

Vorschlag eines Dissertationsthemas

Christoph Benzmüller

10. Juli 1995

Inhaltsverzeichnis

1 Einleitung	1
1.1 Automatisches Beweisen und Ω	2
1.2 Einordnung des Dissertationsthemas	4
1.3 Rahmenbedingungen in der AG Siekmann	4
1.4 Aufbau dieses Exposé und Anmerkung	5
2 Gleichheitsbehandlung erster Stufe	5
2.1 Termersetzende Verfahren	6
2.2 Differenzreduzierende Verfahren	7
3 Logik höherer Stufe	10
3.1 Syntax	11
3.2 Semantik	13
3.3 Sortierte Logiken höherer Stufe	15
3.4 Resolution höherer Stufe	15
4 Gleichheitsbehandlung höherer Stufe	18
4.1 Gleichheitsbehandlung durch Leibnizpaare	18
4.2 Termersetzende Verfahren	19
4.2.1 Paramodulation	19
4.2.2 Ordnungsreduzierende Verfahren	20
4.3 Differenzreduzierende Verfahren	21
4.3.1 RUE-Resolution höherer Stufe	21
4.3.2 Rippling-Technik	22
5 Zusammenfassung und Zeitplan	23
5.1 Zeitplan	23

1 Einleitung

Das Ziel der Dissertation ist die Entwicklung und Implementation spezieller Gleichheitsverfahren für das automatische Beweisen höherer Stufe. Diese Verfahren sollen die Deduktionskomponente LEO der Beweisentwicklungsumgebung Ω der AG Siekmann (Universität des Saarlandes) [HKK⁺94b] ergänzen und zu einer adäquaten Behandlung von Gleichheitsproblemen befähigen.

1.1 Automatisches Beweisen und Ω

Ω ist ein interaktives Beweisentwicklungssystem für Anwendungen in der Mathematik. Seit etwa 1990 bildet dieses System den Mittelpunkt der Forschungsinteressen der AG Siekmann. Sowohl die fast zwanzigjährigen Erfahrungen und die Systeme dieser Forschungsgruppe auf dem Gebiet des vollautomatischen Beweisens erster Stufe, als auch neuere Forschungsergebnisse zum Beweisplanen und der Logik höherer Stufe sollen zur Entwicklung eines Systems beitragen, das auf langfristige Sicht den Mathematiker bei der Erstellung komplexer Beweise sinnvoll unterstützen soll.

Die Ideologie der Gruppe knüpft damit, in leicht abgeschwächter Form, an eine alte Idee von Leibniz an: Er wollte zeigen, daß das logische Denken automatisiert werden kann. Im 17. Jahrhundert hatte Leibniz dazu ein sehr ehrgeiziges Forschungsprojekt formuliert. Ziel war die Entwicklung einer universellen formalen Sprache (*lingua charaktieristica*) und eines dazugehörigen formalen Kalküls (*calculus ratiocinator*). Nach Leibniz' Vision sollten zwei debattierende Philosophen ihren Disput mithilfe der *lingua charaktieristica* und des *calculus ratiocinator* aufklären können, nach dem Motto *Calculemus – Rechnen wir es aus*.

Erst 1879 wurde die erste, im Leibniz'schen Sinne annähernd brauchbare, formale Sprache durch G. Frege definiert. Diese Sprache ist heute als Prädikatenlogik erster Stufe (PL1) bekannt und bildet die Grundlage fast aller modernen Deduktionssysteme.

Der erste vollständige Kalkül für die PL1 wurde dann Anfang dieses Jahrhunderts von D. Hilbert [Hil27] aufgestellt. Zu dieser Zeit war ein Grundlagenstreit über die Formalisierbarkeit der Mathematik in der Fachwelt entbrannt. Hilbert wollte diesem ein Ende setzen, indem er die Mathematik vollständig in einem formalen System (Sprache und Kalkül) zu entwickeln versuchte. Dieses Vorhaben scheiterte. 1930 wurde zwar von Gödel die Vollständigkeit¹ des Hilbertkalküls noch nachgewiesen [Göd30], doch dann folgte eine Reihe negativer Ergebnisse. 1931 zeigte Gödel in seinen Unvollständigkeitssätzen [Göd31], daß die Arithmetik in keinem System vollständig axiomatisierbar ist und daß die Konsistenz eines Systems, daß mindestens die Arithmetik enthält, nicht innerhalb des Systems bewiesen werden kann. Der Traum von der Mechanisierbarkeit des logischen Denkens schien geplatzt. Ein Entscheidungsverfahren für beliebige Aussagen konnte nicht mehr länger erwartet werden und der Disput der Leibniz'schen Philosophen war ein für allemal nicht durch *einfaches Ausrechnen* zu beenden – jedenfalls nicht unbedingt.

Es blieb aber die Semantscheidbarkeit der Prädikatenlogik erster Stufe, die durch die Arbeiten J. Herbrand [Her30] und T. Skolem gezeigt worden war. Sie war der Grund, warum das Ziel die Logik zu mechanisieren weiter Bestand hatte, auch wenn man nun in Kauf nehmen mußte, daß ein Beweissystem sich möglicherweise unendlich lange mit einem erfüllbaren Problem beschäftigte, ohne aber jemals ein Ergebnis zu liefern. Während die ersten automatischen Beweiser (im Folgenden auch Deduktionssysteme genannt) nur mäßigen Erfolg hatten, sorgte ab 1965 der Resolutionskalkül von A. Robinson für einen neuen Aufschwung in der Fachwelt. Dieser verhinderte das bis dahin übliche blinde Durchlaufen des Herbrand-Universums² beim Instanziieren von Formeln und ersetzte es durch das Konzept der Unifikation. In der Folgezeit entstanden weitere erfolgreiche Kalküle, wie zum Beispiel der Tableau-Kalkül [Bet69], das Klauselgraph-Verfahren [Kow75] und das Matrix-Verfahren [And76, Bib83]. Die AG Siekmann stützte sich auf das Klauselgraph-Verfahren und entwickelte das MKRP³-Beweissystem, das damals zu den weltweit leistungsfähigsten Systemen gehörte und mit dem ein großer Teil eines mathematischen Lehrbuchs über die Automatentheorie [Deu71] formal nachbewiesen wurde. Dem MKRP-System liegt folgender Ansatz zugrunde, der typisch ist für das vollautomatische Beweisen: Ausgehend von der Eingabe eines Problems (und möglicherweise einiger weniger Zusatzinformationen zur

¹Ein Kalkül ist vollständig für eine Sprache, wenn sich alle logisch gültigen Aussagen der Sprache auch im Kalkül ableiten lassen.

²Die Träger dieses Universums sind die Grundterme der Logik selbst.

³Markraph Karl Refutation Procedure

Beweissteuerung) übernimmt das Beweissystem, sieht man von geringfügigen Möglichkeiten der Interaktion ab, die alleinige Initiative zur Problembearbeitung. Nach Problemeingabe beginnt das System mit seiner Arbeit, indem es Transformationen auf den systeminternen Darstellungen der eingegebenen PL1-Formeln ausführt, und meldet sich erst wieder zurück, wenn es einen Beweis gefunden oder physische Schranken, zum Beispiel der Speicherbelegung, überschritten hat.

Seit Anfang der 90er Jahre setzte in der AG Siekmann ein Paradigmenwechsel ein. Aufbauend auf einem oder mehreren vollautomatischen Beweisern nach obiger Grundidee strebt die AG Siekmann im Ω -Projekt die Entwicklung einer interaktiven Beweisentwicklungsumgebung an – Mensch und Maschine sollen gemeinsam die Lösung schwieriger Probleme angehen und einen maschinenprüfaren, sowie einen für den Menschen verständlichen Beweis liefern. Die wichtigsten Argumente für diesen Paradigmenwechsel sind:

- Trotz ihres relativen Erfolgs (zum Beispiel bei offenen kombinatorischen Problemen) konnten vollautomatische Beweiser nicht annähernd die Leistungsfähigkeit von Menschen und speziell von Mathematikern erlangen. Der Suchraum, der in diesen Systemen beim Beweisen schwieriger mathematischer Probleme entsteht, ist sehr groß und die bisher entwickelten Verfahren zur Suchraumeinschränkung sind zu schwach, als daß die Maschine bei solchen Problemen in Konkurrenz zum Menschen treten könnte.
- Mathematiker verwenden offensichtlich spezielle Techniken, um sich in den immensen Suchräumen zurechtzufinden. Aus einem Gespür für die richtige Beweistechnik in einer bestimmten Problemsituation, entwickeln sie zunächst einen groben Beweisplan, schränken dadurch den Suchraum ein, und füllen erst anschließend die Lücken im Beweisplan auf, bis sie einen fertigen Beweis erhalten. Dieses Vorgehen konnte bisher nicht den Kontext des vollautomatischen Beweisens übertragen werden.

Diese Beobachtungen führten zu folgender Systemkonzeption für Ω [HKK⁺94a]: Ω verbindet eine obere Planungsebene mit einer darunterliegenden Deduktionsebene. Ziel der Planungsebene ist es, ein gegebenes Problem zu strukturieren und in kleinere Unterprobleme zu zerlegen, also einen Beweisplan zu erstellen. Dazu stützt sich das System auf eine (zu erstellende) Datenbank von Beweistaktiken und ermöglicht eine Kommunikation zwischen Mensch und Maschine, um neue Beweistaktiken zu akquirieren. Die Deduktionsebene stellt ein oder mehrere vollautomatische Beweiser zur Verfügung, deren Aufgabe es ist, kleinere Lücken in den groben Beweisplänen zu schließen. Ein gegebenes Problem wird also durch die Planungsebene in kleinere Unterprobleme zerlegt, die dann von der Deduktionsebene gelöst werden sollen.

Für ein solches System ergeben sich folgende Forderungen:

1. Es sollte eine komfortable Benutzerschnittstelle und eine verständliche Arbeitssprache zur Verfügung stellen. Sehr wichtig ist, daß die systeminterne Darstellung von Beweistaktiken, mathematischen Aussagen und Beweisen auf den Menschen zugeschnitten ist.
2. Die Planungsebene soll zurückgreifen auf eine Datenbank mit zahlreichen anwendungsspezifischen Beweistaktiken. Eine solche Datenbank muß angelegt und sollte ständig erweitert werden.
3. Das Deduktionssystem soll in der Lage sein, auch größere Lücken in den Beweisplänen vollautomatisch zu schließen. Dazu müssen leistungsfähige vollautomatische Beweiser zur Verfügung gestellt werden.

Die Eingabe- und Kommunikationssprache \mathcal{POST} des Ω -Systems basiert auf einer sortierten Logik höherer Stufe, erfüllt damit also die erste Forderung, da sie der in der Mathematik üblichen Sprache sehr nahe kommt. Ein aktuelles Forschungsthema ist die Weiterentwicklung

der Planungsebene, sowie die Akquisition von Beweistaktiken zur Erstellung einer Datenbank. Die Deduktionsebene des Systems stellt derzeit lediglich vollautomatische Beweiser erster Stufe bereit. Damit aber auf Planungs- und Deduktionsebene die gleiche Logik verwendet werden kann, sollen diese Beweiser ersetzt werden durch die Logik-Maschine LEO (Logic Engine for Omega), einem vollautomatischen Beweiser höherer Stufe. Die Grundlagen hierzu wurden von Michael Kohlhase in dessen Dissertation [Koh94] gelegt, in der er einen Resolutionskalkül für eine sortierte Logik höherer Stufe entwickelt. Nach der Fertigstellung von LEO wird sich das Ω -System sowohl auf der Planungs- als auch auf der Deduktionsebene auf eine Logik höherer Stufe stützen.

1.2 Einordnung des Dissertationsthemas

Diese Dissertation hat zum Ziel, die Logik-Maschine LEO um adäquate Verfahren zur Gleichheitsbehandlung zu ergänzen. Sie ist damit im Bereich des vollautomatischen Beweisen angesiedelt, steht aber dennoch nicht im Widerspruch zum erläuterten Paradigmenwechsel der Gruppe: Die Logik-Maschine LEO ist natürlich ein wichtiger Bestandteil des Gesamtsystems Ω .

Das Thema knüpft an die Dissertation von Michael Kohlhase [Koh94] an. In dieser Arbeit wird die sortierte Logik höherer Stufe ΣHOL aufbauend auf dem getypten λ -Kalkül von Church [Chu40] eingeführt. Zu dieser Sprache wird eine mengentheoretische Semantik angegeben. Es werden Transformationsregeln zur Unifikation und Prä-Unifikation vorgestellt und die Prä-Unifikation wird dann als zentraler Inferenzmechanismus zum widerlegungsvollständigen Resolutionskalkül ΣHOR erweitert.

Die beabsichtigte Dissertation wird auf diesen Konzepten aufbauend verschiedene Möglichkeiten zur Gleichheitsbehandlung untersuchen. Neben theoretischen Betrachtungen zur Korrektheit und Vollständigkeit von Verfahren wird die Dissertation solche Verfahren, die als potentiell praxistauglich eingestuft werden können, implementieren, in die Logik-Maschine LEO integrieren und an einem Korpus von Beispielen testen.

Die Arbeit wird betreut werden von Prof. Dr.-Ing. Jörg Siekmann und Dr. Michael Kohlhase.

1.3 Rahmenbedingungen in der AG Siekmann

Die AG Siekmann bietet ein außergewöhnlich gutes Umfeld für die beabsichtigte Dissertation. Sie kann auf eine langjährige Tradition im Beweisen erster Stufe zurückblicken und hat mit MKRP einen der leistungsfähigsten Beweiser für die Prädikatenlogik erster Stufe der damaligen Zeit entwickelt. Aufgrund zahlreicher Veröffentlichungen hat sie sich international unter den ersten Adressen im automatischen Beweisen etabliert.

Seit etwa 1990 beschäftigt sich die Gruppe auch eingehend mit Logiken höherer Stufe und der Projektleiter Michael Kohlhase ist dazu ein kompetenter Diskussionspartner.

Als technische Voraussetzung für die Dissertation steht mit KEIM [HKK⁺94a] eine Software-Plattform zur Entwicklung von Deduktionssystemen erster und höherer Stufe bereit, die eine zügige Implementierung der zu untersuchenden Verfahren ermöglichen wird. Auf der Grundlage von KEIM wird zur Zeit der vollautomatische Beweiser LEO implementiert, der dann als Ausgangspunkt für die Entwicklung spezieller Verfahren zur Gleichheitsbehandlung höherer Stufe zur Verfügung stehen wird.

Zahlreiche Erfahrungen zum automatischen Beweisen existieren auch in der VSE⁴-Gruppe am DFKI⁵, die ebenfalls von Prof. Dr.-Ing. Jörg Siekmann geleitet wird und zu der eine enge Verbindung besteht. Diese Gruppe entwickelt ein System zur formalen Spezifikation und Verifikation von Software und kann verstärkt auf praktische Erfahrungen bei der Entwicklung

⁴Verification Support Environment.

⁵Deutsches Forschungsinstitut für Künstliche Intelligenz, Saarbrücken

von Deduktionssystemen verweisen. Eine Säule des VSE-Systems bildet der Induktionsbeweiser INKA[BHHW86], der unter der Leitung von Dieter Hutter weiter verbessert wird.

Insgesamt sind in Saarbrücken zahlreiche grundlegende Beweisverfahren entwickelt oder verbessert worden. Für die AG Siekmann sind zu erwähnen: Die Arbeiten von Norbert Eisinger und Hans Jürgen Ohlbach [Eis88, EOP89] zum Klauselgraphverfahren, die Arbeiten von Karl Hans-Hans Bäsius und Jörg Siekmann [Blä86, BS88] zur Gleichheitsbehandlung im Kontext des Klauselgraphverfahrens (siehe auch Seite 9) und der Resolutionskalkül Σ HOR von Michael Kohlhase (siehe auch Seite 15). Die VSE-Gruppe kann auf die Arbeiten von Dieter Hutter zur Rippling/Colouringtechnik [Hut90] verweisen (siehe auch Seite 9). Am MPI⁶ für Informatik in Saarbrücken entstanden die Arbeiten von L. Bachmair und Harald Ganzinger zur Superposition [BG90, BGLS92] (siehe auch Seite 7).

1.4 Aufbau dieses Exposé und Anmerkung

Kapitel 2 stellt bisher untersuchte Verfahren zur Gleichheitsbehandlung auf erster Stufe vor. Diese sollen als Ausgangspunkt für die Untersuchungen der Dissertation dienen.

In Kapitel 3 wird eine Logik höherer Stufe aufbauend auf dem getypten λ -Kalkül von Church eingeführt und der Resolutionskalkül Σ HOR von Michael Kohlhase skizziert.

Kapitel 4 diskutiert dann erste Ideen zur Gleichheitsbehandlung auf höherer Stufe.

Kapitel 5 faßt das Ziel der Dissertation zusammen und macht Angaben zur Zeitplanung.

Ich möchte mich bei Michael Kohlhase und Dieter Hutter dafür bedanken, daß sie mich bei der Einarbeitung in dieses Thema unterstützten und mir hoffentlich auch weiterhin mit ihrem Rat zur Seite stehen werden. Die ersten Ideen dieser Dissertation zur Gleichheitsbehandlung auf höherer Stufe, die in Kapitel 4 vorgestellt werden, resultieren aus zahlreichen Diskussionen mit den erwähnten Personen.

2 Gleichheitsbehandlung erster Stufe

Bei einem Blick in ein mathematisches Lehrbuch, wie zum Beispiel [Deu71], wird man sofort sehen, daß die Gleichheitsrelation ein fundamentales Objekt in der Mathematik ist. Also muß jedes System zur Formalisierung mathematischer Aussagen die Gleichheitsrelation geeignet berücksichtigen und dieser Schritt wurde auch in der formalen Logik vollzogen. Das gleiche gilt nun für die Mechanisierung der Logik auf dem Rechner: Die Gleichungsrelation muß in geeigneter und effizienter Weise behandelt werden. Dieses Kapitel stellt einige gut untersuchte Verfahren zur Gleichheitsbehandlung auf erster Stufe vor, die als Ausgangspunkt der Untersuchungen dieser Dissertation dienen sollen.

Im Allgemeinen ist die Gleichheitsrelation in Prädikatenlogik erster Stufe nicht endlich axiomatisierbar. Lediglich bei endlicher Signatur ist dies durch die Kongruenzaxiome möglich, die die Gleichheit als eine reflexive, symmetrische und transitive Relation ausweisen und zudem für jedes Prädikats-, Funktions- und Skolemsymbol⁷ eine Substitutionsbedingung formulieren.

In der Praxis des automatischen Beweisens erweist sich diese Definition der Gleichheit aber meist als wenig brauchbar. Der Suchraum eines um die Kongruenzaxiome erweiterten Kalküls ist zu stark verzweigend; dies liegt unter anderem am Symmetriexiom, das zur Folge hat, daß jede Gleichung stets in beiden Richtungen angewendet darf.

Eine geeignetere Möglichkeit besteht darin, die Gleichheit als primitives Symbol mit in die Logik aufzunehmen und diesem eine entsprechende Semantik zu verleihen (Prädikatenlogik erster Stufe mit Gleichheit). Es werden dann nur solche Interpretationen betrachtet, die das Gleichheitssymbol auf eine Relation mit den gewünschten Eigenschaften abbilden. Zahlreiche korrekte

⁶Max Planck Institut für Informatik, Saarbrücken.

⁷Skolemsymbole entstehen bei der Skolemisierung [Sko28], also bei der Eliminierung von Existenzquantoren.

und vollständige Kalküle wurden nach dieser Idee für die Gleichheitsbehandlung erster Stufe entwickelt. Man kann diese grob einteilen in **termersetzende**, und **differenzreduzierende Verfahren**. Die wichtigsten Verfahren bzw. Kalküle beider Gruppen werden kurz erläutert, wobei die ‘Semantik’ eines jeden Kalküls durch die den Kalkül implementierenden Prozeduren gegeben ist.

2.1 Termersetzende Verfahren

Einer der ersten und grundlegenden Ansätze zur Mechanisierung der Gleichheit bildet die **Paramodulationsregel**, die gegen Ende der 60er Jahre von G. Robinson und L. Wos [RW69] vorgeschlagen wurde. Gemeinsam mit Resolutions- und Faktorisierungsregel, sowie einem **Reflexivitätsaxiom** ($x = x$) bildet sie auf Klauselnormalformebene⁸ einen widerlegungsvollständigen Kalkül für die Prädikatenlogik erster Stufe mit Gleichheit⁹.

$$\frac{P[t] \vee N_1 \vee \dots \vee N_n \quad l = r \vee M_1 \vee \dots \vee M_m \quad \sigma(t) = \sigma(l)}{\sigma(P[t \leftarrow r] \vee N_1 \vee \dots \vee N_n \vee M_1 \vee \dots \vee M_m)} \text{Paramodulation}$$

Die Anwendung einer Gleichung ist immer dann erlaubt, falls eine der beiden Seiten einer Gleichung mit einem Unterterm t von P unifizierbar ist¹⁰. Der Unifikator σ wird auf die Resolvente angewendet, die neben den Literalen der ersten Elternklausel auch die Bedingungen der angewendeten Gleichung¹¹ aus der zweiten Elternklausel enthält. Die Paramodulation spannt für praktische Anwendungen aber immer noch einen zu großen Suchraum auf. Selbst wenn man die Ersetzung von Variablen verbietet, kann die Regel nahezu immer auf eine Klauselmenge angewendet werden.

Eine Idee diesen Suchraum einzuschränken ist es, die Anwendung von Gleichungen nur in einer Richtung zu erlauben, wobei sich natürlich die Frage stellt, ob eine solche Einschränkung auch zu einem vollständigen Kalkül führen kann. In [KB70] stellen D. Knuth und P. Bendix ein Verfahren vor, daß versucht, eine gegebene Menge von Gleichungen mithilfe einer Reduktionsordnung, kritischer Paarbildung und Überlagerung von Regeln in ein kanonisches (noethersches und konfluentes) Termersetzungssystem (TES) zu überführen. Terminiert das Verfahren, so liefert es ein System von gerichteten Gleichungen zurück, daß ein Entscheidungsverfahren für Gleichheitsprobleme unter der gegebenen Gleichungstheorie definiert und das anstelle der gegebenen ungerichteten Gleichungen in der Demodulationsregel (gerichtete Paramodulationsregel) eingesetzt werden kann:

$$G = \{s_1 = t_1, \dots, s_n = t_n\} \rightsquigarrow^{\text{Knuth–Bendix}} G' = \{l_1 \rightarrow r_1, \dots, l_m \rightarrow r_m\}$$

$$\frac{P[t] \vee N_1 \vee \dots \vee N_n \quad l \rightarrow r \vee M_1 \vee \dots \vee M_m \quad \sigma(t) = \sigma(l)}{\sigma(P[t \leftarrow r] \vee N_1 \vee \dots \vee N_n \vee M_1 \vee \dots \vee M_m)} \text{Demodulation}$$

⁸Jede Formelmenge der Prädikatenlogik erster Stufe kann in Klauselnormalform (eine Konjunktion von Disjunktionen) transformiert werden. Eine Klauselmenge ist im allgemeinen nicht logisch äquivalent zur ursprünglichen Formelmenge, aber es gilt: Hat die ursprüngliche Formelmenge ein Modell, so auch die zugeordnete Klauselmenge.

⁹Die Symmetrie von $=$ wird implizit vorausgesetzt.

¹⁰Die Notation $P[t]$ beschreibt ein Literal P , daß einen wohlgeformten Subterm t hat, während $P[t \leftarrow r]$ das veränderte Literal P' beschreibt, daß man erhält, wenn man den den Subterm t in P durch den Term r ersetzt.

¹¹ $l = r \vee M_1 \vee \dots \vee M_m$ ist logisch äquivalent zu $\neg M_1 \wedge \dots \wedge \neg M_m \Rightarrow l = r$. $\neg M_1, \dots, \neg M_m$ sind also die Bedingungen unter denen die Gleichung gilt.

Natürlich gibt es Gleichungsmengen zu denen kein äquivalentes kanonisches Termersetzungssystem existieren kann, weil dies sonst im Widerspruch zur Unentscheidbarkeit der Prädikatenlogik erster Stufe mit Gleichheit stehen würde. In vielen Fällen terminiert das Knuth-Bendix-Verfahren also nicht. Als weiterer Unterschied zur Paramodulation taucht die aufwendige allgemeine Unifikation nur noch innerhalb des Knuth-Bendix-Verfahrens bei der Bildung kritischer Paare auf und wird im eigentlichen Kalkül durch die einseitige Unifikation (Semiunifikation) ersetzt.

Es hat sich gezeigt, daß viele praktische Probleme sehr erfolgreich mit dem Knuth-Bendix-Verfahren bearbeitet werden können. Selbst wenn zu einer gegebenen Gleichungsmenge kein äquivalentes kanonisches Termersetzungssystem existiert, können durch ein modifiziertes Vorgehen [Hue81] – bekannt als **Unfailing Completion** – dennoch Gleichheitsbeweise geführt werden. Dabei wird nicht gewartet, bis das Knuth-Bendix-Verfahren terminiert und ein Ergebnis liefert, sondern die Zwischenergebnisse des Knuth-Bendix-Verfahrens (Mengen von gerichteten Gleichungen) werden zum Gleichheitsbeweisen im Kalkül verwendet.

Eine Kombination aus herkömmlicher Resolution und der vorgestellten Idee des Knuth-Bendix-Verfahrens bildet die **Superposition** [Zha88, BG90, BGLS92]. Grundlegende Idee dieses Verfahrens ist es, auch Literale, zum Beispiel $P(f(a, b))$ und $\neg P(f(x, b))$, als (gerichtete) Gleichungen aufzufassen: $P(f(a, b)) \rightarrow \text{true}$ und $P(f(x, b)) \rightarrow \text{false}$. Ein herkömmlicher Resolutionsschritt kann nun als Spezialfall der Paramodulation (bzw. Demodulation) angesehen werden: Ein Paramodulationsschritt zwischen $P(f(a, b)) \rightarrow \text{true}$ und $P(f(x, b)) \rightarrow \text{false}$ mit Unifikator $\sigma = [a/x]$ liefert die Gleichung $\text{true} = \text{false}$ als elementaren Widerspruch und simuliert damit die herkömmliche Resolution auf den ursprünglichen Literalen.

Termersetzende Verfahren, die mit gerichteten Gleichungen arbeiten – wir werden diese im folgenden als **ordnungsreduzierende Verfahren** bezeichnen –, haben sich zur Mechanisierung der Prädikatenlogik erster Stufe mit Gleichheit in der Praxis sehr bewährt. Ihre Stärke liegt in der gewaltigen Suchraumeinschränkung als Folge der Anwendung ausschließlich gerichteter und reduzierender¹² Gleichungen. Sie sind aber essentiell davon abhängig, ob es gelingt starke Reduktionsordnungen zu finden, mithilfe derer gegebene Gleichungsmengen gerichtet werden können. Genau hier vermute ich ein Problem für Logiken höherer Stufe: In Abschnitt 4.2.2 wird angedeutet, warum es möglicherweise sehr schwer sein wird, auf höherer Stufe solche Reduktionsordnungen zu finden.

2.2 Differenzreduzierende Verfahren

Im Gegensatz zu den termersetzenden Verfahren verfolgen die differenzreduzierenden Verfahren einen anderen Ansatz, um den großen Suchraum beim Gleichheitsbeweisen einzuschränken. Sie erlauben größere Schritte im Suchraum, die sich aber am Beweisziel orientieren, das heißt, sie versuchen zunächst wichtige, grobe Schritte zu ermitteln, um sich rasch auf das Beweisziel hinzubewegen und betrachten das Auffüllen der entstehenden Beweislücken dann als ein Unterproblem. Die Klassifizierung von ‘wichtigen Schritten’ und die Durchführung der groben Schritte selbst wird durch Heuristiken gesteuert – deren Qualität in hohem Maße die Qualität des Gesamtverfahrens beeinflußt. Deshalb sind differenzreduzierende Verfahren in ähnlich starker Weise von guten Heuristiken abhängig, wie die termersetzenden Verfahren von starken Reduktionsordnungen.

Das den differenzreduzierenden Verfahren zugrundeliegende Inferenzprinzip ist die Resolution. Die Resolutionsregel wird dabei insofern verallgemeinert, als das sie nun anstelle der syntaktischen Komplementarität zwischen den Resolutionsliteralen (die bei herkömmlicher Re-

¹²Die Anwendung reduzierender Gleichungen auf einen Term hat zur Folge, daß das Resultat der Ersetzung bezüglich der Reduktionsordnung kleiner als der ursprüngliche Term.

solution durch syntaktische Unifikation ermittelt wird) die *E*-Komplementarität beider Literale überprüft. Unter *E*-Komplementarität versteht man die Komplementarität zweier Literale in der Theorie zu einer Gleichungsmenge *E*. Die differenzreduzierenden Verfahren, die nun kurz skizziert werden, unterscheiden sich hauptsächlich in diesem Punkt: Sie verfolgen unterschiedliche Wege, um den Nachweis der *E*-Komplementarität zweier Resolutionsliterale zu erbringen.

Der erste Ansatz basiert auf dem unveränderten Resolutionskalkül und verlagert die Überprüfung der *E*-Komplementarität in die ***E*-Unifikation** [GS89]. Als Erweiterung zur naiven syntaktischen Unifikation¹³, versucht die *E*-Unifikation zwei Terme oder Literale *s* und *t* bezüglich einer vorgegebenen (Gleichungs-)Theorie *E* anzugeleichen. Ziel ist es also, einen allgemeinsten Unifikator σ zu ermitteln, so daß $E \models \sigma(s) = \sigma(t)$. Während die syntaktische Unifikation den Vorteil hat, daß sie entscheidbar ist und immer einen eindeutigen allgemeinsten Unifikator liefert, falls zwei Terme (oder Literale) unifizierbar sind, gilt dies für die *E*-Unifikation im allgemeinen nicht. Aus diesem Grund eignet sich die *E*-Unifikation zunächst nicht als ein allgemeines Verfahren zum Gleichheitsbeweisen, sondern kommt nur für einige (aber oft benötigte) Gleichheitstheorien (z.B. Kommutativität, Idempotenz oder Assoziativität eines Funktionssymbols) in Frage. Ausführliche Betrachtungen zur Unifikationstheorie findet man in [BS94, Kir90].

Sehr ähnlich zur *E*-Unifikation ist die ***E*-Resolution** [Mor69]. Allerdings verlagert diese die Überprüfung der *E*-Komplementarität nicht in die Unifikation, sondern geht dieses Problem in einer veränderten Resolutionsregel an. Die Gleichungen aus *E* werden nun innerhalb der Resolutionsregel dazu verwendet, die Voraussetzungen zur Durchführung des Resolutionsschritts sicherzustellen¹⁴.

$$\frac{\begin{array}{l} P(s_1, \dots, s_m) \vee N \\ \neg P(t_1, \dots, t_m) \vee M \\ C_i := l_i = r_i \vee R_i \quad (i := 1 \dots k) \quad \{l_i = r_i | 1 \leq i \leq k\} \models \sigma(s_j) = \sigma(t_j) \quad (j := 1 \dots m) \end{array}}{\sigma(N \vee M \vee R_1 \vee \dots \vee R_k)} E\text{-Resolution}$$

Man kann die erweiterte Resolutionsregel als Verknüpfung eines herkömmlichen Resolutionsschritts mit mehreren Paramodulationsschritten ansehen. Allerdings wird bei dieser Sichtweise die Paramodulation als Unterprozess der Resolution zielgerichtet dazu eingesetzt, um die beiden Resolutionsliterale unter Verwendung der gegebenen Gleichungsmenge einander anzugeleichen. In der Praxis hat sich gezeigt, daß *E*-Resolutionsschritte meist zu groß sind und einen zu großen Suchraum aufspannen. Wie schon bei der *E*-Unifikation ist die Unentscheidbarkeit der Gleichheit das größte Problem: Es ist im allgemeinen unmöglich zu entscheiden, ob ein *E*-Resolutionsschritt ausgeführt werden darf oder nicht. Auf jeden Fall sind spezielle Steuerungsmechanismen zum Nachweis der *E*-Komplementarität notwendig, wenn man die Idee der *E*-Resolution sinnvoll in ein Beweissystem umsetzen möchte.

Den Nachteil der *E*-Resolution versucht die **RUE-Resolution** [Dig79] zu vermeiden, indem sie Resolutionsschritte immer erlaubt und die nachzuweisende *E*-Komplementarität der Resolutionsliterale als Nebenbedingungen in der Resolvente vermerkt. Es muß nun also nicht gleichzeitig mit der Regelanwendung, wie bei der *E*-Resolution, auf einen eventuell erfolglosen Gleichheitsbeweis gewartet werden, sondern die Resolvente wird sofort gebildet und zusätzlich wird das noch zu lösende Gleichheitsproblem negiert in der Resolvente vermerkt. Formal läßt sich die RUE-Resolution durch die RUE-Regel und die *NRF*-Regel notieren. Gemeinsam bilden die beiden Regeln einen korrekten und widerlegungsvollständigen Kalkül auf Klauselnormalformebene. Die *NRF*-Regel ist speziell auf die Behandlung der (negierten) Gleichheitsprobleme

¹³Man kann auch die syntaktische Unifikation, als eine *E*-Unifikation ansehen, nämlich als *E*-Unifikation zur leeren Menge von Gleichungen.

¹⁴**N**, **M** und **R_i** stehen für Disjunktionen von Literalen. Zur *E*-Resolution gehört eine zweite Regel, die speziell auf Klauseln mit negativen Gleichungen ausgerichtet ist und die hier nicht vorgestellt wird.

ausgerichtet.

RUE – Regel

(Resolution by Unification and Equality)

$$P(s_1, \dots, s_n) \vee N$$

$$\neg P(t_1, \dots, t_n) \vee M$$

$D = \{[l_1, r_1], \dots, [l_k, r_k]\}$ Differenzmenge von $\sigma(P(s_1, \dots, s_n))$ und $\sigma(\neg P(t_1, \dots, t_n))$

$$\sigma(N) \vee \sigma(M) \vee l_1 \neq r_1 \vee \dots \vee l_k \neq r_k$$

NRF – Regel

(Negative Reflexive Function)

$$C := \{\neg s = t, N\}$$

$D = \{[l_1, r_1], \dots, [l_k, r_k]\}$ Differenzmenge von $\sigma(s)$ und $\sigma(t)$

$$\sigma(N) \vee l_1 \neq r_1 \vee \dots \vee l_k \neq r_k$$

Anmerkungen: In beiden Regeln kann eine beliebige Substitution σ gewählt werden. In praktischen Anwendungen beschränkt man sich aber auf bestimmte Substitution, zum Beispiel auf die leere Unifikation oder auf einen allgemeisten partiellen Unifikator (mgpu). Dieser wird ähnlich wie ein allgemeiner Unifikator durch Unifizieren der Unterterme von links nach rechts berechnet wird, allerdings ohne mit einem Fehlschlag zu stoppen, falls zwei Unterterme nicht unifizierbar sind. Die ermittelte Substitution σ wird dann auf die beiden Resolutionsliterale (beziehungsweise Terme im Fall der NRF-Regel) angewendet und eine Differenzmenge wird berechnet. Diese Differenzmenge enthält genau die Informationen über noch bestehende Differenzen zwischen den beiden substituierten Resolutionsliteralen (beziehungsweise Termen). Es können natürlich im allgemeinen unterschiedliche Differenzmengen gebildet werden: Sei $\sigma(L_1) = f(g(a, x), g(a, b))$ und sei $\sigma(L_2) = f(g(a, c), c)$; mögliche Differenzmengen Sets sind: $D_1 = \{[f(g(a, x), g(a, b)), f(g(a, c), c)]\}$, $D_2 = \{[g(a, x), g(a, c)], [g(a, b), c]\}$ und $D_3 = \{[x, c], [g(a, b), c]\}$.

Eine Weiterentwicklung der RUE-Resolution in spezieller Anlehnung an ein von Jörg Siekmann und G. Wrightson [SW80] um die Paramodulation erweitertes Klauselgraphverfahren [Kow75] wurde in der AG Siekmann von Karl-Hans Bläsius [Blä86] vorgestellt. Grundsätzliche Idee der **Gleichheitsgraphen** ist, systematisch nach Substitutionen und Gleichungsketten zu suchen, um die Differenz zweier anzugeleichender Literale oder Formeln sukzessive zu verringern. Das Verfahren tastet sich dabei ausgehend von einem sehr hohen Abstraktionsniveau (zum Beispiel nur die Top-Symbole werden betrachtet) über verfeinerte Ebenen (auch Unterterme müssen angeglichen werden) an die endgültige Lösung heran. Ein wesentlicher Unterschied zur RUE-Resolution besteht in der Auswahl eines partiellen Unifikators.

Zuletzt soll noch eine sehr junge Idee skizziert werden, die es ermöglicht die **Rippling/Coullouring-Technik** von Alan Bundy [Bun88] [BvHSI90] beziehungsweise Dieter Hutter [Hut90] für das allgemeine Gleichheitsbeweisen nutzbar zu machen. Die Rippling-Technik (bei Dieter Hutter wird sie Coullouring-Technik genannt) wurde zunächst speziell für das Induktionsbeweisen entwickelt. In einem Induktionsbeweis muß im Induktionsschritt die Induktionshypothese auf den Induktionsschluß angewendet werden. Weil diese Problemsituation in jedem Induktionsbeweis auftritt und im allgemeinen zunächst erhebliche Umformungen des Induktionsschluß erforderlich sind, um die Induktionshypothese auf diesen anwenden zu können, ist es sinnvoll, eine Zielvorstellung für diese Umformungen zu entwickeln und sich dann bei der Suche an dieser zu orientieren. . Die Rippling-Technik betrachtet dazu die syntaktischen Differenzen zwischen Induktionsschluß und Induktionshypothese – diese werden durch zusätzliche Farbinformationen an den einzelnen Untertermen repräsentiert – und versucht die Differenzen durch sukzessive Anwendung gefärbter Gleichungen¹⁵ aus der Datenbasis zu beseitigen. In der eingeschränkten Anwendungsdomäne des Induktionsbeweisens ist die Rippling-Technik so stark, daß eine aufwendige Suche weitgehend vermieden werden kann. Zur Zeit wird diese Technik sehr erfolgreich in den Induktionsbeweisern CLAM [BvHHS90] und INKA [BHHW86] eingesetzt, findet aber auch in weiteren Bereichen, wie zum Beispiel der Programmsynthese [KBB93], Anwendung.

¹⁵Die syntaktischen Differenzen zweier Terme werden durch Farbinformationen an den Symbolvorkommen repräsentiert.

Die Ansatz von Jürgen Cleve und Dieter Hutter [CH94] beschreibt eine Möglichkeit wie die Rippling-Technik speziell in differenzreduzierenden Verfahren sinnvoll verwendet werden kann. Differenzreduzierende Verfahren erlauben es, größere, zielgerichtete Schritte im Suchraum zu tätigen, indem sie die syntaktische Komplementarität der herkömmlichen Resolution durch E-Komplementarität ersetzen. Als Rechtfertigung eines solchen Ableitungsschritts muß also die E-Komplementarität der Resolutionsliterale nachgewiesen, das heißt, ein Gleichheitsbeweis muß geführt werden. Im allgemeinen können diese Gleichheitsbeweise aber selbst sehr kompliziert sein und deshalb benötigt man spezielle Techniken und Heuristiken zur Steuerung.

Zur informellen Erläuterung des Ansatzes von Jürgen Cleve und Dieter Hutter betrachte man folgendes Beispiel:

- Als Rechtfertigung eines Beweisschritts muß die *E*-Komplementarität der beiden Literale L_1 und L_2 gezeigt werden. Die beiden Literale sind durch reine syntaktische Unifikation nicht angleichbar, sondern nur anhand einiger Gleichungen aus der Menge $E = \{E_1, \dots, E_n\}$. Möglicherweise deuten aber syntaktische Kriterien darauf hin, daß eine Gleichung E_i ($1 \leq i \leq n$) besonders wichtig für den Nachweis der *E*-Komplementarität von L_1 und L_2 ist, also auf jedenfall angewendet werden sollte¹⁶.

Mit viel Glück läßt sich die die Gleichung E_i direkt auf L_1 oder L_2 anwenden und die Differenz zwischen beiden Literalen kann vermindert werden. Genau wie die Induktionshypothese meist nicht unmittelbar auf den Induktionsschluß angewendet werden kann, wird die ermittelte, wichtige Gleichung E_i nicht unmittelbar auf eines der Literale L_1 oder L_2 anwendbar sein. Vielmehr werden zunächst Umformungen von L_1 oder L_2 anhand weiterer Gleichungen aus E erfolgen müssen, bevor E_i anwendbar wird. Jürgen Cleve und Dieter Hutter schlagen vor, zur Steuerung dieser Umformungen die Rippling-Technik einzusetzen. Weil die Problemsituation sehr ähnlich zum ursprünglichen Anwendungsgebiet der Rippling-Technik im Induktionsbeweisen ist, sollte sich auf diese Weise der Suchraum für die notwendigen Umformungen erheblich einschränken lassen. Das erläuterte Vorgehen läßt sich kurz zusammenfassen:

1. Auswahl einer Gleichung E_i aus $E = \{E_1, \dots, E_n\}$, die für den Nachweis der *E*-Komplementarität von L_1 und L_2 besonders wichtig erscheint. Ermittelt wird eine solche Gleichung anhand heuristischer Funktionen, die die syntaktische Differenz zwischen L_1 und L_2 untersuchen und dann mit den jeweiligen syntaktischen Eigenschaften der Gleichungen aus E vergleichen.
2. Die Gleichung E_i soll auf L_1 oder L_2 angewendet werden. Ist dies ohne weitere Umformungen eines der beiden Literale nicht möglich, so wird die Rippling-Technik verwendet, um die notwendigen Umformungen des Literals zu steuern, bis E_i angewendet werden kann.

Es darf nicht unerwähnt bleiben, daß der Erfolg dieses Verfahrens sehr stark von der Qualität der Heuristiken unter Punkt 1 abhängen wird und dieser Ansatz bisher kaum in der Praxis untersucht wurde.

3 Logik höherer Stufe

In diesem Abschnitt werden die Grundlagen, auf denen diese Dissertation aufbauen wird, formal eingeführt. Vorgestellt wird die Syntax und Semantik einer Logik höherer Stufe, sowie ein widerlegungsvollständiger Resolutionskalkül.

¹⁶Um die Literale $P(f(a))$ und $\neg P(a)$ anzugeleichen, muß auf jedenfall eine Gleichung angewendet werden, die das Funktionssymbol f eliminiert beziehungsweise einführt

Vorgeschlagen wurde das Resolutionsprinzip als Inferenzmechanismus für die Logik höherer Stufe 1971 von P. Andrews [And71]. Kurze Zeit später erschienen die ersten erfolgreichen Arbeiten zur Mechanisierung und Implementierung der Logik höherer Stufe: [Hue72, JP73, JP76]. Eine Korrektur der auf höherer Stufe nicht korrekten naiven Skolemisierung wird in [Mil83] vorgestellt. In [And89] werden von zur Primitiven Substitution, das heißt, zur Instanziierung von flexiblen Literalen, vorgestellt.

3.1 Syntax

Der einfach getypte λ -Kalkül von Church [Chu40] bildet eine geeignete Grundlage zur Definition von Logiken höherer Stufe. Eine umfassende Einführung in den ungetypten und getypten λ -Kalkül bietet [HS86].

Definition 3.1 (Typen) Die Menge der *Basistypen* sei gegeben durch $\mathcal{BT} := \{o, \iota\}$. Dabei symbolisiert o die Menge der Wahrheitswerte und ι die Menge der Individuen.

Typen werden induktiv definiert:

- a) Jeder Basistyp ist ein Typ.
- b) Sind α und β Typen, so ist auch $\alpha \rightarrow \beta$ ein Typ.

Die Menge der *Typen* \mathcal{T} ist die kleinste aller Mengen, die abgeschlossen sind unter den Konstruktionsschriften a) und b).

Ein **funktionaler Typ** $\alpha \rightarrow \beta$ symbolisiert die Menge der Funktionen mit Urbildtyp α und Bildtyp β .

(Um eine unnötig aufgeblähte Notation zu vermeiden wird festgelegt, daß \rightarrow rechtsassoziativ ist. $\alpha \rightarrow \beta \rightarrow \gamma$ steht demnach als Abkürzung für $(\alpha \rightarrow (\beta \rightarrow \gamma))$.)

Definition 3.2 (Getypte Familie) Eine **getypte Familie von Mengen** \mathcal{D} ist eine Familie von Mengen \mathcal{D}_α , die indexiert ist über einer Menge \mathcal{T} von Typen: $\mathcal{D} := \mathcal{D}_\mathcal{T} := \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$. Eine **getypte Funktion** $\mathcal{I}: \mathcal{D}_\mathcal{T} \rightarrow \mathcal{E}_\mathcal{T}$ ist eine Familie \mathcal{I} von Funktionen, die indexiert ist über einer Menge \mathcal{T} von Typen: $\mathcal{I} := \{\mathcal{I}^\alpha \in \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{E}_\alpha) \mid \alpha \in \mathcal{T}\}$

Während der einfach getypte λ -Kalkül lediglich auf einer abzählbar unendlichen Menge \mathcal{V} von Variablensymbolen aufbaut, möchten wir zur Definition einer Logik höherer Stufe eine weitere Basismenge voraussetzen: Eine getypte Mengenfamilie $\Sigma_\mathcal{T}$, im folgenden **Signatur** genannt, die fest vorgegebene Funktionskonstanten unterschiedlichen Typs enthält. Jedem Konstantensymbol wird also ein fester Typ zugeordnet und dieser wird berechnet durch die Funktion τ mit $\tau(c) = \alpha$ gdw. $c \in \Sigma_\alpha$. Die Variablensymbole sind dagegen zunächst ungetypt. Ihnen werden wir einen Typ erst bei konkreter Verwendung in einem bestimmten Kontext zuordnen. Verwaltet wird die Typinformation zu den bereits verwendeten Variablensymbolen durch einen **Variablenkontext**. Formal ist ein Variablenkontext eine partielle Funktion $\Gamma: \mathcal{V} \rightsquigarrow \mathcal{T}$ von der Menge der Variablen in die Menge der Typen. Notiert wird Γ durch $[X^1: \alpha^1], \dots, [X^n: \alpha^n]$, wobei

$$\Gamma, [X^n: \beta](Y) = \begin{cases} \Gamma(Y) & ; Y \neq X \\ \beta & ; \text{sonst} \end{cases}$$

Eine Signatur für eine Logik höherer Stufe soll mindestens die Konstantensymbole \neg , \wedge und Π^α (für jeden Typ α) mit den Typen $o \rightarrow o$, $o \rightarrow o \rightarrow o$ und $\alpha \rightarrow o \rightarrow o$ enthalten.

Ausgehend von einer abzählbar unendlichen Menge \mathcal{V} von Variablen und einer Signatur Σ für eine Logik höherer Stufe können wir nun die wohlgeformten Ausdrücke (Terme und Formeln¹⁷)

¹⁷ Während beim zweistufigen Aufbau der Prädikatenlogik erster Stufe explizit zwischen Termen und Formeln unterschieden wird, verschwindet diese Trennung hier fast vollständig. Terme und Formeln werden hier auf gleicher Ebene eingeführt und sind lediglich durch ihren Typ unterscheidbar.

einer Logik¹⁸ höherer Stufe definieren. Jedem Ausdruck wird dabei ein eindeutiger Typ zugeordnet.

Im folgenden gehen wir davon aus, daß eine Menge \mathcal{T} von Typen, eine abzählbar unendliche Menge \mathcal{V} von Variablen, sowie eine Signatur Σ für eine Logik höherer Stufe gegeben ist.

Definition 3.3 (Wohlgeformte Ausdrücke) Für jedes $\alpha \in \mathcal{T}$ definieren wir die Menge der Ausdrücke $wff_\alpha(\Sigma, \Gamma)$ zum Typ α induktiv:

1. $\Sigma_\alpha \subseteq wff_\alpha(\Sigma, \Gamma)$
2. Ist $X \in \mathcal{V}$ und $\Gamma(X) = \alpha$, so ist $X \in wff_\alpha(\Sigma, \Gamma)$
3. Ist $\mathbf{A} \in wff_{\beta \rightarrow \alpha}(\Sigma, \Gamma)$ und $\mathbf{B} \in wff_\beta(\Sigma, \Gamma)$, so ist $(\mathbf{AB}) \in wff_\alpha(\Sigma, \Gamma)$
4. Ist $\mathbf{A} \in wff_\alpha(\Sigma, (\Gamma, [X:\beta]))$, so ist $(\lambda X_\beta.\mathbf{A}) \in wff_{\beta \rightarrow \alpha}(\Sigma, \Gamma)$

Ausdrücke der Form (\mathbf{AB}) beziehungsweise $(\lambda X_\beta.\mathbf{A})$ werden **Applikation** beziehungsweise **Abstraktion** genannt.

Folgende Vereinbarungen zur Schreibweise wollen wir voraussetzen: Variablen werden durch Großbuchstaben (X) repräsentiert, Konstanten durch Kleinbuchstaben (c). Für λ -Ausdrücke sind Großbuchstaben in Fettdruck vorgesehen (\mathbf{A}). Falls explizit ein Hinweis auf den Typ eines Ausdrucks erfolgen soll, wird ein zusätzlicher Typindex verwendet (\mathbf{A}_α). Um ein unnötiges Aufblähen von Ausdrücken zu vermeiden wird vereinbart, daß ein Teilausdruck $\lambda X^1 \dots X^n.\mathbf{AE}^1 \dots \mathbf{E}^m$ und $\lambda X^1 \dots \lambda X^n.\mathbf{AE}^1 \dots \mathbf{E}^m$ stellvertretend für $(\lambda X^1(\lambda X^2(\dots(\lambda X^n(\dots((\mathbf{AE}^1)\mathbf{E}^2)\dots\mathbf{E}^m))\dots))$ steht, das heißt, bei Applikationen wird eine implizite Linksassoziativität und bei Abstraktionen eine implizite Rechtsassoziativität angenommen. Weil es zu einem intuitiveren Verständnis beiträgt wird ferner $\Pi^\alpha(\lambda X_\alpha.\mathbf{A})$ durch $\forall X_\alpha.\mathbf{A}$ ersetzt und $\exists X_\alpha.\mathbf{A}$ steht wiederum als Abkürzung für $\neg\forall X_\alpha.\neg\mathbf{A}$.

Wir benötigen folgende wichtige Begriffe:

Definition 3.4 (Freie und gebundene Variablen, geschlossene Formeln) Ein Vorkommen der Variable X in einem Teilausdruck $\mathbf{T} = \lambda X.\mathbf{E}'$ von \mathbf{E} heißt **gebunden** in \mathbf{T} beziehungsweise \mathbf{E} . Ein Vorkommen von X heißt **frei** in \mathbf{E} , falls X nicht gebunden ist in \mathbf{E} . Ein Ausdruck wird als **geschlossen** bezeichnet, falls er keine freie Variablen enthält.

Nun wird ein konfluentes und terminierendes Ersetzungssystem ($\rightarrow_{\beta\eta}$) für die eingeführte Sprache der wohlgeformten Ausdrücke angegeben. Dieses kanonische Ersetzungssystem induziert eine Äquivalenzrelation ($\leftrightarrow_{\beta\eta}$) auf der definierten Sprache höherer Stufe. Dieser Äquivalenzrelation kommt im Folgenden eine zentrale Bedeutung zu: Ausdrücke, die in der Relation $\leftrightarrow_{\beta\eta}$ zueinander stehen, werden miteinander identifiziert.

Bevor wir dieses Ersetzungssystem definieren können, müssen wir ein Konzept einführen, daß es uns erlaubt, von den Namen gebundener Variablen zu abstrahieren. Analog zu den durch Quantoren gebundenen Variablen der Prädikatenlogik erster Stufe erlauben wir die Umbenennung von Variablen, die in Abstraktionen gebunden sind (zum Beispiel $(\lambda X_\alpha.X)$ wird zu $(\lambda Y_\alpha.Y)$). Im folgenden gehen wir nun davon aus, daß wir in zwei Ausdrücken \mathbf{C} und \mathbf{D} durch Variablenbenennung stets dafür sorgen können, daß gebundene Variablen nirgends sonst auftreten, also Namenseindeutigkeit vorliegt. Insbesondere bei der folgenden Definition stezen wir die Namenseindeutigkeit der Ausdrücke \mathbf{C} und \mathbf{D} voraus.

¹⁸Korrechterweise müßte man stets von Logiken höherer Stufe sprechen, weil jede konkrete Signatur und Variablenmenge eine konkrete Logik höherer Stufe definiert. Denn man kann die folgende Definition als parametrisiert über der vorausgesetzten Signatur und Variablenmenge betrachten. Von dieser Betrachtung wird im folgenden jedoch meist abstrahiert und es wird nur von **Logik** oder **Sprache** höherer Stufe die Rede sein.

Definition 3.5 (Reduktionen $\rightarrow_\beta, \rightarrow_\eta, \rightarrow_{\beta\eta}$) Für Ausdrücke aus $wff(\Sigma, \Gamma)$ werden folgende Ersetzungsregeln definiert

- $(\lambda X.C)D \rightarrow_\beta [D/X]C$
- Falls X nicht frei in C auftritt, dann $(\lambda X.CX) \rightarrow_\eta C$.
- $E \rightarrow_{\beta\eta} E'$ gdw. $(E \rightarrow_\beta E' \text{ oder } E \rightarrow_\eta E')$

Die reflexiven, transitiven Hüllen der Relationen werden durch $\rightarrow_\beta^*, \rightarrow_\eta^*$ und $\rightarrow_{\beta\eta}^*$ bezeichnet. Gegeben sie ein Ausdruck A . Ist eine der Regeln \rightarrow_λ für $\lambda \in \{\beta, \eta, \beta\eta\}$ auf einen wohlgeformten Unterterm C von A anwendbar und liefert einen Term B , so spricht man von einer **Ein-Schritt- λ -Reduktion** und der ersetzte Unterterm C wird **λ -Redex** genannt.

Theorem 3.6 (Terminierung und Konfluenz) $\rightarrow_\beta^*, \rightarrow_\eta^*, \rightarrow_{\beta\eta}^*$ sind **terminierend und konfluent**.

(Beweis: [HS86])

Definition 3.7 (Normalformen) Weil die Reduktionen \rightarrow_λ für $\lambda \in \{\beta, \eta, \beta\eta\}$ terminierend und konfluent sind, kann jeder Ausdruck A in eine eindeutige **λ -Normalform** überführt werden, das heißt, in eine Form A' , die keinen λ -Redex enthält, also nicht mehr λ -reduzierbar ist. Eine weitere wichtige Normalform ist die **Kopfnormalform**. Man definiert, daß ein Ausdruck A in Kopfnormalform ist, falls er die Form $A = \lambda X_1 \dots X_n (h E_1 \dots E_m)$ hat ($m, n \geq 0$) und $h \in \mathcal{V}$ oder $h \in \Sigma$. Der Teil $\lambda X_1 \dots X_n$ heißt **Binder**, $(h E_1 \dots E_m)$ **Matrix** und h **Kopf** von A . Gilt $h \in \Sigma$, so wird A **starr**, sonst **flexibel** genannt. Falls $h = X_i$ für $1 \leq i \leq n$, dann wird A als **i-te Projektion** bezeichnet.

3.2 Semantik

Zur Konstruktion einer Semantik für die soeben eingeführten Sprache $wff(\Sigma, \Gamma)$ gehen wir stufenweise vor. Zuerst führen wir **Prä- Σ -Strukturen** ein.

Definition 3.8 (Prä- Σ -Struktur) Eine Prä- Σ -Struktur ist ein Tripel $(\mathcal{D}, @, \mathcal{I})$ bestehend aus einer **Familie \mathcal{D} von Trägermengen** \mathcal{D}_α zu den Typen $\alpha \in \mathcal{T}$, einem **Applikationsoperator** $@$, das heißt, einer Familie $@ := \{@^{\alpha\beta}: \mathcal{D}_{\alpha \rightarrow \beta} \times \mathcal{D}_\alpha \rightarrow \mathcal{D}_\beta \mid \alpha, \beta \in \mathcal{T}\}$ von Abbildungen und einer **(Konstanten-)Interpretationsfunktion** $\mathcal{I}: \Sigma \rightarrow \mathcal{D}$.

Darauf aufbauend können wir nun die Auswertung von Ausdrücken unserer Sprache höherer Stufe definieren. Zusätzlich zu einer Prä- Σ -Struktur und einem Variablenkontext Γ benötigen wir für die **Auswertung** von Variablen eine **Γ -Variablenbelegung**, also eine getypte Funktion $\varphi: \text{Dom}(\Gamma) \rightarrow \mathcal{D}$.

Definition 3.9 (Auswertung von Ausdrücken) Gegeben sei eine Prä- Σ -Struktur $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$, ein Variablenkontext Γ und eine Γ -Variablenbelegung $\varphi: \text{Dom}(\Gamma) \rightarrow \mathcal{D}$. Die **(homomorphe) Erweiterung** von $\mathcal{I}: \mathcal{V} \rightarrow \mathcal{D}$ zu $\mathcal{I}_\varphi: wff(\Sigma, \Gamma) \rightarrow \mathcal{D}$ wird induktiv definiert:

1. $\mathcal{I}_\varphi(X) = \varphi(X)$, für jede Variable $X \in \mathcal{V}$
2. $\mathcal{I}_\varphi(c) = \mathcal{I}(c)$, für jede Konstante $c \in \Sigma$
3. $\mathcal{I}_\varphi(AB) = \mathcal{I}_\varphi(A)@ \mathcal{I}_\varphi(B)$, für jede Applikation AB
4. $\mathcal{I}_\varphi(\lambda X_\alpha.B_\beta) = \begin{cases} f \in \mathcal{D}_{\alpha \rightarrow \beta} \text{ mit } f @ z := \mathcal{I}_{\varphi, [z/X]}(B_\beta) & \text{falls } f \text{ existiert} \\ \text{undefined} & \text{sonst} \end{cases}$

Damit die Auswertung unter 4. auch wirklich eine eindeutige Funktion f liefert, muß von der vorausgesetzten Prä- Σ -Struktur \mathcal{A} eine zusätzliche Eigenschaft gefordert werden: \mathcal{A} muß **funktional** sein, das heißt, es muß für alle $f, g \in \mathcal{D}_{\alpha \rightarrow \beta}$ gelten: $f = g$, falls für alle $a \in \mathcal{D}_\alpha$ gilt $f@a = g@a$.

Wir bezeichnen $d = \mathcal{I}_\varphi(\mathbf{A}_\alpha)$ mit $d \in \mathcal{D}_\alpha$ als **Notat** von \mathbf{A}_α in \mathcal{A} .

Im nächsten Schritt der Semantikkonstruktion werden wir **Σ -Strukturen** definieren. Dabei werden wir nur solche Prä- Σ -Strukturen als Σ -Strukturen ausweisen, für die die Auswertung von Ausdrücken niemals einen undefinierten Wert liefert. Damit ist sichergestellt, daß die Funktionsuniversen von Σ -Strukturen genügend Funktionen enthalten und dadurch eine undefinierte Auswertung von Formeln verhindern.

Definition 3.10 (Σ -Struktur) Gegeben sei eine funktionale Prä- Σ -Struktur $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ und ein Variablenkontext Γ . \mathcal{A} erfülle folgende Bedingung (**Notatpflicht**): Für jede Γ -Variablenbelegung $\varphi: \text{Dom}(\Gamma) \rightarrow \mathcal{D}$ liefert die homomorphe Erweiterung $\mathcal{I}_\varphi: wff(\Sigma, \Gamma) \rightarrow \mathcal{D}$ eine totale Funktion. Dann ist \mathcal{A} eine **Σ -Struktur**.

Zunächst werden wir nun ein sehr allgemeines semantisches Konzept die Logik $wff(\Sigma, \Gamma)$ einführen, nämlich das der Σ -Modell Strukturen. Dazu benötigen wir den Begriff der Σ -Valuation.

Definition 3.11 (Σ -Valuation) Sie $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ eine Σ -Struktur. Eine surjektive, totale Funktion $v: \mathcal{D}_o \rightarrow \{\text{T}, \text{F}\}$ mit

1. $v(\mathcal{I}(\neg)@a) = \text{T}$, iff $v(a) = \text{F}$,
2. $v(\mathcal{I}(\vee)@a@a) = \text{T}$, iff $v(a) = \text{T}$ or $v(b) = \text{T}$,
3. $v(\mathcal{I}(\Pi^\alpha)@f) = \text{T}$, iff $v(f@a) = \text{T}$ for each $a \in \mathcal{D}_\alpha$

heißt Σ -Valuation.

In Σ -Modell-Strukturen wird die Trägermenge \mathcal{D}_o zum Typ o nicht auf die Menge $\{\text{T}, \text{F}\}$ der Wahrheitswerte eingeschränkt, sondern es wird lediglich gefordert, daß eine Σ -Valuation v von \mathcal{D}_o nach $\{\text{T}, \text{F}\}$ existiert.

Definition 3.12 (Σ -Modell-Struktur) Sei $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ eine Σ -Struktur und $v: \mathcal{D}_o \rightarrow \{\text{T}, \text{F}\}$ eine Σ -Valuation. Dann heißt das Quadrupel $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v)$ **Σ -Modell-Struktur**.

Als Spezialfall der Σ -Modell-Strukturen erhält man die **Generalisierten Σ -Modelle**. Diese schreiben die Menge $\{\text{T}, \text{F}\}$ als Trägermenge für den Typ o fest vor und fixieren die Semantik der (Konstanten-)Symbole \neg , \wedge und Π^α gemäß unserer üblichen Intuition.

Definition 3.13 (Verallgemeinerte Σ -Modelle) Gegeben sei eine Σ -Modell-Struktur $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v)$ und sei $(\mathcal{D}, \mathcal{I})$ eine Σ -Algebra. Falls $\mathcal{D}_o = \{\text{T}, \text{F}\}$, dann heißt \mathcal{M} **Verallgemeinertes Σ -Modell**¹⁹. Verallgemeinerte Σ -Modelle werden nach Leon Henkin [Hen50] auch als Henkin-Modelle bezeichnet. Weil jede Σ -Modell-Struktur auch eine Σ -Struktur ist, ist die Notatpflicht erfüllt, das heißt, die Auswertung eines Ausdrucks ist niemals undefiniert. Enthält ein Verallgemeinertes Σ -Modell zusätzlich nur volle Funktionsuniversen, so spricht man von einem **Standard- Σ -Modell**.

¹⁹Die Σ -Valuation v ist nun die identische Funktion und $@$ ist die Funktionensapplikation.

Jede der drei eingeführten semantischen Konzepte Σ -Modell-Struktur, Verallgemeinertes Σ -Modell und Standard- Σ -Modell induziert einen Gültigkeits- und Vollständigkeitsbegriff. Ein wichtiges Ergebnis ist, das bei Betrachtung von Standard- Σ -Modellen keine vollständigen Kalküle möglich sind. Betrachten wir aber Verallgemeinerte Σ -Modelle und lassen damit auch nicht volle Funktionsuniversen zu, so können wir nach Henkin [Hen50] vollständige Kalküle finden. Der Resolutionskalkül Σ HOL aus [Koh94], der in Abschnitt 3.4 vorgestellt wird, ist aber selbst für Verallgemeinerte Σ -Modelle nicht vollständig, sondern nur für Σ -Modell-Strukturen. Weil aber die Klasse der Standard- Σ -Modelle eine Unterklasse der Verallgemeinerten Σ -Modelle ist und die Klasse der Verallgemeinerten Σ -Modelle wiederum eine Unterklasse der Σ -Modell-Strukturen gilt, daß jede Aussage, die wir mit Σ HOR beweisen können, auch in jedem Standard- Σ -Modell gültig sein muß. Der Umkehrschluß gilt natürlich nicht, denn es gibt Aussagen die zwar in Standard- Σ -Modellen gültig sind, nicht aber in Verallgemeinerten Σ -Modellen oder in Σ -Modell-Strukturen.

3.3 Sortierte Logiken höherer Stufe

Im Beweisen auf erster Stufe haben sich sortierte Logiken und damit definierte Kalküle sehr bewährt. Wesentlicher Vorteil sortierter Logiken ist, das aufgrund der zusätzlichen Sorteninformation eine drastische Reduzierung des Suchraums für den Kalkül erreicht werden kann. Die Sorteninformation wirkt dabei als Filter bezüglich der Unifikation und Instanzierung und filtert im Allgemeinen riesige Suchräume aus [Wal83].

Diesen Vorteil möchten wir uns natürlich auch beim Beweisen auf höherer Ebene zunutze machen und wir können Überlegungen anstellen, wie die soeben definierte Sprache höherer Stufe um ein Sortenkonzept erweitert werden kann. Betrachten wir dazu das Typkonzept des einfach getypten λ -Kalküls, das offensichtlich Ähnlichkeiten zu einem einfachen Sortenkonzept für Logiken erster Stufe aufweist. Zweck des Typkonzepts ist es, die Formulierbarkeit von Antinomien und Paradoxien²⁰ zu verhindern. Während solche Aussagen im ungetypten λ -Kalkül ohne Probleme notiert werden können²¹, verhindert das Typkonzept dies im einfach getypten λ -Kalkül²².

Um ein flaches Sortenkonzept in den getypten λ -Kalkül zu integrieren, müssen wir offensichtlich nichts weiter tun, als die Basistypen entsprechend den gewünschten Basissorten wählen, da in diesem einfachen Fall beide Konzepte äquivalent sind. Das Typkonzept repräsentiert dann die gewünschten Sorteninformationen und verhindert gleichzeitig die Formulierbarkeit von Antinomien.

Für den Fall, daß wir mächtigere Sortenkonzepte bereitstellen wollen, also zum Beispiel auch Subsortenbeziehungen beschreiben möchten, schlägt M. Kohlhase [Koh94] vor, Typkonzept und Sortenkonzept voneinander zu trennen. Auf eine formale Definition einer Anreicherung von $wff(\Sigma, \Gamma)$ um ein solch mächtiges Sortenkonzept wird hier aus Umfangsgründen verzichtet. Die beabsichtigte Dissertation soll ihre Betrachtungen aber auf eine solche sortierte Sprache ausdehnen.

3.4 Resolution höherer Stufe

In diesem Abschnitt wird ein Resolutionskalkül Σ HOL für die vorgestellte Logik höherer Stufe (beziehungsweise deren sortierte Erweiterung) skizziert. Dieser wurde innerhalb unserer Arbeitsgruppe von M. Kohlhase entwickelt und in [Koh94] als widerlegungsvollständig für Σ -Modell-Strukturen nachgewiesen.

²⁰Zum Beispiel Russel Paradoxon: Die Menge aller Mengen, die sich nicht selbst enthalten.

²¹Definition Russel's paradocher Menge M : $\forall Q.M(Q) \Leftrightarrow (\text{Menge}(Q) \wedge \neg Q(Q))$.

²²In der Definition von Russel's paradocher Menge M müßte Q sowohl den Typ α als auch den Typ $\alpha \rightarrow o$ haben.

Zwei prinzipielle Probleme mußten beseitigt werden, um solch einen Resolutionskalkül höherer Stufe zu definieren:

Das erste Problem betrifft das Herz der Resolution, die Unifikation. Von der allgemeinen Unifikation höherer Stufe ist bekannt, daß sie unentscheidbar ist [Gol81] und daß es lösbare Unifikationsprobleme gibt, für die kein allgemeinster Unifikator angegeben werden kann [Hue75]. Es werden deshalb (möglicherweise unendliche) Mengen allgemeinster Unifikatoren als Lösungen für Unifikationsprobleme betrachtet und man muß sich mit der – immerhin bleibenden – Semantischeidbarkeit der Unifikation begnügen. Weil der allgemeine Unifikationsalgorithmus, bei der Unifikation zweier flexibler Ausdrücke, einen unendlich verzweigenden Suchraum bearbeitet, ist dieser als Herz eines Resolutionskalküls nicht brauchbar. Deshalb stützen wir uns auf die Prä-Unifikation. Diese ist zwar auch nur semantischeidbar, hat aber den Vorteil, daß sie Paare von flexiblen Ausdrücken als bereits gelöst ansieht und dadurch einen unendlich verzweigenden Suchraum vermeidet (aber auch keine Menge allgemeinster Unifikatoren aufzählt). Außerdem gilt, daß ein Paar von Ausdrücken unifizierbar ist, genau dann wenn es prä-unifizierbar ist. Damit ist die die Prä-Unifikation prinzipiell für den praktischen Einsatz in einem Widerlegungskalkül geeignet. Vollständige Mengen von Unifikatoren interessieren einen dabei weniger, weil es genügt eine geeignete Instanziierung der Formeln zu finden, die sämtliche, für den Widerlegungsbeweis wichtigen, Resolutionsschritte rechtfertigt. Auf Unifikation und Prä-Unifikation auf höherer Stufe wird hier nicht weiter eingegangen, sondern auf [SG89] verwiesen. Eine allgemeine Übersicht zur Unifikationstheorie vermitteln [BS94, Kir90].

Das zweite Problem betrifft die Resolutionsregel selbst. Weil nämlich die Prä-Unifikation nur semantischeidbar ist, kann sie nicht vor der Resolventenbildung dazu verwendet werden, die Komplementarität der beiden Resolutionsliterale zu überprüfen und einen Unifikator zu ermitteln. Im Prinzip ziehen wir deshalb die Resolventenbildung der Überprüfung der Komplementarität vor und vermerken den noch zu erbringenden Komplementaritätsnachweis als Nebenbedingung in der Resolvente. Dadurch werden während des Beweisprozess in den erweiterten Klauseln²³ Gleichheitsprobleme (Nebenbedingungen) angehäuft und gleichzeitig Literale eliminiert, solange bis in einer Klausel nur noch Unifikationsprobleme übrigbleiben, die dann mit den Regeln zur Unifikation bearbeitet werden können. Diese Darstellung ist nur prinzipiell korrekt, denn es gibt eine Regel zur Rückpropagierung gelöster Unifikationsprobleme auf die erweiterte Klausel. Mit dieser Regel können Sackgassen im Suchraum vorzeitig aufgedeckt und vermieden werden.

Der Resolutionskalkül Σ HOR setzt neben einem Prä-Unifikationsalgorithmus einen Algorithmus zur Transformation von Formeln (λ -Ausdrücke vom Typ o) in Klauselnnormalform (CNF) voraus. In [Koh94] werden für den sortierten λ -Kalkül zwei entsprechende Regelsysteme angeben. Ein zentrales Problem bei der Transformation von Formeln in Klauselnnormalform betrifft die Eliminierung von Existenzquantoren. Leider ist die naive Skolemisierung²⁴, wie man sie in der Prädikatenlogik erster Stufe verwendet, in diesem Rahmen nicht korrekt. Sie wird deshalb durch einen Mechanismus ersetzt, der die Abhängigkeiten zwischen existenz- und allquantifizierten Formeln in einer nichtlogischen Form – in sogenannten **Variablenbedingungen** – verwaltet. Die Variablenbedingung einer Klausel ist demnach als Zusatzinformation zur Formel zu verstehen, die die Informationen über existenz- und allquantifizierte Variablen und deren gegenseitige Beziehungen speichert²⁵. Variablen, die aus einem Allquantor stammen, werden dazu mit einem

²³Weil diese Klauseln sowohl normale Literale als auch Unifikationsprobleme enthalten, ist es sinnvoll von erweiterten Klauseln zu sprechen.

²⁴Die naive Skolemisierung auf erster Stufe ersetzt Variablen, die durch Existenzquantoren gebunden sind durch Skolemkonstanten, falls keine Abhängigkeiten zu allquantifizierten Variablen bestehen, oder durch Skolemfunktionen, denen die allquantifizierten Variablen zu denen eine Abhängigkeit besteht als Argumente übergeben werden.

²⁵Beispiel: $\forall X \rightarrow \exists Y, XY = Y$ wird transformiert in die Klausel $XY = Y$ mit der Variablenbedingung (X^+, Y^-) . Diese besagt, daß die Variable Y aus einem Existenzquantor stammt und abhängig ist von der allquantifizierten Variablen X. Anhand dieses Relationseintrags wird im Beweisprozess verhindert, daß für X ein

+ gekennzeichnet (X^+) und diejenigen aus einem Existenzquantor mit einem – (Y^-)²⁶.

Aus Übersichtsgründen werden Variablenbedingungen im Folgenden nicht explizit angegeben, sondern implizit vorausgesetzt.

Bevor nun die Regeln des Resolutionskalküls Σ HOL vorgestellt werden, ist eine weitere Definition erforderlich.

Definition 3.14 (Annotierte Formeln und annotierte Literale) Sei \mathbf{A} ein Ausdruck aus $wff(\Sigma, \Gamma)$ mit Typ o . Dann heißt \mathbf{A}^α , für $\alpha \in T, F$ **annotierte Formel**. Falls \mathbf{A} eine atomare Formel (das heißt, keines der Konstantensymbole \neg , \wedge und Π^α tritt in \mathbf{A} auf) ist, dann heißt \mathbf{A}^α **annotiertes Literal**. Die intuitive Bedeutung der annotierten Formel $(\mathbf{A})^T$ beziehungsweise $(\mathbf{A})^F$ ist: \mathbf{A} sei gültig beziehungsweise \mathbf{A} gelte nicht.

Die Resolutionsregeln aus [Koh94] können nun folgendermaßen dargestellt werden²⁷:

$$\frac{\mathbf{N}^\alpha \vee \mathbf{C} \quad \mathbf{M}^\beta \vee \mathbf{D} \quad \alpha \neq \beta}{\mathbf{C} \vee \mathbf{D} \vee \mathbf{M} \neq? \mathbf{N}} \text{Resolution}$$

$$\frac{\mathbf{M}^\alpha \vee \mathbf{N}^\alpha \vee \mathbf{C}}{\mathbf{M}^\alpha \vee \mathbf{C} \vee \mathbf{M} \neq? \mathbf{N}} \text{Faktorisierung}$$

In beiden Regeln entstehen Unifikationsprobleme, die als Nebenbedingungen vermerkt werden. Auf diese sind die Regeln des Prä-Unifikationsalgorithmus anwendbar.

$$\frac{\mathbf{C} \vee \mathcal{E}}{\mathbf{C} \vee \mathcal{E}'} \text{Prä-Unifikation}$$

\mathcal{E}' geht dabei aus \mathcal{E} durch Anwendung einer der Prä-Unifikationsregeln hervor. Durch die Unifikationsregel wird also die notwendige Verbindung zur Prä-Unifikation hergestellt.

Damit aber gelöste²⁸ Unifikationspaare auch aus einer Klausel entfernt werden können – wir wollen ja letztlich eine leere erweiterte Klausel erzeugen – und damit ein frühzeitiges Aufdecken von Sackgassen möglich ist, wird eine weitere Regel eingeführt, die das Rückpropagieren gelöster Paare auf die erweiterte Klausel vorsieht.

$$\frac{\mathbf{C} \vee \mathcal{E} \vee \mathbf{X} \neq? \mathbf{A}}{\mathcal{C}} \text{Rückpropagierung}$$

\mathcal{C} repräsentiert hier die Klauselnnormalform von $[\mathbf{A}/X](\mathbf{C}) \vee \mathcal{E}$, wobei $[\mathbf{A}/X](\mathbf{C})$ die Substitution von \mathbf{X} durch \mathbf{A} in \mathbf{C} beschreibt. Eine Transformation in Klauselnnormalform ist hier deshalb notwendig, weil die substituierte Formel $[\mathbf{A}/X](\mathbf{C})$ möglicherweise nicht mehr in Klauselnnormalform ist.

Ausdruck substituiert wird, der die Variable Y frei enthält.

²⁶Zusätzlich werden in [Koh94] lokal gebundene Variablen mit 0 markiert.

²⁷C und D repräsentieren Disjunktionen von Literalen und Gleichheitsproblemen.

²⁸Ein Unifikationspaar $X = A$ heißt gelöst im Unifikationsproblem E , wenn X in der Variablenbedingung als positiv vermerkt ist, A für X substituiert werden darf und X nicht in E nicht weiter auftritt.

Eine weitere Regel ermöglicht es, Instanziierungen für flexible Literale zu generieren. Flexible Literale sind Literale deren Kopfsymbole freie Variablen sind.

$$\frac{(FT_1 \dots T_n)^\alpha \vee C}{(FT_1 \dots T_n)^\alpha \vee C \vee F \neq? P} \text{ Primitive Substitution}$$

Dabei ist P ein möglichst allgemeiner Ausdruck (General Binding, siehe [Koh94], Seite 115 und Seite 70) vom selben Typ wie die Variable S , der eine logische Konstante $k \in \{\wedge, \vee, \Pi^\delta | \delta \in \mathcal{T}\}$ als Kopfsymbol einführt, oder P ist der allgemeinsten Projektionsterm auf den i -ten ($i = 1 \dots n$) Unterterm von $(FT_1 \dots T_n)$. Folgendes Beispiel verdeutlicht die Verwendung der eingeführten Regeln: Zum Beweis der gültigen Formel $(\exists X_{o \rightarrow o}. XP)^T$ wird diese zunächst negiert und in Klauselnform überführt: $(XP)^F$ (siehe Transformationsregeln in [Koh94], Seite 111). Um den Widerspruch zeigen zu können, muß die Klausel mit sich selbst resolvieren werden, dies erlaubt die Resolutionsregel aber nicht, weil diese komplementäre Literalannotationen vorschreibt. Mithilfe der primitiven Substitution kann die geforderte Annotations-Komplementarität aber hergestellt werden: $(XP)^F$ wird durch primitive Substitution zu $(XP)^F \vee X \neq? (\lambda Y. \neg ZY)$ und durch Rückpropagierung (dabei wird β -Reduktion und Klauseltransformation angewendet) zu $(ZP)^T$. Nun kann $(XP)^F$ mit $(ZP)^T$ resolvieren werden und übrig bleibt das Unifikationsproblem $XP \neq? XQ$, das durch Unifikation und erneute Rückpropagierung zur leeren Klausel abgeleitet werden kann.

4 Gleichheitsbehandlung höherer Stufe

In diesem Kapitel werden erste Ideen zur Gleichheitsbehandlung auf höherer Stufe vorgestellt. Diese Ideen skizzieren natürlich nur einen Ausgangspunkt für die geplante Forschungsarbeit.

Grundsätzlich soll sich die Vorgehensweise der Dissertation an den Vorarbeiten zur Gleichheitsbehandlung erster Stufe orientieren. Dazu sollen erfolgreiche Verfahren der ersten Stufe aufgegriffen und im neuen Kontext der höheren Stufe auf ihre Übertragbarkeit und Eignung untersucht werden. Dieses Kapitel wird deshalb, analog zu Kapitel 76 2, termersetzende und differenzreduzierende Verfahren in getrennten Abschnitten diskutieren und ich möchte eine erste Einschätzung zur Eignung der Verfahren auf höherer Stufe wagen.

Der folgende Abschnitt soll zunächst begründen, warum die Verwendung von Leibnizdefinition der Gleichheit auf höherer Stufe nicht sinnvoll erscheint.

4.1 Gleichheitsbehandlung durch Leibnizpaare

Die Gleichheitsrelation läßt sich auf höherer Stufe durch den λ -Ausdruck $\lambda X. \lambda Y. \forall P. PX \Rightarrow PY$ formalisieren. Dieser λ -Ausdruck beschreibt die Gleichheitsrelation gemäß ihrer Definition nach Leibniz: Zwei Dinge sind gleich, wenn sie bezüglich all ihrer Eigenschaften gleich sind.

Beispielweise kann die Gleichheit der beiden Ausdrücke (fZ) und Z durch den Ausdruck $(\lambda X. \lambda Y. \forall P. PX \Rightarrow PY)(fZ)Z$ beschrieben werden. Dieser reduziert zu $\forall P. P(fZ) \Rightarrow PZ$ und entspricht damit der Klausel $(P(fZ))^F \vee (PZ)^T$. Solche Klauseln werden im folgenden als Leibnizpaare bezeichnet.

Auf obige Weise kann jede Gleichung in ein entsprechendes Leibnizpaar überführt werden. Diese Leibnizpaare können dann im Resolutionskalkül Σ HOR zum Führen von Gleichheitsbeweisen verwendet werden. Dies soll das folgende Beispiel demonstrieren.

- Die Signatur Σ enthalte unter anderem $f : \nu \rightarrow \iota, a : \iota, b : \iota$ und $c : \iota$

Zu Beweisen sei:

$$\forall F_{\nu \rightarrow \iota}, X_\iota, Y_\iota, Z_\iota. (FXY) = (FYX) \wedge (FX(FYZ)) = (F(FXY)Z) \Rightarrow (fa(fbc)) = (fc(fba))$$

Die negierte Aussage wird zunächst durch Klauseltransformation in die Axiome 1-4 überführt und dann werden die Regeln des Kalküls ΣHOR angewendet²⁹.

- 1 AX : $(P(FXY))^F \vee (P(FYX))^T$
- 2 AX : $(P(FX(FYZ)))^F \vee (P(F(FXY)Z))^T$
- 3 AX : $(Q(fa(fbc)))^T$
- 4 AX : $(Q(fc(fba)))^F$
- 5 PS2, PROP : $(P(FX(FYZ)))^T \vee (P(F(FXY)Z))^F$
- 6 RES 4 + 5, UNI*, PROP* : $(Q(f(fcb)a))^F$
- 7 RES 6 + 1, UNI*, PROP* : $(Q(f(fbc)a))^F$
- 8 RES 7 + 1, UNI*, PROP* : $(Q(fa(fbc)))^F$
- 9 RES 8 + 3, UNI* : \square

Damit steht auf höherer Stufe eine sehr einfache Möglichkeit zur Gleichheitsbehandlung zur Verfügung. Allerdings haben Leibnizpaare einen entscheidenden Nachteil: Sie enthalten je zwei flexible Literale, die den Suchraum für die Regel der primitiven Substitution sehr stark aufzulähen. Deshalb erscheint ein erfolgreiches Verhalten dieses Verfahrens in der Praxis kaum realistisch.

Aus diesem Grund wird es notwendig sein, auch auf höherer Stufe spezielle Kalküle zur effektiven Handhabung der Gleichheitsrelation zu entwickeln.

4.2 Termersetzende Verfahren

Zunächst möchte ich erste Ideen für einen Paramodulationskalkül auf höherer Stufe diskutieren. Im zweiten Unterabschnitt wird dann erläutert, warum ordnungsreduzierende Verfahren auf höherer Stufe vermutlich nicht an ihren Erfolg auf erster Stufe anknüpfen können.

4.2.1 Paramodulation

Folgender Versuch der Definition einer Paramodulationsregel soll als Ausgangspunkt für weitergehende Betrachtungen dienen³⁰. Vorausgesetzt ist natürlich, daß anstelle der Logik ΣHOL nun eine Logik betrachtet wird, deren Signatur um Konstantensymbole $=_\alpha$ für jeden Typ $\alpha \in \mathcal{T}$ erweitert wurde³¹.

$$\frac{(L[t])^\alpha \vee \mathbf{C} \quad (l = r)^T \vee \mathbf{D}}{(L[r])^\alpha \vee \mathbf{C} \vee \mathbf{D} \vee (t = l)^F} \text{Paramodulation}$$

Zu dieser Regel sind einige wichtige Anmerkungen notwendig: Gemeinsam mit den Regeln des Resolutionskalküls ΣHOR soll diese Regel einen widerlegungsvollständigen Kalkül bilden. Die Regeln des Resolutionskalküls werden dazu leicht modifiziert. Während im bisherigen Resolutionskalkül zwischen normalen Literalen und Unifikationsproblemen unterschieden wird³², ist

²⁹Die Kürzel AX, RES, FAK, PS, UNI, PROP stehen für Axiom, Resolution, Faktorisierung, Primitive Substitution, Unifikation und Rückpropagierung. Mehrfachanwendungen von Regeln werden durch * gekennzeichnet. Die Schreibweise RES 2 + 3, UNI*, PROP* besagt, daß die Resolutionsregel auf die Klauseln 2 und 3 angewendet wird, das Ergebnis dann mehrfach mit der Unifikationsregel bearbeitet wird, bevor dann mehrfach eine Rückpropagierung erfolgt.

³⁰Weder Korrektheit noch Vollständigkeit dieses Kalküls wurden bereits formal bewiesen.

³¹ \mathbf{C} und \mathbf{D} repräsentieren Disjunktionen von Literalen. Außerdem wird die symmetrische Verwendung der Literale $(l = r)^T$ implizit vorausgesetzt.

³²Dies ist zwar nicht unbedingt notwendig, wurde aber so gehandhabt, weil in der Sprache $wff(\Sigma, \Gamma)$ kein Gleichheitssymbol vorhanden war, mit dem Unifikationsprobleme hätten ausgedrückt werden können und deshalb das Ungleichheitszeichen \neq als Hilfsymbol eingeführt wurde.

dies nicht länger notwendig: Wir können nun auf die zur Verfügung stehenden Gleichheitssymbole zurückgreifen und Unifikationsprobleme (bisher $M \neq? N$) als negierte Gleichheitsliterale der Form $(A = B)^F$ darstellen. Auch die Regeln zur Unifikation und Rückpropagierung sollen fortan auf negierten Gleichheitsliteralen arbeiten. Dadurch fällt die explizite Unterscheidung zwischen normalen Literalen und Gleichheitsproblemen weg. Dadurch können nun aber die Regeln zur Unifikation, Paramodulation und Resolution auf gleichen Ebenen beliebig interagieren. Möglicherweise können die Unifikationsregeln gemeinsam mit der Paramodulationsregel zur E -Unifikation verwendet werden. Es ist aber abzusehen, daß uneingeschränkte Regelanwendungen den Suchraum in diesem Fall sehr aufblähen werden und es deshalb erforderlich sein wird, die Interaktionsmöglichkeiten stark einzuschränken.

Es ergibt sich ein weiteres angenehmes Nebenresultat dieses Paramodulationskalküls: Es sind keine Reflexivitätsaxiome, oder eine weitere Regel zur Behandlung des Reflexivitätsproblems³³, notwendig, weil jede Klausel der Form $(T = T)^F$ nun durch die Regeln der Unifikation beseitigt werden kann.

Die bereits mit den Leibnizaxiomen bewiesene Formel wird erneut aufgegriffen und nun im Paramodulationskalkül bewiesen³⁴:

- Die Signatur Σ enthalte unter anderem $f : u \rightarrow \iota, a : \iota, b : \iota$ und $c : \iota$

Zu Beweisen sei:

$$\forall F_{\iota \rightarrow u}, X_\iota, Y_\iota, Z_\iota, (FXY) = (FYX) \wedge (FX(FYZ)) = (F(FXY)Z) \Rightarrow (fa(fbc)) = (fc(fba))$$

Die Klauselnnormalformtransformation der negierten Formeln führt zu den Axiomen 1-3.

$$1 \text{ } AX : (FXY = FYX)^T$$

$$2 \text{ } AX : (FX(FYZ) = F(FXY)Z))^T$$

$$3 \text{ } AX : (fa(fbc) = fc(fba))^F$$

$$4 \text{ } PARA \ 3 + 2, \ UNI^*, \ PROP^* : (fa(fbc) = f(fcb)a)^F$$

$$5 \text{ } PARA \ 4 + 1, \ UNI^*, \ PROP^* : (fa(fbc) = f(fbc)a)^F$$

$$6 \text{ } PARA \ 5 + 1, \ UNI^*, \ PROP^* : (fa(fbc) = fa(fbc))^F$$

$$7 \text{ } UNI^* \ 6 : \square$$

Man erkennt, daß die primitive Substitution in diesem Beweis keine Rolle spielt. Vielmehr wurden erst gar keine flexiblen Literale erzeugt, was zur Folge hat, daß auch kein Suchraum für die Regel der primitiven Substitution entsteht.

Obwohl dieser Kalkül im Vergleich zur Leibnizgleichheit sicherlich eine Verbesserung darstellt, ist er aus den gleichen Gründen wie auf erster Stufe für eine praktische Verwendung noch zu schwach: Gleichungen können in beiden Richtungen angewendet werden, was zu einem immer noch zu großen Suchraum führt.

4.2.2 Ordnungsreduzierende Verfahren

Auf erster Stufe haben sich ordnungsreduzierende Verfahren sehr bewährt. Sie beheben den Nachteil der Paramodulation und lassen die Anwendung von Gleichungen nur in einer bestimmten Richtung zu. Eine wesentliche Voraussetzung ist aber, daß man zum Richten der Gleichungen auf starke Reduktionsordnungen zurückgreifen kann. Während sich auf erster Stufe oft geeignete Reduktionsordnungen finden lassen, ist dies für die höhere Stufe zu bezweifeln. Schuld daran ist, daß auf höherer Stufe Kopfsymbole, die aus freien Variablen bestehen, natürlich wie andere freie Variablen auch, ihrem Typ entsprechend beliebig substituiert werden dürfen. Dieser Umstand wird instanziierungsstabile Reduktionsordnungen nur schwerlich ermöglichen. Folgendes Beispiel soll dies belegen:

³³Auf erster Stufe ist das Reflexivitätsaxiom $x = x$ oder eine entsprechende Regel erforderlich, um einen widerlegungsvollständigen Kalkül zu erhalten, denn die Klausel $\neg t = t$ kann sonst nicht widerlegt werden.

³⁴Zusätzlich zu den bereits bekannten Kürzeln steht PARA für die Anwendung der Paramodulationsregel.

- In der Signatur Σ seien unter anderem folgende Funktionssymbole gegeben: a_ℓ, b_ℓ und c_ℓ . Die Gleichungsmenge E enthalte die Gleichung $E_1 := (\lambda X.YT) = b$. Gesucht: Eine Reduktionsordnung mit der man E_1 richten kann.
- Sobald man sich für eine Grundordnung auf den Konstantensymbolen entschieden hat, zum Beispiel $a <_R b <_R c$, wird es schwierig sein die notwendige Instanziierungsstabilität für die Erweiterung dieser Grundordnung zu einer Reduktionsordnung erreichen zu können. Denn wenn man $(\lambda X.YT) = b$ richten möchte zu $(\lambda X.YT) <_R b$, dann führt die Instanziierung beider Seiten mit $[\lambda Z.c/Y]$ aber zu $[\lambda Z.c/Y](\lambda X.YT) \equiv (\lambda X.(\lambda Z.c)T) \rightarrow_\beta \lambda X.c \rightarrow_\eta c$ und zu $[\lambda Z.c/Y](b) \equiv b$. Die Instanziierung der gerichteten Gleichung $(\lambda X.YT) <_R b$ mit $[\lambda Z.c/Y]$ kehrt die gerichtete Gleichung also um ($c <_R b$), das heißt, die geforderte Instanziierungsstabilität wurde nicht erreicht.
- Richtet man dagegen $(\lambda X.YT) = b$ zu $(\lambda X.YT) >_R b$, dann stellt sich das analoge Problem mit der Instanziierung $[\lambda Z.a/Y]$. Denn $[\lambda Z.a/Y](\lambda X.YT) \equiv (\lambda X.(\lambda Z.a)T) \rightarrow_\beta \lambda X.a \rightarrow_\eta a$ und $[\lambda Z.a/Y](b) \equiv b$. Diese Instanziierung führt also zu $a >_R b$, wiederum ein Widerspruch zur geforderten Instanziierungsstabilität.

Sobald eine beliebige Grundordnung auf den Konstantensymbolen fixiert wurde, tritt der beschriebene Effekt auf und deshalb ist zu befürchten, daß sich auf höherer Stufe nur schwer geeignete Reduktionsordnungen finden lassen werden und ordnungsreduzierenden Verfahren kaum an die Erfolge der ersten Stufe werden anknüpfen können. Die Dissertation wird sich aus diesem Grund verstärkt auf differenzreduzierende Verfahren konzentrieren, aber ohne dabei den ordnungsreduzierenden Ansatz ganz aus den Augen zu verlieren.

4.3 Differenzreduzierende Verfahren

Zwei Verfahren des differenzreduzierenden Ansatzes werden im neuen Kontext der höheren Stufe diskutiert: Die RUE-Resolution und die Verallgemeinerte Rippling-Technik von Jürgen Cleve und Dieter Hutter.

4.3.1 RUE-Resolution höherer Stufe

Zwischen der RUE-Resolution erster Stufe (Seite 9) und dem Σ HOR-Kalkül von Kohlhase (Seite 17) besteht offensichtlich eine große Ähnlichkeit. In beiden Ansätzen wird der anvisierte Resolutionsschritt in jedem Fall durchgeführt und der noch zu erbringende Komplementaritätsnachweis zu den Resolutionsliteralen wird in Form von Nebenbedingungen in die Resolvente aufgenommen. Während die Abspaltung des Komplementaritätsnachweises bei der RUE-Resolution primär durch den Gedanken der Suchraumeinschränkung motiviert ist, ist diese Abspaltung für den Σ HOR-Kalkül von essentieller Bedeutung (Seite 16): Sie erst ermöglicht die Widerlegungsvollständigkeit des Kalküls.

Bild 1 soll die Beziehungen von RUE- und Σ HOR-Ansatz graphisch erläutern. Ausgangspunkt der Raute (unten) ist ein Resolutionskalkül mit verzögerter Unifikation auf einer Sprache erster Stufe ohne primitives Gleichheitssymbol, das heißt, betrachtet wird die RUE-Resolution erster Stufe, eingeschränkt auf eine Sprache ohne Gleichheit (dies ist identisch zu normaler Resolution). Dann ergibt sich die RUE-Resolution erster Stufe als Erweiterung dieses Ansatzes, indem man ein primitives Gleichheitssymbol mit in die zugrundegelegte Sprache aufnimmt. Andererseits gelangt man zum Σ HOR-Ansatz, wenn man anstelle einer Sprache erster Stufe zu einer Sprache höherer Stufe übergeht. Die Raute gipfelt im Σ HOR-RUE-Ansatz, zudem man vom Σ HOR-Ansatz gelangt durch eine Erweiterung der Sprache höherer Stufe um ein primitives Gleichheitssymbol und vom RUE-Ansatz erster Stufe aus durch den Übergang zu einer Sprache höherer Stufe.

Abbildung 1: Zusammenhang von Resolution und RUE-Resolution auf erster und höherer Stufe

Gegeben sind bereits Vollständigkeitsbeweise zu den drei unteren Ansätzen der Raute. Einen Beweis der Vollständigkeit des Σ HOR-Ansatzes liefert Kohlhase in [Koh94]. Der Beweis zum allgemeinen RUE-Ansatz erster Stufe [Dig79] kann aus dem Beweis des eingeschränkten Ansatzes [Rob65], also der herkömmlichen Resolution, abgeleitet werden.

Weil es, wie bereits angedeutet wurde, enge Beziehungen zwischen der von Kohlhase untersuchten Resolution höherer Stufe und der RUE-Resolution auf erster Stufe gibt, erscheint es sinnvoll die Erweiterung des Σ HOR-Ansatzes zum Σ HOR-RUE-Ansatz als Ausgangspunkt der Dissertation zu wählen. Zunächst ist es dazu notwendig, die verwendete Sprache höherer Stufe $wff(\Sigma, \Gamma)$ (Seite 12) syntaktisch um ein primitives Gleichheitssymbol zu $wff(\Sigma', \Gamma)$ zu erweitern. Dazu wird eine zugrundegelegte Signatur Σ , die bisher zumindest die Konstanten \neg , \wedge und Π^α (für jeden Typ α) enthalten sollte, um Konstanten $=_\alpha$ (für jeden Typ $\alpha \in \mathcal{T}$) angereichert. Anschließend muß diesen Konstanten eine adäquate Semantik zugeordnet werden, das heißt, das Symbol $=_\alpha$ wird interpretiert als die Gleichheitsrelation auf den Elementen des Funktionenraums zum Typ α . Als ein Vorschlag für einen widerlegungsvollständigen Kalkül soll dann der Ansatz der RUE-Resolution aufgegriffen und auf die Sprache $wff(\Sigma', \Gamma)$ übertragen werden. Danach sollen die Korrektheit und Vollständigkeit dieses Kalküls untersucht werden. Eine Orientierung zur Durchführung des Vollständigkeitsbeweises soll die Untersuchung der Zusammenhänge zwischen den Vollständigkeitsbeweisen zur RUE-Resolution auf erster Stufe und der herkömmlichen Resolution bieten.

4.3.2 Rippling-Technik

Neben der Entwicklung eines vollständigen Kalküls zur Gleichheitsbehandlung wird sich die Dissertation mit der Integration von geeigneten Steuerungsheuristiken beschäftigen. Im Zusammenhang zur erläuterten Idee des Σ HOR-RUE-Kalküls stellt sich insbesondere die Frage, wie man die generierten Gleichheitsprobleme mithilfe von Steuerungsheuristiken möglichst geschickt behandeln kann. Deshalb soll überprüft werden, ob die auf Seite 9 skizzierte Idee von Jürgen Cleve und Dieter Hutter, auf höherer Ebene sinnvolle Anwendung finden kann. Die Transformation dieser Idee auf höhere Stufe setzt einerseits eine Rippling-Technik für die höhere Stufe

voraus und fordert anderseits die Existenz geeigneter heuristischer Funktionen zur Auswahl für ein bestimmtes Beweziel wichtiger Gleichungen aus der Datenbank. Die erste Forderung ist durch eine aktuelle Arbeit von Michael Kohlhase und Dieter Hutter bereits ansatzweise erfüllt. In [HK95] stellen sie eine Erweiterung der Unifikation höherer Stufe vor, die auch zusätzliche Farbinformationen an den λ -Ausdrücken adäquat erfaßt und damit die Grundlage für eine Rippling-Technik auf höherer Stufe legt.

Die zweite Forderung wird möglicherweise schwieriger zu erfüllen sein. Gesucht sind heuristische Funktionen, die anhand ihres Argumentes – einem Gleichheitsproblem – in der aktuellen Datenbank eines Beweissystems Gleichungen nach ihrer Bedeutung für den zu erbringenden Beweis klassifizieren können. Dieser Aspekt war bisher der Schwachpunkt für die Idee von Jürgen Cleve und Dieter Hutter auf erster Stufe und wird es zunächst auch auf höherer Stufe sein.

5 Zusammenfassung und Zeitplan

Das Ziel der Dissertation ist die Entwicklung spezieller und effizienter Verfahren zur Gleichheitsbehandlung auf höherer Stufe.

Ausgangspunkt werden die gut untersuchten Verfahren zur Gleichheitsbehandlung auf erster Stufe sein. Zunächst sollen diese Verfahren im Kontext der höheren Stufe analysiert und untereinander verglichen werden, um dann Aussagen über deren Eignung für die höhere Stufe treffen zu können. Somit ist das Ziel der ersten Arbeitsphase, einzelne Verfahren für die weitere Untersuchungen zu präferieren. Auf den momentanen Stand dieser Untersuchungen wurde in Kapitel 4 bereits näher eingegangen: Die Untersuchungen werden sich demnach ausgehend von der Paramodulation hin zu dem differenzreduzierenden Ansatz der RUE-Resolution verlagern.

In der zweiten Arbeitsphase sollen die ausgewählten Gleichheitsverfahren dann formal im Kontext der höheren Stufe eingeführt und durch Betrachtungen zur Korrektheit und Vollständigkeit ergänzt werden. Wichtigster Aspekt dieser Arbeitsphase wird es sein, die Verfahren zu implementieren und ihre Eignung in der Praxis an mathematischen Problemstellungen zu analysieren. Es soll aber nicht nur das isolierte Verhalten des vollautomatischen Verfahrens höherer Stufe betrachtet werden, sondern auch ein mögliches Zusammenspiel mit dem Beweisplaner des Ω -Systems. Die Untersuchungen sollen letztendlich dazu führen, daß ein bestimmtes Verfahren zur Gleichheitsbehandlung favorisiert werden kann, um es dann durch die Entwicklung spezieller und effizienter Steuerungsheuristiken zu optimieren.

In der dritten Arbeitsphase sollen die Ergebnisse der vorangehenden Phasen zusammengetragen und niedergeschrieben werden. Zudem soll das favorisierte Beweisverfahren möglichst optimal in das Ω -System integriert werden. Ein weiterer Aspekt dieser Arbeitsphase könnte es sein, nach Möglichkeiten zur Entwicklung spezieller Methoden für das Gleichheitsbeweisen auf der Planungsebene zu suchen.

5.1 Zeitplan

Die Dissertation soll in einem Zeitraum von drei Jahren angefertigt werden. Für die erste Arbeitsphase sind etwa 9 Monate vorgesehen. Die sich anschließende, arbeitsaufwendige zweite Phase soll etwa nach weiteren 18 Monaten abgeschlossen werden. Für die Niederschrift stehen dann zunächst weitere 9 Monate zur Verfügung.

Literatur

- [And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 3(36):414–432, 1971.
- [And76] Peter B. Andrews. Refutations by matings. *IEEE Trans. Comp.*, C-25(8):801–807, 1976.

- [And89] Peter B. Andrews. On connections and higher order logic. *Journal of Automated Reasoning*, 5:257–291, 1989.
- [Bet55] E. W. Beth. Semantic entailment and formula derivability. *Medelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, 18(13):309–342, 1955.
- [Bet69] E. W. Beth. Semantic entailment and formula derivability. In J. Hintikka, editor, *The Philosophy of Mathematics*, pages 9–49. Oxford University Press, 1969. First published in [Bet55].
- [BG90] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In Stickel [Sti90], pages 427–441.
- [BGLS92] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation and superposition. In Deepak Kapur, editor, *Proceedings [Kap92]*, pages 462–476, Saratoga Springs, New York, USA, June 1992. Springer Verlag.
- [BHHW86] S. Biundo, B. Hummel, D. Hutter, and C. Walther. The Karlsruhe Induction Theorem Proving System. In Jörg Siekmann, editor, *Proceedings of [Sie86]*, pages 672 – 674. Springer Verlag, 1986.
- [Bib83] Wolfgang Bibel. Matings in matrices. *Communications of the ACM*, 26:844–852, 1983.
- [Blä86] Karl Hans Bläsius. *Equality Reasoning Based on Graphs*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, 1986. Also published as SEKI-Report SR-87-01, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern.
- [BS88] Karl Hans Bläsius and Jörg Siekmann. Partial unification for graph based equational reasoning. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings [LO88]*, pages 397–414, Argonne, Illinois, USA, 1988. Springer Verlag.
- [BS94] Franz Baader and Jörg Siekmann. Unification theory. In Dov Gabbay, editor, *Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1994.
- [Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings [LO88]*, pages 111–120, Argonne, Illinois, USA, 1988. Springer Verlag.
- [Bun94] Alan Bundy, editor. *Proceedings of the 12th Conference on Automated Deduction*, LNAI, Nancy, France, 1994.
- [BvHHS90] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam System. In Stickel [Sti90], pages 647 – 648.
- [BvHSI90] Alan Bundy, Frank van Harmelen, Alan Smaill, and Andrew Ireland. Extensions to the rippling-out tactic for guiding inductive proofs. In Stickel [Sti90], pages 132–146.
- [CH94] Jürgen Cleve and Dieter Hutter. A methodology for equational reasoning. In *Proceedings Hawaii International Conference on System Sciences 27*, 1994.
- [Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
- [Deu71] Peter Deussen. *Halbgruppen und Automaten*, volume 99 of *Heidelberger Taschenbücher, Sammlung Informatik*. Springer Verlag, 1971.
- [Dig79] Vincent J. Digricoli. Resolution by unification and equality. In William H. Joyner, editor, *Proceedings of the 4th Workshop on Automated Deduction*, Austin, Texas, USA, 1979.
- [Eis88] Norbert Eisinger. *Completeness, Confluence, and Related Properties of Clause Graph Resolution*. PhD thesis, Universität Kaiserslautern, 1988.
- [EOP89] Norbert Eisinger, Hans Jürgen Ohlbach, and Axel Präcklein. Elimination of redundancies in clause sets and clause graphs. SEKI-Report SR-89-10, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, October 1989. Also published as: Reduction Rules for Resolution-Based Systems, in *Artificial Intelligence* 50 (1991), pages 141–181, Elsevier Science Publishers B.V.
- [Gö31] Kurt Gödel. über vollständigkeit und widerspruchsfreiheit. *Ergebnisse eines mathematischen Kolloquiums (1932)*, 3:12–13, 1931.
- [Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360, 1930. English Version in [vH67].
- [Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.
- [GS89] Jean H. Gallier and Wayne Snyder. Complete sets of transformations for general *E*-unification. *Theoretical Computer Science*, (67):203–260, 1989.
- [Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

- [Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la société des sciences et de lettres de Varsovie, Class III Science mathématique et physique*, 33, 1930.
- [Hil27] David Hilbert. Die Grundlagen der Mathematik. In *Abhandlungen aus dem mathematischen Seminar der Hamburgischen Universität 6*, pages 65–85, 1927.
- [HK95] Dieter Hutter and Michael Kohlhase. A coloured version of the λ -calculus. Technical Report SR 95-05, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, 1995.
- [HKK⁺94a] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Dan Nesmith, Jörn Richts, and Jörg Siekmann. Keim: A toolkit for automated deduction. In Bundy [Bun94], pages 807–810.
- [HKK⁺94b] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts, and Jörg Siekmann. Ω -MKRP a proof development environment. In Bundy [Bun94], pages 788–792.
- [HS86] J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, 1986.
- [Hue72] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic*. PhD thesis, Case Western Reserve University, 1972.
- [Hue75] Gérard P. Huet. An unification algorithm for typed λ -calculus. *Theoretical Computer Science*, 1:27–57, 1975.
- [Hue81] Gérard Huet. A complete proof of correctness of the knuth-bendix-completion-algorithm. *Journal of Computer and System Science*, 23, 1981.
- [Hut90] Dieter Hutter. Guiding induction proofs. In Mark E. Stickel, editor, *Proceedings [Sti90]*, pages 147–161, Kaiserslautern, Germany, July 1990. Springer Verlag.
- [JP73] D. Jensen and T. Pietrzykowski. Mechanizing ω -order type theory through unification. Internal Report CS-73-13, Department of Applied Analysis and Computation, University of Waterloo, 1973.
- [JP76] D. C. Jensen and T. Pietrzykowski. Mechanizing ω -order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.
- [Kap92] D. Kapur, editor. *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, Saratoga Springs, NY, USA, 1992. Springer Verlag.
- [KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
- [KBB93] I. Kraan, D. Basin, and A. Bundy. Middle-out reasoning for program synthesis. In P. Szeredi, editor, *Proceedings of the 10-th International Conference on Logic Programming*. MIT Press, 1993.
- [Kir90] Claude Kirchner, editor. *Unification*. Academic Press, London, 1990.
- [Koh94] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. PhD thesis, Universität des Saarlandes, 1994.
- [Kow75] Robert Kowalski. A proof procedure using connection graphs. *Journal of the Association for Computing Machinery (ACM), ACM, Inc., 1133 Avenue of the Americas, New York 10036*, 22(4):572–595, 1975.
- [LO88] Ewing L. Lusk and Ross A. Overbeek, editors. *Proceedings of the 9th Conference on Automated Deduction*, number 310 in *LNCS*, Argonne, Illinois, USA, 1988.
- [Mil83] Dale Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie-Mellon University, 1983.
- [Mor69] James B. Morris. *E-resolution*. In Donald E. Walker and Lewis Norton, editors, *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 287–294, 1969.
- [Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.
- [RW69] Arthur Robinson and Larry Wos. Paramodulation and TP in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.
- [SG89] Wayne Snyder and Jean Gallier. Higher-Order Unification Revisited: Complete Sets of Transformations. *J. Symbolic Computation*, 8:101–140, 1989.
- [Sie86] J. Siekmann, editor. *Proceedings of the 8th Conference on Automated Deduction*, volume 230 of *LNCS*, Oxford, England, 1986. Springer Verlag.
- [Sko28] T. Skolem. Über die mathematische Logik. *Norsk matematisk tidsskrift 10*, 1928.
- [Sti90] Mark Stickel, editor. *Proceedings of the 10th Conference on Automated Deduction*, number 449 in *LNCS*, Kaiserslautern, Germany, 1990.

- [SW80] Jörg H. Siekmann and Graham Wrightson. Paramodulated connection graphs. *Acta Informatica*, 13:67–86, 1980.
- [vH67] Jean van Heijenoort, editor. *From Frege to Gödel A Source Book in Mathematical Logic, 1879-1931*. Source Books in the History of the Sciences. Harvard University Press, 1967.
- [Wal83] Christoph Walther. A many-sorted calculus based on resolution and paramodulation. In Alan Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 882–891, Los Altos, California, USA, August 1983. William Kaufmann.
- [Zha88] Hantao Zhang. Reduction, superposition, and induction: Automated reasoning in an equational logic. Technical Report 88-06, The University of Iowa, Iowa City, USA, 1988.