils der von der Taktik beschriebene Rechenschritt angegeben und die Taktik formal spezifiziert. Die in den formalen Definitionen verwendeten einfachen Taktiken entsprechen den in Kapitel 6 Abbildung 6.2 aufgeführten Hypothesen.

Im folgenden arbeiten wir stets in einem Ring von Polynomen  $\mathbb{K}[x_1, \dots, x_n]$  über einem Koeffizientenkörper  $\mathbb{K}$ .

### B.1 Taktiken für die Polynomaddition

#### POLYNOMIAL-ADDITION-Taktik

Seien  $p, q \in \mathbb{K}[x_1, \dots, x_n]$  Polynome, dann beschreibt POLYNOMIAL-ADDITION den Rechenschritt:

$$(p \oplus q) = (p + q)$$

und hat als formale Definition:

*Tactic* POLYNOMIAL-ADDITION APPLY(P+)

**POP-FIRST-Taktik**

Sei  $a \in \mathbb{K}[x_1, \dots, x_n]$  ein Monom, und seien  $b, c \in \mathbb{K}[x_1, \dots, x_n]$  Polynome, dann beschreibt POP-FIRST den Rechenschritt:

$$((a + b) + c) = (a + (b + c))$$

und hat als formale Definition:

*Tactic* POP-FIRST COND(((a + b) + c) APPLY(A+))

**POP-SECOND-Taktik**

Seien  $a, c \in \mathbb{K}[x_1, \dots, x_n]$  Polynome, und sei  $b \in \mathbb{K}[x_1, \dots, x_n]$  ein Monom, dann beschreibt POP-SECOND den Rechenschritt:

$$(a + (b + c)) = (b + (a + c))$$

und hat als formale Definition:

*Tactic* POP-SECOND COND((a + (b + c)) APPEND(APPLY(A+)  
 APPLY(C+)  
 APPLY(A+)))  
 ((a + b) APPLY(C+))

**MONOMIAL-ADDITION-Taktik**

Seien  $b, d \in \mathbb{K}[x_1, \dots, x_n]$  Polynome und  $a, c, e \in \mathbb{K}[x_1, \dots, x_n]$  Monome, so daß gilt:  $a =_\varepsilon c$  und  $e = a + c$ , dann beschreibt MONOMIAL-ADDITION den Rechenschritt:

$$((a + b) + (c + d)) = (e + (b + d))$$

und hat als formale Definition:

*Tactic* MONOMIAL-ADDITION COND(((a + b) + (c + d)) APPEND(APPLY(A+)  
 APPLY(C+)  
 APPLY(A+)  
 APPLY(M+)  
 APPLY(A+)))  
 (((a + b) + c) APPEND(APPLY(C+)  
 APPLY(A+)  
 APPLY(M+)))  
 ((a + (c + d)) APPEND(APPLY(A+)  
 APPLY(M+)))  
 ((a + c) APPLY(M+))



**PUSH-LAST-Taktik**

Seien  $a, b \in \mathbb{K}[x_1, \dots, x_n]$  Monome und  $c, d, e \in \mathbb{K}[x_1, \dots, x_n]$  Polynome mit  $c = e + b$ , dann beschreibt PUSH-LAST den Rechenschritt:

$$((a + c) + d) = (a + (e + (b + d)))$$

und hat als formale Definition:

*Tactic* PUSH-LAST REPEAT-IF(((a + c) + d)) APPLY(A+))

**MULTIPLY-ZERO-LEFT-Taktik**

Sei  $a \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom, dann beschreibt MULTIPLY-ZERO-LEFT den Rechenschritt:

$$(0 * a) = 0$$

und hat als formale Definition:

*Tactic* MULTIPLY-ZERO-LEFT APPLY(0\*)

**MULTIPLY-ZERO-RIGHT-Taktik**

Sei  $a \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom, dann beschreibt MULTIPLY-ZERO-RIGHT den Rechenschritt:

$$(a * 0) = 0$$

und hat als formale Definition:

*Tactic* MULTIPLY-ZERO-RIGHT APPEND(APPLY(C\*)  
APPLY(0\*))

### B.3 Taktiken für die Polynomdifferenziation

#### MONOMIAL-DIFFERENTIATION-Taktik

Sei  $p \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom und seien  $m, m' \in \mathbb{K}[x_1, \dots, x_n]$  Monome, so daß  $m = \alpha x_1^{e_1} \cdot \dots \cdot x_i^{e_i} \cdot \dots \cdot x_n^{e_n}$  und  $m' = (\alpha \cdot e_i) x_1^{e_1} \cdot \dots \cdot x_i^{e_i - 1} \cdot \dots \cdot x_n^{e_n}$  gilt, wobei  $\alpha \in \mathbb{K}$ ,  $e_1, \dots, e_n \in \mathbb{N}$  und  $e_i > 0$  mit  $1 \leq i \leq n$ , dann beschreibt MONOMIAL-DIFFERENTIATION den Rechenschritt:

$$\partial_p(m + p, i) = (m' \oplus \partial_p(p, i)),$$

und hat als formale Definition:

*Tactic* MONOMIAL-DIFFERENTIATION (*i*) COND((*m + p*) APPEND(APPLY(P+)  
 APPLY( $\partial$ -pol)  
 APPLY( $\partial$ -mon-i))  
 ((*m*) APPLY( $\partial$ -mon-i))

#### CONSTANT-DIFFERENTIATION-Taktik

Sei  $p \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom und  $m \in \mathbb{K}[x_1, \dots, x_n]$  ein Monom mit  $m = \alpha x_1^{e_1} \cdot \dots \cdot x_i^0 \cdot \dots \cdot x_n^{e_n}$ , wobei  $\alpha \in \mathbb{K}$ ,  $e_1, \dots, e_n \in \mathbb{N}$ , dann beschreibt CONSTANT-DIFFERENTIATION den Rechenschritt:

$$\partial_p(m + p, i) = (0 \oplus \partial_p(p, i))$$

und hat als formale Definition:

*Tactic* CONSTANT-DIFFERENTIATION (*i*) COND((*m + p*) APPEND(APPLY(P+)  
 APPLY( $\partial$ -pol)  
 APPLY( $\partial$ -const-i))  
 ((*c*) APPLY( $\partial$ -const-i))

#### ADD-MONOMIAL-Taktik

Sei  $p \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom, und sei  $m \in \mathbb{K}[x_1, \dots, x_n]$  ein Monom, dann beschreibt ADD-MONOMIAL den Rechenschritt:

$$(m \oplus p) = (m + p)$$

und hat als formale Definition:

*Tactic* ADD-MONOMIAL APPEND(APPLY(P+)  
 COND((0 + *p*) APPLY(0+)  
 COND((*m + 0*)APPEND(APPLY(C+)  
 APPLY(0+))))))

## B.4 Taktiken für das Sortieren von Polynomen

### POLYNOMIAL-SORTING-Taktik

Seien  $a, b \in \mathbb{K}[x_1, \dots, x_n]$  Monome und  $c, d \in \mathbb{K}[x_1, \dots, x_n]$  Polynome mit  $c = (a + d)$  in normaler Ordnung und  $a <_{\varepsilon} c$ , dann beschreibt POLYNOMIAL-SORTING den Rechenschritt:

$$(c + b) = (b + (a + d))$$

und hat als formale Definition:

*Tactic* POLYNOMIAL-SORTING APPEND(REPEAT-IF(((a + d) + b) APPEND(APPLY(A+)  
 APPLY(C+)  
 APPLY(A+)))  
 APPLY(C+))

### REDUCTION-ON-SUM-Taktik

Seien  $a, b, e \in \mathbb{K}[x_1, \dots, x_n]$  Monome mit  $a =_{\varepsilon} b$  und  $e = (a + b)$ , und sei  $c \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom, dann beschreibt REDUCTION-ON-SUM den Rechenschritt:

$$(a + (b + c)) = (e + c)$$

und hat als formale Definition:

*Tactic* REDUCTION-ON-SUM COND((a + (b + c)) APPEND(APPLY(A+)  
 APPLY(M+))  
 ((a + b) APPLY(M+)))

### REDUCTION-ON-ZERO-Taktik

Sei  $a \in \mathbb{K}[x_1, \dots, x_n]$  ein Polynom, dann beschreibt REDUCTION-ON-ZERO den Rechenschritt:

$$(0 + a) = a$$

und hat als formale Definition:

*Tactic* REDUCTION-ON-ZERO COND((0 + a) APPLY(0+))

# Literaturverzeichnis

- [And80] P. B. Andrews: *Transforming Matings into Natural Deduction Proofs*. In: W. Bibel und R. Kowalski (Herausgeber): *Proceedings of the 5<sup>th</sup> International Conference on Automated Deduction (CADE-5)*, LNCS-87, S. 281–292, Les Arc, Frankreich, 1980. Springer Verlag, Berlin.
- [And86] P. B. Andrews: *An Introduction To Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, Inc., 1986.
- [ASS86] H. Abelson, G. J. Sussman und J. Sussman: *Structure and Interpretation of Computer Programs*. The MIT Press, Cambridge, London, 1986.
- [AvLS95] J. Abbot, A. van Leeuwen und A. Strotmann: *Objectives of OpenMath*. Journal of Symbolic Computation, 11, 1995.
- [BC94] W. Bosma und J. Cannon: *Handbook of Magma Functions*. Sydney, 1994.
- [BE93] W. Bibel und E. Eder: *Methods and Calculi for Deduction*. In: D. Gabbay, C. Hogger und J. Robinson (Herausgeber): *Handbook of Logic in Artificial Intelligence and Logic Programming*, Band 1: Logical Foundations, S. 67–182. Oxford University Press, Oxford, Großbritannien, 1993.
- [BHC95] C. Ballarin, K. Homann und J. Calmet: *Theorems and Algorithms: An Interface between Isabelle and Maple*. In: A. H. M. Levelt (Herausgeber): *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'95)*, S. 150–157. ACM Press, 1995.
- [Ble83] W. W. Bledsoe: *The UT Natural-Deduction Prover*. Technischer Bericht ATP-17B, University of Texas at Austin, April 1983.
- [Boc89] A. V. Bocharov: *DEliA: A System of Exact Analysis of Differential Equations using S. Lee Approach*. Technischer Bericht OWIMEX, Program Systems Institute, Pereslavl-Zalessky, UdSSR, 1989.

- [Boo54] George Boole: *An Investigation of The Laws of Thought*. Macmillan, Barclay, & Macmillan, Cambridge, Großbritannien, 1854.
- [Bos93] S. Bosch: *Algebra*. Springer Verlag, Berlin, 1993.
- [BS91] I. N. Bronstein und K. A. Semendjajew: *Taschenbuch der Mathematik*. B. G. Teubner, Stuttgart, 25. Auflage, 1991.
- [Buc96] Bruno Buchberger: *Mathematische Software-Systeme: Drastische Erweiterung des "Intelligenzniveaus" entsprechender Programme erwartet*. Informatik Spektrum, 19/2:100–101, 1996.
- [Bun88] A. Bundy: *The Use of Explicit Plans to Guide Inductive Proofs*. In: E. Lusk und R. Overbeek (Herausgeber): *Proceedings of the 9<sup>th</sup> International Conference on Automated Deduction (CADE-9)*, LNCS-310, Argonne, Illinois, USA, 1988. Springer Verlag, Berlin.
- [Bün94] R. Bündgen: *Combining Computer Algebra and Rule Based Reasoning*. In: J. Calmet und J. A. Campbell (Herausgeber): *Integrating Symbolic Mathematical Computation and Artificial Intelligence*, LNCS-958, S. 209–223, Second International Conference, AISMC-2 Cambridge, United Kingdom, 3.–5. August 1994. Springer Verlag, Berlin, 1995.
- [BvHHS90] Alan Bundy, Frank van Harmelen, Christian Horn und Alan Smail: *The OYSTER-CIAM system*. In: M. E. Stickel (Herausgeber): *Proceedings of the 10<sup>th</sup> International Conference on Automated Deduction (CADE-10)*, LNCS-449, S. 647–648, Kaiserslautern, Deutschland, 1990. Springer Verlag, Berlin.
- [CAB<sup>+</sup>86] R.L. Constable, S.F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, D.J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, J.T. Sasaki und S.F. Smith: *Implementing Mathematics with the Nuprl Proof Development System*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1986.
- [CGG<sup>+</sup>92] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, Benton L. Leong, Michael B. Monagan und Stephen M. Watt: *First leaves: a tutorial introduction to Maple V*. Springer Verlag, Berlin, 1992.
- [Chu40] A. Church: *A Formulation of the Simple Theory of Types*. Journal of Symbolic Logic, 5:56–68, 1940.
- [CL73] Chin-Liang Chang und Richard Char-Tung Lee: *Symbolic logic and mechanical theorem proving*. Academic Press, San Diego, 1973.

- [CLR92] T. H. Cormen, C. E. Leiserson und R. L. Rivest: *Introduction to Algorithms*, The MIT electrical engineering and computer science series. The MIT electrical engineering and computer science series. MIT Press, Cambridge, Massachusetts, USA, sechste Auflage, 1992.
- [CMP91] D. Clément, F. Montagnac und V. Prunet: *Integrated Software Components: a Paradigm for Control Integration*. In: *Proceedings of the European Symposium on Software Development Environments and CASE Technology*, LNCS-509. Springer Verlag, Berlin, Juni 1991.
- [CN94] Peter Coad und Jill Nicola: *OOP : Objektorientierte Programmierung*. Prentice Hall, München, 1994.
- [CZ92a] Edmund Clarke und Xudong Zhao: *Analytica – A Theorem Prover for Mathematics*. Technical Report CMU//CS-92-117, Carnegie Mellon University, School of Computer Science, Oktober 1992.
- [CZ92b] Edmund Clarke und Xudong Zhao: *Analytica – An Experiment in Combining Theorem Proving and Symbolic Computation*. Technical Report CMU//CS-92-147, Carnegie Mellon University, School of Computer Science, Oktober 1992.
- [CZ92c] Edmund Clarke und Xudong Zhao: *Analytica-A Theorem Prover in Mathematics*. In: *Automated Deduction-CADE-II*, S. 761–763, 11th International Conference on Automated Deduction, Saratoga Springs, New York, 15.–18. Juni 1992.
- [CZ94] Edmund Clarke und Xudong Zhao: *Combining Symbolic Computation and Theorem Proving: Some Problems of Ramanujan*. Technical Report CMU//CS-94-103, Carnegie Mellon University, School of Computer Science, Januar 1994.
- [Dav92] J. H. Davenport: *The AXIOM System*. AXIOM Technical Report TR5/92 (ATR/3) (NP2492), Numerical Algorithms Group, Inc., Downer’s Grove, IL, USA and Oxford, UK, Dezember 1992.
- [DGH<sup>+</sup>94] B. I. Dahn, J. Gehne, T. Honigmann, L. Walther und A. Wolf: *Integrating Logical Functions with ILF*. Technischer Bericht 94-10, Naturwissenschaftliche Fakultät II, Institut für Mathematik, Humboldt Universität zu Berlin, 1994.
- [DST88] J. H. Davenport, Y. Siret und E. Tournier: *Computer Algebra. Systems and Algorithms for Algebraic Computation*. Academic Press, London; San Diego; New York, 1988.

- [EFT92] Heinz-Dieter Ebbinghaus, Jörg Flum und Wolfgang Thomas: *Einführung in die mathematische Logik*. BI Wissenschaftsverlag, Mannheim, dritte Auflage, 1992.
- [EHH<sup>+</sup>92] H.-D. Ebbinghaus, H. Hermes, F. Hinzebruch, M. Koecher, K. Mainzer, J. Neukirch, A. Prestel und R. Remmert: *Zahlen*. Springer Verlag, Berlin, dritte Auflage, 1992.
- [EO87] N. Eisinger und H. J. Ohlbach: *Grundlagen und Beispiele*. In: K. H. Bläsius und H. J. Bürckert (Herausgeber): *Deduktionssysteme: Automatisierung des logischen Denkens*, Kapitel II, S. 22–83. Oldenbourg Verlag, München, erste Auflage, 1987.
- [FGT90] William M. Farmer, Joshua D. Guttman und F. Javier Thayer: *IMPS: An Interactive Mathematical Proof System*. System Report, MITRE Corporation, Bedford, MA01730 USA, 1990.
- [Fie96] Armin Fiedler: *Mikroplanungstechniken zur Präsentation mathematischer Beweise*. Diplomarbeit, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1996.
- [Fit90] M. Fitting: *First-Order Logic and Automated Theorem Proving*. Springer Verlag, Berlin, 1990.
- [Fre79] Gottlieb Frege: *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Halle, 1879.
- [Fuc96] Benno Fuchssteiner et al. (The MuPAD Group): *MuPAD User's Manual*. John Wiley and sons, Chichester, New York, erste Auflage, März 1996.
- [Gen35] G. Gentzen: *Untersuchungen über das Logische Schließen I und II*. Mathematische Zeitschrift, 39:176–210, 405–431, 1935.
- [Ger70] H. Gericke: *Geschichte des Zahlbegriffs*. Bibliographisches Institut, Mannheim, 1970.
- [GM93] M. J. C. Gordon und T. F. Melham: *Introduction to HOL*. Cambridge University Press, Cambridge, UK, 1993.
- [GMW79] M. J. Gordon, R. Milner und Ch. P. Wadsworth: *Edinburgh LCF: A Mechanized Logic of Computation*, LNCS-78. Springer Verlag, Berlin, 1979.
- [Göd31] Kurt Gödel: *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*. Monatshefte der Mathematischen Physik, 38:173–198, 1931.

- [GPT94] Fausto Giunchiglia, Paolo Pecchiari und Carolyn Talcott: *Reasoning Theories – Towards an Architecture for Open Mechanized Reasoning Systems*. IRST-Technical Report 9409-15, IRST, Trento, Italy, Juni 1994.
- [Gro94] The SIMATH Group: *SIMATH Manual*. Fachbereich Mathematik, Universität des Saarlandes, Saarbrücken, 1994.
- [GT94] F. Giunchiglia und P. Traverso: *Program Tactics and Logic Tactics*. In: *LPAR'94, 5th International Conference on Logic Programming and Automated Reasoning*, Kiev, Ukraine, 16.–21. Juli 1994.
- [HC94] K. Homann und J. Calmet: *Towards an Intelligent Mathematical Environment*. In: F. Giunchiglia, M. Kerber und D. Pastre (Herausgeber): *Proceedings of Workshop From Theorem Provers to Mathematical Assistants – Issues and Possible Solutions – 11th European Conference on Artificial Intelligence (ECAI '94)*, S. 48–53, Amsterdam, Niederlande, 8.–12. August 1994.
- [HC95] K. Homann und J. Calmet: *An Open Environment for Doing Mathematics*. In: M. Wester, S. Steinberg und M. Jahn (Herausgeber): *Proceedings of 1st International IMACS Conference on Applications of Computer Algebra*, Albuquerque, USA, 1995.
- [Hea87] A. C. Hearn: *REDUCE User's Manual: Version 3.3*. Technischer Bericht Rand Corporation, Santa Monica, CA, USA, 1987.
- [Hen50] Leon Henkin: *Completeness in the Theory of Types*. *Journal of Symbolic Logic*, 15:81–91, 1950.
- [Her89] J. Hertzberg: *Planen – Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz*, Informatik-65. B.I. Wissenschaftsverlag, Mannheim, 1989.
- [Heu91] H. Heuser: *Lehrbuch der Analysis*. B. G. Teubner, Stuttgart, siebte Auflage, 1991.
- [Hil04] D. Hilbert: *Grundlagen der Geometrie*, Kapitel Über die Grundlagen der Logik und der Arithmetik, S. 243–258. fünfte Auflage, 1904.
- [HKK<sup>+</sup>92] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts und J. Siekmann:  $\Omega$ -MKRP – *A Proof Development Environment*. SEKI Report SR-92-22, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1992.
- [HKK<sup>+</sup>94a] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts und J. Siekmann:  $\Omega$ -MKRP: *A Proof Development Environment*. In: A. Bundy (Her-

- ausgeber): *Proceedings of the 12<sup>th</sup> International Conference on Automated Deduction (CADE-12)*, LNAI-814, S. 788–792, Nancy, Frankreich, 1994. Springer Verlag, Berlin.
- [HKK<sup>+</sup>94b] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts und J. Siekmann: *KEIM: A Toolkit for Automated Deduction*. In: A. Bundy (Herausgeber): *Proceedings of the 12<sup>th</sup> International Conference on Automated Deduction (CADE-12)*, LNAI-814, S. 807–810, Nancy, Frankreich, 1994. Springer Verlag, Berlin.
- [HKK<sup>+</sup>95] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts und Jörg Siekmann:  *$\Omega$ -MKRP, ein mathematisches Assistenzsystem*. SEKI Working Paper SWP-95-01, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1995.
- [HKRS94a] X. Huang, M. Kerber, J. Richts und A. Sehn: *A declarative language for formulating methods*. In: J. Kunze und H. Stoyan (Herausgeber): *KI-94 Workshops – Extended Abstracts*, S. 277–278, Saarbrücken, September 1994. Gesellschaft für Informatik, Bonn.
- [HKRS94b] X. Huang, M. Kerber, J. Richts und A. Sehn: *Planning Mathematical Proofs with Methods*. *Journal of Information Processing and Cybernetics (formerly: EIK)*, 30(5/6):277–291, 1994.
- [HT93a] J. Harrison und L. Théry: *Extending the HOL Theorem Prover with a Computer Algebra System to Reason About the Reals*. In: C.-J. H. Seger J. J. Joyce (Herausgeber): *Higher Order Logic Theorem Proving and its Applications (HUG '93)*, LNCS-780, S. 174–184. Springer Verlag, Berlin, 1993.
- [HT93b] John Harrison und Laurent Théry: *Reasoning About the Reals: The Marriage of HOL and Maple*. In: A. Voronkov (Herausgeber): *Proceedings of the 4th International Conference on Logic Programming and Automated Reasoning (LPAR '93)*, LNAI-698, S. 351–353, St. Petersburg, Russia, Juli 1993. Springer Verlag, Berlin.
- [Hua94a] Xiaorong Huang: *Human Oriented Proof Presentation: A Reconstructive Approach*. Doktorarbeit, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1994.
- [Hua94b] Xiaorong Huang: *Planning Reference Choices for Argumentative Texts*. In: *Proc. of 7th International Workshop on Natural Language Generation*, S. 145–152, Kennebunkport, Maine, USA, 1994.

- [Hua94c] Xiaorong Huang: *Reconstructing Proofs at the Assertion Level*. In: A. Bundy (Herausgeber): *Proceedings of the 12<sup>th</sup> International Conference on Automated Deduction (CADE-12)*, LNAI-814, LNAI-814, S. 738–752, Nancy, Frankreich, 1994. Springer Verlag, Berlin.
- [Kaj92] Norbert Kajler: *CAS/PI: a Portable and Extensible Interface for Computer Algebra Systems*. In: Paul S. Wang (Herausgeber): *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, S. 376–386, Berkeley, CA, Juli 1992. ACM Press.
- [KdRB91] G. Kiczales, J. des Rivières und D. G. Bobrow: *The Art of the Metaobject Protocol*. The MIT Press, Cambridge, London, 1991.
- [Kee89] S. E. Keene: *Object-Oriented Programming in COMMON LISP: A Programmer's Guide to CLOS*. Addison-Wesley Publishing Company, 1989.
- [KO63] A. Karatsuba und Y. Ofman: *Multiplication of Multidigit Numbers by Automata*. Soviet Physics-Doklady, 1963.
- [Koh92] M. Kohlhase: *Beweissysteme mit Logiken höherer Stufe*. In: K. H. Bläsius und H. J. Bürckert (Herausgeber): *Deduktionssysteme: Automatisierung des logischen Denkens*, Kapitel I, S. 3–21. Oldenbourg Verlag, München, zweite Auflage, 1992.
- [Koh94] M. Kohlhase: *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. Doktorarbeit, Universität des Saarlandes, 1994.
- [Lei86] Gottfried Wilhelm Leibniz: *Projet et Essais pour arriver à quelque certitude pour finir une bonne partie des disputes et pour avancer l'art d'inventer*. In: Karel Berka und Lothar Kreiser (Herausgeber): *Logiktexte*, Kapitel I.2, S. 16–18. Akademie-Verlag, deutsche Übersetzung, 1983, Berlin, 1686.
- [May88] O. Mayer: *Programmieren in COMMON LISP*, BI-Hochschultaschenbücher-638. BI-Wissenschaftsverlag, Mannheim, 1988.
- [McC90] W. McCune: *Otter 2.0*. In: M. E. Stickel (Herausgeber): *Proceedings of the 10<sup>th</sup> International Conference on Automated Deduction (CADE-10)*, LNAI-449, S. 663–664, Kaiserslautern, Deutschland, 1990. Springer Verlag, Berlin.
- [Nes94] D. Nesmith: *KEIM-Manual, Version 1.2*. Universität des Saarlandes, Im Stadtwald, Saarbrücken, 1994.

- [New92] M. C. Newey: *Proof Based Computation*. In: Myla Archer, Jennifer J. Joyce, Karl N. Levitt und Phillip J. Windley (Herausgeber): *Proceedings of the International Workshop on the HOL Theorem Proving System and its Applications*, S. 380–383. IEEE Computer Society Press, 1992.
- [NSS57] Allen Newell, Cliff Shaw und Herbert Simon: *Empirical Explorations with the Logic Theory Machine: A Case Study in Heuristics*. In: *Proceedings of the 1957 Western Joint Computer Conference*, New York, USA, 1957. McGraw-Hill. Reprinted in *Computers and Thought*, Edward A. Feigenbaum, Julian Feldman, Hrsg., New York, USA, 1963.
- [OW90] Thomas Ottmann und Peter Widmayer: *Algorithmen und Datenstrukturen*, Reihe Informatik-Band 70. Wissenschaftsverlag, Mannheim, 1990.
- [Pau90] L. C. Paulson: *Isabelle: The Next 700 Theorem Provers*. *Logic and Computer Science*, S. 361–386, 1990.
- [Pau94] L. C. Paulson: *Isabelle: a generic theorem prover*, LNCS-828. Springer Verlag, Berlin, 1994.
- [Prä92] Axel Präcklein: *The MKRP User Manual*. Technischer Bericht Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1992.
- [Qui67] Willard Van Orman Quine: *Set Theory and its Logic*. The Belknap Press of Harvard University Press, zweite Auflage, 1967.
- [Rob63] L. Robin: *La pensée grecque*. Paris, 1963.
- [Rob65] John Alan Robinson: *A Machine Oriented Logic Based on the Resolution Principle*. *Journal of the ACM*, 12:23–41, 1965.
- [Seh95] Arthur Christian Sehn: *DECLAME – eine deklarative Sprache zur Repräsentation von Methoden*. Diplomarbeit, Fachbereich Informatik, Universität des Saarlandes, Oktober 1995.
- [Sie92] J. Siekmann: *Geschichte und Anwendungen*. In: K. H. Bläsius und H. J. Bürckert (Herausgeber): *Deduktionssysteme: Automatisierung des logischen Denkens*, Kapitel I, S. 3–21. Oldenbourg Verlag, München, zweite Auflage, 1992.
- [SSY94] G. Sutcliffe, C. Suttner und T. Yemenis: *The TPTP Problem Library*. In: Alan Bundy (Herausgeber): *Twelfth International Conference on Automated Deduction, CADE-12*, LNAI-814, S. 252–266, Nancy, France, Juni 1994. Springer Verlag, Berlin.

- [ST89] P. Suppes und S. Takahashi: *An Interactive Calculus Theorem-prover for Continuity Properties*. Journal of Symbolic Computation, 7(1):573–590, Januar 1989.
- [Ste90] G. L. Steele: *COMMON LISP: The Language – second edition*. Digital Press, zweite Auflage, 1990.
- [Ueb94] J. Ueberberg: *Interactive Theorem Proving and Computer Algebra*. In: J. Calmet und J. A. Campbell (Herausgeber): *Integrating Symbolic Mathematical Computation and Artificial Intelligence*, LNCS-958, S. 1–9, Second International Conference, AISMC-2 Cambridge, United Kingdom; Selected Papers, 3.–5. August 1994. Springer Verlag, Berlin, 1995.
- [vdW50] B. L. van der Waerden: *Moderne Algebra*. Springer Verlag, Berlin, dritte Auflage, 1950.
- [Ver92] S. Vere: *Planning*. In: S. C. Shapiro (Herausgeber): *Encyclopedia of Artificial Intelligence*, Band 1, S. 1159–1171. John Wiley & Sons, New York, zweite Auflage, 1992.
- [Wan91] Dongming Wang: *Reasoning about geometric problems using algebraic methods*. Technischer Bericht Linz, 1991.
- [WH87] P. H. Winston und B. K. Horn: *Lisp*. Addison-Wesley, Bonn, 1987.
- [Wir83] Niklaus Wirth: *Algorithmen und Datenstrukturen*. Teubner, Stuttgart, 1983.
- [WiW] *Diplomthemen SS89 Nr. 35*. Fachschaft WiWi, Universität des Saarlandes.
- [Wol91] S. Wolfram: *Mathematica: a System for Doing Mathematics by Computer*. Addison-Wesley, 1991.
- [Zai95] Marek Zaionc: *Fixpoint Technique for Counting Terms in Typed Lambda Calculus*. Technischer Bericht 95-20, Department of Computer Science at State University of New York at Buffalo, April 1995.
- [Zer08] Ernst Zermelo: *Untersuchungen über die Grundlagen der Mengenlehre. I*. Mathematische Annalen, 65:261–281, 1908.
- [Zip93] Richard Zippel: *Effective Polynomial Computation*. Kluwer Academic Press, Boston; Dordrecht; London, 1993.

# Abbildungsverzeichnis

2.1	Beweis für $3 + 2 = 5$ im Kalkül des natürlichen Schließens . . . . .	11
2.2	Der verkürzte ND-Beweis für Gleichung (2.1) . . . . .	12
4.1	Die $\Omega$ -MKRP Beweisentwicklungsumgebung . . . . .	36
4.2	Die CLOS-Definition der Oberklasse von Polynomen in $\mu\mathcal{CAS}$ . . . . .	44
4.3	Die CLOS-Definition der Unterklassen von <code>ca+polynomial</code> . . . . .	44
4.4	Rekursiver Algorithmus zur Polynomaddition in CLOS . . . . .	45
4.5	Taktikausgabe der $\mu\mathcal{CAS}$ -Berechnung von (4.4) . . . . .	46
4.6	Die Systemarchitektur von SAPPER . . . . .	47
4.7	Die Schnittstelle zwischen $\Omega$ -MKRP und $\mu\mathcal{CAS}$ . . . . .	49
4.8	Definition eines abstrakten CAS für $\mu\mathcal{CAS}$ . . . . .	49
4.9	Datenflußdiagramm für die Übersetzungseinheit . . . . .	51
4.10	Methoden für die polymorphe Funktion <code>ca~build-object</code> . . . . .	51
4.11	Schnittstelle zwischen $\Omega$ -MKRP und mehreren CAS . . . . .	52
4.12	Datenflußdiagramm für den Taktikgenerator . . . . .	53
4.13	Expansion der komplexen Taktiken für die Berechnung von (4.4) . . . . .	53
4.14	Datenflußdiagramm für das SAPPER-System . . . . .	54
4.15	Aufruf von $\mu\mathcal{CAS}$ als Black-Box-System . . . . .	55
4.16	Aufruf von $\mu\mathcal{CAS}$ als Beweisplaner . . . . .	55
4.17	Der Beweis von THM durch $\mu\mathcal{CAS}$ als Black-Box-System . . . . .	55
4.18	Der Beweis von THM auf der Faktenebene . . . . .	56
4.19	Die Schnittstelle zwischen $\Omega$ -MKRP und Computeralgebrakomponenten . . . . .	58
5.1	Der algorithmische Rahmen der verschiedenen Polynommultiplikationen . . . . .	60

---

5.2	Die notwendigen Hypothesen für die Polynommultiplikation . . . . .	61
5.3	Iterative Polynommultiplikation . . . . .	63
5.4	Anfang der Taktik-Ausgabe zur Berechnung der Formel in THM . . . . .	64
5.5	Expansion der komplexen Taktiken aus Abbildung 5.4 . . . . .	65
5.6	Von der Kontrolle verwalteten Termpositionen . . . . .	65
5.7	Der Faktenebenenbeweis für den in Abbildung 5.5 dargestellten Beweisplan	66
5.8	Rekursive Polynommultiplikation . . . . .	67
5.9	Polynommultiplikation nach KARATSUBA und OFMAN . . . . .	70
6.1	Analytische Theoreme . . . . .	77
6.2	Die notwendigen Hypothesen für die Polynommanipulationen . . . . .	78
6.3	Beweisskizze des Optimierungsproblems . . . . .	79

# Symbolverzeichnis

$\mathcal{T}_B$	Menge der Basistypen	22
$\mathcal{T}$	Menge der Typen	22
$\tau$	Typfunktion	22
$\Sigma$	Signatur	22
$\neg$	Negationsjunktork	22
$\vee$	Disjunktionsjunktork	22
$\Pi^\alpha$	Quantork	22
$=^\alpha$	Gleichheitsprädikat	22
$\mathcal{V}$	Menge der Variablen	23
$\mathbf{wff}(\Sigma)$	Menge der wohlgeformten Formeln über $\Sigma$	23
$\mathbf{FV}(\mathbf{A})$	Menge der freien Variablen in $\mathbf{FV}(\mathbf{A})$	23
$\rightarrow_\alpha$	$\alpha$ -Reduktion	23
$\rightarrow_\beta$	$\beta$ -Reduktion	23
$\rightarrow_\eta$	$\eta$ -Reduktion	23
$\top$	wahr	24
$\perp$	falsch	24
$\mathcal{I}$	Interpretation der Konstanten	24
$\varphi$	Variablenbelegung	24
$\mathcal{I}_\varphi$	Denotat	24

$\models$	Folgerungssymbol	25
$\neg$	Negationsjunktork	26
$\vee$	Disjunktionsjunktork	26
$\Pi^\alpha$	Quantork	26
$=^\alpha$	Gleichheitsprädikat	26
$\forall$	Allquantork	26
$\exists$	Existenzquantork	26
$\wedge$	Konjunktionsjunktork	26
$\Rightarrow$	Implikationsjunktork	26
$\Leftrightarrow$	Äquivalenzjunktork	26
APPLY	Taktikal	32
APPEND	Taktikal	32
REPEAT-IF	Taktikal	32
COND	Taktikal	32
$\emptyset$	leere Menge	38
$s : \mathbb{N} \rightarrow \mathbb{N}$	Nachfolgerfunktion	38
$p : \mathbb{Z} \rightarrow \mathbb{Z}$	Vorgängerfunktion	38
$\nu$	Typ der reellen Zahlen	38
$+$	Addition auf reellen Zahlen	38
$\ominus$	Subtraktion auf reellen Zahlen	38
$*$	Multiplikation auf reellen Zahlen	38
$/$	Division auf reellen Zahlen	38
$\uparrow$	Exponentiation auf reellen Zahlen	38
SIMPLIFY-NUM	Taktik	39
$\oplus$	Polynomaddition	40

---

$\otimes$ .....	Polynommultiplikation .....	40
$\partial_p$ .....	Differentiation eines Polynoms .....	40
$\int_p$ .....	Integration eines Polynoms .....	40
<i>pplus-rn</i> .....	Polynomaddition in <i>HOL</i> .....	41
<i>pmult-rn</i> .....	Polynommultiplikation in <i>HOL</i> .....	41
<i>pderiv-rn</i> .....	Polynomdifferentiation in <i>HOL</i> .....	41
<i>pinteg-rn</i> .....	Polynomintegration in <i>HOL</i> .....	41
$=_\varepsilon$ .....	exponentengleich .....	43
$>_\varepsilon$ .....	exponentengrößer .....	43
$<_\varepsilon$ .....	exponentenk kleiner .....	43

# Index

$\alpha$ -Reduktion, 23

$\beta$ -Reduktion, 23

$\eta$ -Reduktion, 23

$\lambda$ -Abstraktion, 23

$\lambda$ -Reduktion, 23

$\mu\mathcal{CAS}$ , 35

$\Omega$ -MKRP, 7, 11, 35, 36

    Datenbank, 57

## A

Addition, 38

Äquivalenzjunktoren, 26

Allquantor, 26

Analytica, 14

ANDREWS, 29

Applikation, 23

Aussonderungsaxiom, 38

## B

BALLARIN, 15

Basistypen, 22

Beweisbaum, 27

Beweisplan, 17, 18, 37

Beweisplanung, 17

Beweissuche, 17

Black-Box, 48

BOOLE, 5

BÜNDGEN, 15

## C

CALMET, 15

CAS, 14

    abstraktes, 49

CAS/ $\pi$ , 15

CLARKE, 14

CLOS, 35

COMMON LISP, 35

## D

DECLAME, 18

dedekindsche Schnitte, 38

Denotat, 24

Disjunktionsjunktoren, 26

Division, 38

## E

Erweiterung von Algorithmen, 45

Existenzquantor, 26

Exponent

    exponentengleich, 43

    exponentengrößer, 43

    exponentenkülein, 43

    Nullexponent, 42

Exponentiation, 38

Extensionalitätsaxiom, 38

## F

Faktenebene, 18, 52, 63

falsch, 24

Falsum, 27

Folgerungssymbol, 25

Formel

    allgemeingültige, 25

    gültige, 25

    wohlgeformte, 23

FRAENKEL, 38



- Methoden, 7, 17  
 Minimierung, 73  
 MKRP, 6, 37  
 Modell  
   Henkin-, 25  
   Standard-, 25  
   verallgemeinertes, 25  
 Multiplikation, 38  
 MuPAD, 8
- N**  
 Nachfolgerfunktion, 38  
 Negationsjunktork, 26  
 NEWELL, 6  
 normale Ordnung, 43  
 normale Polynomordnung, 61  
 Notation  
   infix, 23  
   präfix, 23  
 Nuprl, 37
- O**  
 objektorientierte Programmierung, 51  
 OFMAN, 68  
 OpenMath, 15  
 Optimierungsproblem, 73  
 OTTER, 6, 37
- P**  
 PECCHIARI, 15  
 planen, 7, 17  
 Planverfahren, 17  
 Polynom  
   in *HOL*, 39  
   in *μCAS*, 43  
   in  $\Omega$ -MKRP, 39  
   multivariates, 42  
   univariates, 42, 68  
 Polynomdarstellung  
   graddichte, 42  
   expandierte, 42  
   gradarme, 42  
   rekursive, 42  
   variablenarme, 42  
   variablendichte, 42  
 Polynommultiplikation  
   rekursive, 67  
   devide-and-conquer, 68  
   iterative, 62  
 Präfix-Notation, 23  
 Preis, 74  
 Produktionsfaktoren, 74  
*PROVERB*, 37  
 PYTHAGORAS, 5
- Q**  
 Quantor, 26
- R**  
 REDUCE, 14, 19  
 reflexive Halbordnung, 77
- S**  
 SAPPER, 9, 35  
 Schlußregeln, 26  
 semantisches Überladen, 40  
 SHAW, 6  
 Signatur, 22  
 SIMON, 6  
 Standardmodelle, 25  
 Subtraktion, 38  
 SUPPES, 14  
 Syntaxtransformation, 49
- T**  
 TAKAHASHI, 14  
 Taktik, 7, 17, 29, 31, 52  
   Ausgabestatus einer, 31  
   der Faktenebene, 52  
   einfache, 52, 60

- Eingabezustand einer, 31
- komplexe, 52, 60
- Taktikal, 17, 29, 31, 52
- Taktikgenerator, 48, 52, 63, 64
- TALCOTT, 15
- Tautologie, 25
- Theorem, 26
- Theorie, 57
  - der Einheiten, 75
- THÉRY, 14
- totales Minimum, 74
- Träger, 24
- Typ, 22
  - der Individuen, 22
  - der reellen Zahlen, 38
  - der Wahrheitswerte, 22
- Typfunktion, 22
  
- U**
- UEBERBERG, 15
- Übersetzer, 48, 49
  
- V**
- Variable
  - freie, 23
  - gebundene, 23
  - getypte, 23
- Variablenbelegung, 24
- verallgemeinerte Modelle, 25
- Verbose-Mode, 48
- Verbrauchsfunktion, 74
- Vorgängerfunktion, 38
- VERIFY, 14
  
- W**
- wahr, 24
  
- Z**
- Zahlen
  - ganze, 38
  - imaginäre, 5
  - irrationale, 5
  - natürliche, 38
  - rationale, 38
  - reelle, 38
- Zahlenharmonie, 5
- Zeile
  - geplante, 30
  - offene, 30
- ZERMELO, 38
- ZHAO, 14