Equality and Extensionality in Automated Higher-Order Theorem Proving

Christoph Benzmüller

Dissertation

zur Erlangung des Grades des Doktors der Ingenieurwissenschaften
der Technischen Fakultät der Universität des Saarlandes

Saarbrücken, April 1999

Contents

English Abstract v

Deutsche Kurzzusammenfassung vi

Acknowledgements vii

Deutsche Zusammenfassung viii

1 Introduction 1

1.1 Motivation 1

1.2 Higher-Order Logic 3

1.3 Higher-Order Model Existence 6

1.4 Traditional and Extensional Higher-Order Resolution 10

1.5 Adding Primitive Equality 12

1.6 The LEO System 14

2 Syntax and Semantics of Higher-Order Logic 16

2.1 Syntax and Preliminaries 16

2.2 Pre-Σ-Structures 19

2.3 Σ-Structures 21

2.4 Functional Σ-structures 23

2.5 Σ-Models 25

2.6 Leibniz Equality 27

2.7 Primitive Equality 30

2.8 A Note on Defined and Primitive Equality 32

3 Higher-Order Model Existence 34

3.1 Abstract Consistency 34

3.2 Hintikka Sets 38

3.3 Primitive Equality 41

3.4 Model Existence 44

4 Extensional Higher-Order Resolution: $\mathcal{ER}$ 48

4.1 A Review of $\mathcal{HORES}$ and $\mathcal{ER}$ 48

4.2 Basic Definitions 54

4.3 Lifting Properties 57

4.4 Completeness 60

4.5 Theorem Equivalence 63

5 Extensional Higher-Order Paramodulation: $\mathcal{EP}$ 65

5.1 A Naive and Incomplete Adaptation of Paramodulation 65

5.2 Positive Extensionality Rules 69

5.3 Basic Definitions 70

5.4 Lifting Properties 71

5.5 Completeness 72

5.6 Theorem Equivalence 82

6 Extensional Higher-Order RUE-Resolution: $\mathcal{ERUE}$ 83

6.1 Resolution on Unification Constraints 83

6.2 Basic Definitions 83

6.3 Lifting Properties 85

6.4 Completeness 85

6.5 Theorem Equivalence 91

7 The Leo-System 92

7.1 Basic Data-Structures and Algorithms 93

7.2 Extended SOS Architecture and Main Loop 94

7.3 New Insights gained from Leo 102

7.4 Case Study 103

7.5 Additional Features of Leo 107

8 Examples 111

8.1 Extensionality in $\mathcal{ER}$ 111

8.2 Decomposition in $\mathcal{ER}$ 113

8.3 Leibniz Equality and Alternative Definitions in $\mathcal{ER}$ 115

8.4 Reasoning about Sets with Leibniz Equality 116

8.5 Positive Extensionality Rules in $\mathcal{EP}$ and $\mathcal{ERUE}$ 117

8.6 Comparing $\mathcal{EP}$ and $\mathcal{ERUE}$ 118

8.7 Raised Questions 125

9 Applications and Related Work 127

9.1 Cooperation and Joint Work 127

9.2 Abstract Consistency and Model Existence 127

9.3 Extensional Higher-Order Resolution 129

9.4 Primitive Equality in Higher-Order Theorem Proving 131

9.5 Theorem Provers for Higher-Order Logic 132

9.6 Examples 132

10 Conclusion and Outlook 133

Bibliography 146

Index 147

English Abstract

This thesis focuses on equality and extensionality in automated higher-order theorem proving based on Church's simply typed λ-calculus (classical type theory).

First, a landscape of various semantical notions is presented that is motivated by the different roles equality adopts in them. Each of the semantical notions in this landscape — including Henkin semantics — is then linked with an abstract consistency principle that can be employed for analysing the connection between syntax and semantics of higher-order calculi.

Apart from this proof theoretic tools, the main contributions of this thesis are the three new calculi $\mathcal{ER}$ (extensional higher-order resolution), $\mathcal{EP}$ (extensional higher-order paramodulation) and $\mathcal{ERUE}$ (extensional higher-order RUE-resolution) which improve the mechanisation of defined and primitive equality in classical type theory. In contrast to the refutation approaches for classical type theory developed so far, these calculi reach Henkin completeness without requiring additional extensionality axioms. The key idea is to allow for recursive calls from higher-order unification to the overall refutation search.

Calculus $\mathcal{ER}$, which in contrast to $\mathcal{EP}$ and $\mathcal{ERUE}$, considers equality only as a defined notion, has been implemented in the theorem prover Leo and the suitability of this approach for proving simple theorems about sets has been demonstrated in a case study.

Quo facto, quando orientur controversiae, non magis disputatione opus erit inter duos philosophos, quam inter duos Computistas. Sufficiet enim calamos in manus sumere sedere ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus.

Gottfried Wilhelm Leibniz
(C.I. Gerhardt (ed.), Die philosophischen Schriften von
Gottfried Wilhelm Leibniz, vol. 7, Berlin 1890, p. 200.)

Deutsche Kurzzusammenfassung

Diese Arbeit untersucht Gleichheit und Extensionalität im automatischen Beweisen in Logik höherer Stufe. Die betrachtete Sprache ist die klassische Typtheorie, d.h. eine Logik höherer Stufe basierend auf Church's einfach getypten λ-Kalkül.

Zunächst werden unterschiedlich starke Semantikbegriffe für die klassische Typtheorie erarbeitet, die durch die jeweils unterschiedlichen Rollen, die die Gleichheit in ihnen einnimmt, motiviert sind.

Jedem der eingeführten Semantikbegriffe wird dann eine Menge abstrakter Konsistenzeigenschaften zugeordnet mit dem Ziel, die Analyse der Verbindung zwischen Syntax und Semantik von Beweiskalkülen für die klassische Typtheorie zu erleichtern.

Der Hauptbeitrag dieser Arbeit sind drei neue Beweiskalküle: $\mathcal{ER}$ (extensionale Resolution höherer Stufe), $\mathcal{EP}$ (extensionale Paramodulation höherer Stufe) und $\mathcal{ERUE}$ (extensionale RUE-Resolution höherer Stufe). Ziel dieser Kalküle ist es, die Mechanisierung definierter und primitiver Gleichheit in klassischer Typtheorie zu verbessern. Im Gegensatz zu den zuvor in der Literatur diskutierten Widerlegungsansätzen erreichen diese Kalküle Vollständigkeit hinsichtlich der Henkin Semantik, ohne dem Suchraum zusätzliche Extensionalitätsaxiome hinzufügen zu müssen. Die wesentliche Idee der neuen Kalküle ist es, Unifikation und Beweissuche sehr eng miteinander zu verzahnen: erlaubt sind nun rekursive Aufrufe an die Beweissuche aus dem Unifikationsalgorithmus heraus.

Der Kalkül $\mathcal{ER}$ wurde im Beweissystem Leo implementiert und die Eignung dieses Systems zum Beweisen einfacher Aussagen über Mengen wurde in Fallstudien gezeigt.

Acknowledgements

First of all, I want to thank *Jörg Siekmann* and *Michael Kohlhase*. In his enthusiastic lectures *Jörg Siekmann* raised my interest in Artificial Intelligence and especially in Automated Deduction. Later he offered me a job in his research group, and he generously supported my work during the last years. *Michael Kohlhase*, whose thesis decisively influenced the topic of my work, introduced me to higher-order logic and discussed many details of this thesis with me.

I am furthermore deeply indepted to *Peter Andrews* and *Frank Pfenning*. My research stay at Carnegie Mellon University (CMU) in the first half of 1997, that was made possible by *Peter Andrews*, offered me the chance to widen my knowledge on higher-order logic and to discuss particular ideas on extensional higher-order resolution with him and *Frank Pfenning*. With hindsight, this stay was the break-trough in my PhD study. My special thank to *Frank Pfenning* for becoming the second referee of this thesis.

I also need to mention the many fruitful discussions with my colleagues in the AG Siekmann. Especially since the reunion of the beforehand splitted subgroups in 1997, the AG Siekmann has provided an excellent and highly motivating research environment. As I cannot list all my colleagues here, I representatively thank (in alphabetical order) *Serge Autexier, Lassaad Cheikhrouhou, Detlef Fehrer, Holger Gebhard, Dieter Hutter, Stephan Hess, Karsten Konrad, Heiko Mantel, Andreas Meier, Erica Melis*, and *Volker Sorge*. In phases of hard work it is important to have good friends which offer their support and now and then persuade one to have a drink (in order to prevent one from complete isolation) — in this respect I particularly thank my colleague *Volker* and my office- and study-mate of many years *Serge*.

I am grateful to *Carsten Schürmann* and my CMU office-mates *Matt Bishop* and *Hongwei Xi* for making my stay in Pittsburgh that pleasant.

Manfred Kerber and his family provided a lovely place of refuge for a few weeks in Birmingham, when I tried to escape from my duties in Saarbrücken and write down the final version of this thesis.

My special thanks to *Serge Autexier*, *Kerstin Deckert*, *Christine Heinzius*, *Martin Pollet*, and *Volker Sorge* for reading and correcting parts of the final version.

I owe much to the *Studienstiftung des Deutschen Volkes* for their support during my PhD study. The *Studienstiftung* especially funded my research stay at Carnegie Mellon University without insisting on much paperwork.

Above all, I want to thank my wife *Nicole Clemens* for her understanding and patience. In particular I have to apologise for the many times I promised to come home early but worked at university until late night. I am deeply indepted to my family — especially to my parents *Marlies* and *Nikolaus* — who always supported me, even though they often did not fully agree with my ideas. I dedicate this thesis to my father, who lost his desperate fight against cancer, while I was away at Carnegie Mellon University working on the core of this thesis.

Saarbrücken, April 1999 Christoph Benzmüller

Deutsche Zusammenfassung

Motivation Die Formalisierung und Mechanisierung mathematischen Denkens und Problemlösens ist ein alter Menschheitstraum, nicht zuletzt wegen der Hoffnung diese Art der Formalisierung dann auf andere des Bereiche menschlichen Denkens übertragen zu können. In der griechischen Antike beschrieb vor allem Aristoteles die Schlußregeln des menschlichen Denkens und hinterließ uns eine begrenzte Theorie des logischen Schließens bestehend aus Axiomen und Regeln (z.B. der Modus Ponens Regel). Das Ziel der Formalisierung menschlichen Schließens griff dann Gottfried Wilhelm Leibniz im 18. Jahrhundert wieder auf und ergänzte es um den Gedanken der Mechanisierbarkeit. Seine Idee war die Erschaffung einer *Lingua Charakteristika* und eines *Calculus Ratiocinator*, also einer universellen Sprache zusammen mit einem allgemeinen Beweiskalkül, in der sich alle Probleme und Fragen durch *bloßes Ausrechnen* klären lassen könnten.

Auch die vorliegende Arbeit ist durch diese Idee motiviert. Neben vielen Errungenschaften in der mathematischen Logik mußten aber in diesem Jahrhundert auch herbe Rückschläge hingenommen werden: So zeigen uns Gödels fundamentale Unvollständigkeitsresultate [Göd31] eindeutig die Grenzen der Mechanisierbarkeit mathematischen Schließens auf und die Hoffnung auf einen allumfassenden Kalkül, der in der Lage ist, sämtliche *Wahrheiten* einfach auszurechnen muß deshalb aufgegeben werden.

Auf der anderen Seite haben automatische Beweissysteme, basierend auf Herbrand's fundamentalem Resultat der Widerlegungsvollständigkeit [Her30] in einzelnen Anwendungsdomänen eine beachtliche Leistungsstärke errungen und werden heute erfolgreich in industrienahen Anwendungen, wie der Programm- und Hardwareverifikation, eingesetzt. In der Mathematik selbst konnten nicht-triviale Sätze erstmals mit dem Computer bewiesen werden: Der Gruppe um Bill McCune am Argonne National Laboratory, Illinois, USA, gelang es 1997 das seit 60 Jahre unbewiesene Robbins Problem mit dem automatischen Beweiser EQP zu lösen [McC97b].

Aufgrund der enormen Komplexität automatischen Beweisens erscheint es aber als wenig realistisch, daß automatische Beweissysteme bereits in wenigen Jahrzehnten eine den heutigen Schachcomputern[1] vergleichbare Leistungsfähigkeit im Vergleich zum Menschen erreichen werden. Durch *Brute-Force* Suche in Kombination mit schneller Hardware, wie in den meisten Schachcomputern, wird ein Erfolg im automatischen Beweisen aber nicht möglich sein. Vielmehr wird die Integration und Kombination leistungsfähiger traditioneller Deduktionssysteme für Logik erster Stufe (z.B. OTTER [MW97, McC94], SPASS [WGR96], SETHEO [GLMS94], PROTEIN [BF94], BLIKSEM, WALDMEISTER [HBVL97]) und höherer Stufe (z.B. TPS [ABI+96, AINP90], HOL [GM93], PVS [ORS92]) mit Computeralgebrasystemen

[1] Der Schachcomputer Deep Blue von IBM hat im Mai 1997 mit 3.5 zu 2.5 Punkten den Weltmeister Gary Kasparov geschlagen.

(z.B. MAPLE [Red96], MATHEMATICA [Wol88, Kau92]) und Constraint-Lösern in einer abstrakteren und kognitiv adäquateren Beweisplanungsebene (siehe z.B. [CrS98, KKS98, Mel95, Bun88, WNB92, AH97]) wohl eher erforderlich sein. Dieser Ansatz erlaubt es, zunächst grobe Beweispläne aufbauend auf einem Satz theoriespezifischer Beweismethoden zu entwickeln, um diese dann anschließend zu detaillierten Beweisen auf Kalkülebene zu verfeinern. Die integrierten Subsysteme können die Beweisplanung unterstützen, indem sie Unterbeweise führen, Nebenrechnungen ausführen oder die Erfüllbarkeit von Constraints überprüfen. Auch bei der Verfeinerung abstrakter Beweispläne können die eingebauten Subsysteme zur Lösung wichtiger Unterprobleme eingesetzt werden.

Die Weiterentwicklung und Perfektionierung des traditionellen automatischen Beweisen, insbesondere im weitgehend vernachlässigten Bereich der Logik höherer Stufe, ist demnach ein wichtiges Unterziel bei der Entwicklung eines leistungsfähigen Assistenzsystems für die Mathematik, das auch nicht-triviale Probleme autonom lösen kann. Deshalb konzentriert sich diese Doktorarbeit auf die noch relativ wenig untersuchte Mechanisierbarkeit von Logiken höherer Stufe und fokussiert dabei auf die bereits in der Logik erster Stufe schwer handhabbare aber enorm wichtige Gleichheitsrelation. Etwas präziser formuliert, untersucht die vorliegende Arbeit die Rolle und Mechanisierbarkeit der Gleichheit in klassischer Logik höherer Stufe basierend auf Church's einfach getypten λ-Kalkül [Chu40] — dieser Ansatz zur Formalisierung der Mathematik wird traditionell auch als (klassische) Typtheorie bezeichnet (siehe [And86]) — und schlägt dazu drei Kalküle vor, die die Mechanisierung der Gleichheit verbessern sollen.

In der Untersuchung der theoretischen Eigenschaften dieser Kalküle wird, falls nicht anders erwähnt, eine *Standardsemantik*, beziehungsweise bei der Vollständigkeitsanalyse der entwickelten Beweiskalküle die *Henkin-Semantik* [Hen50, Hen96] vorausgesetzt. Wesentliches Charakteristikum der Standardsemantik ist, daß jedem funktionalen Typ $\alpha \rightarrow \beta$, der in der getypten Sprache betrachtet wird, das jeweils volle Funktionsuniversum gebildet über den zugrundeliegenden Universen für die Typen α und β in der Semantikkonstruktion zugeordnet wird. Die Wahl der Universen zu den gewählten Basistypen (üblicherweise ι und o, wobei als das Universum für o die zweielementige Menge der Wahrheitswerte gewählt wird) legt demnach auch bereits alle Funktionsuniversen für die funktionalen Typen in dieser Sprache fest. Der Übergang zur Henkin Semantik bei der Analyse von Kalkülen ist durch die bereits erwähnten Gödelschen Unvollständigkeitsresultate [Göd31] motiviert, die aufzeigen, daß in der Typtheorie mit Standardsemantik keine vollständigen Kalküle möglich sind. Leon Henkin hat den auf vollen Funktionsuniversen aufbauenden Begriff der Standardsemantik in [Hen50, Hen96] verallgemeinert, indem er auch die Betrachtung nicht voller Funktionsuniversen zuläßt; allerdings unter Einhaltung der *Denotatpflicht*, die sicherstellt, daß die gewählten Universen reichhaltig genug sind, um jedem Ausdruck der Sprache auch ein entsprechendes Denotat zuzuordnen. Die Henkin'sche Verallgemeinerung hat zur Folge, daß sich die Menge der allgemeingültigen Formeln verringert, weil mehr Gegenmodelle konstruiert werden können, und diese Generalisierung geht genau soweit, daß vollständige Kalküle möglich werden.

Die in dieser Arbeit vorgeschlagenen Kalküle bauen letztlich auf den Pionierarbeiten zur Resolution höherer Stufe [And71] von Peter Andrews und zur Constraint Resolution höherer Stufe [Hue72] von Gérard Huet auf. Direkt greifen sie auf das unsortierte Fragment des sortierten Resolutionskalkül [Koh94b] neueren Datums von Michael Kohlhase zurück. Das Ziel besteht darin, Henkin-Vollständigkeit zu erreichen, ohne aber dabei auf zusätzliche Gleichheits- oder sonstige Axiome angewiesen zu sein und somit die Mechanisierung der Gleichheitsrelation im Kalkül selbst zu verbessern. Wie auch in den traditionellen Ansätzen wird die Gleichheit dabei zunächst als ein nach dem Leibniz'schen Prinzip (zwei Dinge sind gleich falls sie die gleichen Eigenschaften aufweisen) definiertes Konzept angesehen, das bekanntermaßen leicht in Logik höherer Stufe formalisiert werden kann ($=^{\alpha} := (\lambda X_{\alpha} Y_{\alpha}.\, \forall P_{\alpha\to o}.\, P\, X \Rightarrow P\, Y)$). Leider müssen aber in den obigen Ansätzen sowie in allen anderen dem Autor bekannten Widerlegungsansätzen für klassische Typtheorie die Extensionalitätsaxiome der Leibnizgleichheit den jeweiligen Kalkülen hinzugefügt werden, um Henkin-Vollständigkeit zu erreichen. Das dieses Problem nicht rein theoretischer Natur ist, sondern eine echte Bedeutung in der Praxis erlangt, zeigen die in dieser Arbeit diskutierten Beispiele. Neben der Entwicklung einer geschickten Extensionalitätsbehandlung versucht diese Arbeit aber auch die Mechanisierung der primitiven Gleichheitsrelation voranzutreiben und erarbeitet dazu einen paramodulationsbasierten sowie einen differenzreduzierenden Kalkül für die klassische Typtheorie.

Für die Vollständigkeitsanalysen der Kalküle war es erforderlich, die Beweismethode der abstrakten Konsistenz zu erweitern, die ursprünglich von Smullyan [Smu63] für eine Logik erster Stufe entwickelt und dann von Andrews [And71] für Logik höherer Stufe aufbauend auf Ideen von Takahashi [Tak67] erweitert wurde. Die Andrews'sche Entwicklung basiert auf einem sehr allgemeinen Semantikbegriff (der v-Komplexe), so daß eine weitere Adaptation der Beweismethode der abstrakten Konsistenz im Rahmen dieser Arbeit (gemeinsam mit Michael Kohlhase) erforderlich war, um Vollständigkeitsbeweise auch für die Henkin-Semantik zu ermöglichen.

Die theoretischen Beiträge dieser Doktorarbeit werden ergänzt und untermauert durch die Implementierung der Kalküle im Beweiser LEO und die durchgeführten Fallstudien.

Stichwortartig lassen sich die Ziele (und Beiträge) der Arbeit damit wie folgt zusammenfassen:

- Analyse und Klärung der Rolle der Gleichheit im automatischen Beweisen in klassischer Typtheorie.
- Adaptation der Beweismethode der abstrakten Konsistenz für die Henkin-Semantik.
- Entwicklung eines Henkin-vollständigen Resolutionsansatzes, der zusätzliche Extensionalitätsaxiome im Suchraum vermeidet.
- Entwicklung Henkin-vollständiger primitiver Gleichheitsverfahren, die ebenfalls ohne zusätzliche Axiome auskommen.
- Analyse der Praktikabilität dieser Ansätze im Rahmen einer Implementierung und zusätzlicher Fallstudien.

Die skizzierten Beiträge dieser Arbeit werden nun in chronologisch motivierter Reihenfolge noch etwas näher beleuchtet. Dies bildet auch die Einleitung in Kapitel 1 (allerdings in strukturell statt chronologisch motivierter Reihenfolge).

Probleme mit den Extensionalitätsprinzipien In der Anfangsphase dieser Doktorarbeit stellte sich heraus, daß sowohl der als Ausgangspunkt gewählte sortierte Resolutionskalkül höherer Stufe [Koh94b] als auch der um eine Regel zur Behandlung der Extensionalität auf Wahrheitswerten erweiterte, aber ansonsten analoge Tableauxansatz [Koh95] genauso wie die traditionellen Ansätze [And71, Hue72, JP72] weder das funktionale Extensionalitätsprinzip noch das Extensionalitätsprinzip auf Wahrheitswerten in voller Generalität berücksichtigen. Das funktionale Extensionalitätsprinzip besagt, daß zwei Funktionen genau dann gleich sind, wenn sie sich hinsichtlich aller Argumente gleich verhalten, d.h. punktweise gleich sind

$(\forall F_{\alpha\to\beta}, G_{\alpha\to\beta}.\ (F =^{\beta} G) \equiv (\forall X_{\alpha}.(F\ X =^{\beta} G\ X)))$, während das Extensionalitätsprinzip auf Wahrheitswerten ausdrückt, daß Gleichheitsrelation und Äquivalenzrelation auf Wahrheitswerten übereinstimmen $(\forall A_{o}, B_{o}.\ (A =^{o} B) \equiv (A \equiv B))$. Wir werden nun die Schwäche der beiden Ausgangskalküle und deren Auswirkungen auf dieses Forschungsvorhaben beleuchten. Beide Kalküle behandeln die Gleichheit, ebenso wie die vorangegangenen traditionellen Ansätze, nicht als eine primitive Relation, sondern als ein nach dem Leibniz'schen Prinzip definiertes Konzept. Dieses Prinzip formuliert, daß zwei Dinge gleich sind, wenn sie hinsichtlich ihrer Eigenschaften nicht unterschieden werden können $(=^{\alpha} := (\lambda X_{\alpha} Y_{\alpha}.\, \forall P_{\alpha\to o}.\ P\,X \Rightarrow P\,Y))$. Allerdings beschreibt das Leibniz'sche Prinzip nur dann exakt die intendierte Gleichheitsrelation, falls man von einer genügend starken Semantikkonstruktion – z.B. Henkin-Semantik – für Logik höhere Stufe ausgeht. Eine detaillierte Diskussion dieses Aspektes wird in Kapitel 2.1 präsentiert (siehe aber auch [BK97a]). Setzt man aber beispielsweise Henkin-Semantik voraus, so kann man jede primitive Gleichung in einem Beweisproblem der Logik höheren Stufe durch eine äquivalente Formulierung basierend auf der Leibnizdefinition, also ohne primitives Gleichheitssymbol, ersetzen. Auf dieser Idee bauen die traditionellen Ansätze [And71, Hue72, JP72], sowie die Aufgangskalküle [Koh94b] und [Koh95] auf, und sehen deshalb auch keine primitive Behandlung der Gleichheit im Kalkül vor. Weil aber die Leibnizgleichheit den Suchraum (durch das Einführen flexibler Literalköpfe, d.h. freier Prädikatsvariablen) erheblich aufbläht und im allgemeinen leider unendlich viele funktionale Extensionalitätsaxiome (pro funktionalem Typ der Sprache je ein Axiom) notwendig sind, ist es ein wesentliches Ziel dieser Arbeit, eine primitive Gleichheitsbehandlung auszuarbeiten, die diese Nachteile vermeidet.

Ein wichtiger Aspekt klassischer Typtheorie ist nun aber, daß — wiederum eine genügend starke Semantik vorausgesetzt — die Gleichheit stets als definiertes Konzept behandelt werden kann; sie ist mehr oder weniger a priori in die Logik eingebaut. Selbst wenn man sich grundsätzlich für eine primitive Behandlung der Gleichheit entscheidet und die Signatur der zugrundeliegenden Logik um primitive Gleichheitssymbole (für jeden Typ) erweitert, so besteht immer noch die Möglichkeit, die Gleichheit nach obiger Idee als definiertes Konzept zu betrachten. Wie in Abschnitt 2.8 erläutert wird (siehe auch [Ben98, Ben99]), gibt es neben der obigen Leibniz'schen Formulierung noch unendlich viele alternative Möglichkeiten, die Gleichheit als definiertes Konzept zu betrachten. Insgesamt folgt daraus, daß es unentscheidbar ist, ob eine bestimmte Teilformel einer aufgelösten Gleichung entspricht, mit der Konsequenz, daß rein theoretisch eine rein primitive Behandlung der Gleichheit in einem Kalkül niemals ausreichen kann, um Vollständigkeit (bezüglich Henkin-Semantik) zu erreichen. Vielmehr muß ein Henkin-vollständiger Kalkül in jedem Fall – also unabhängig davon, ob er nun die Gleichheit zusätzlich auch als ein primitives Konzept betrachtet – eine genügend mächtige Gleichheitsbehandlung auch für definierte Gleichungen bereitstellen.

So ist beispielsweise die Gültigkeit der Gleichheitsaxiome (Kongruenzaxiome und zusätzliche Axiome für die funktionale Extensionalität, sowie die Extensionalität auf Wahrheitswerten) ein wichtiges Testkriterium für die Henkin-Vollständigkeit eines Kalküls, selbst oder insbesondere bei Ersetzung der Gleichheitssymbole in diesen Axiomen durch Definitionen im Leibniz'schen Sinne.

Hier versagen nun die in der Literatur diskutierten Widerlegungsansätze und leider zum Teil auch die Ausgangskalküle [Koh94b] und [Koh95]. So müssen die in den Pionierarbeiten zur Resolution in höherer Stufe [And71, Hue72, JP72] diskutierten Ansätze die Extensionalitätsaxiome explizit dem Suchraum hinzufügen um Henkin-Vollständigkeit zu garantieren. Und obwohl [Koh94b] noch davon ausging, daß der dort beschriebene Resolutionskalkül zumindest das funktionalen Extensionalitätsprinzip implizit im Kalkül berücksichtigt und somit auf dessen Axiomatisierung verzichten kann, stellte sich schnell heraus, daß eine auf der Grundlage der Leibnizdefinition formalisierte Variante des funktionalen Extensionalitätsprinzips doch nicht mit diesem Kalkül gezeigt werden kann. Das heißt, daß der Resolutionskalkül in [Koh94b] doch nicht die funktionale Extensionalität implizit berücksichtigt, sondern es auch in diesem Kalkül notwendig ist, sämtliche Extensionalitätsaxiome dem Suchraum hinzuzufügen. Der auf der Grundlage des Resolutionskalkül [Koh94b] weiterentwickelte Tableauxkalkül für höhere Ordnung [Koh95] stellt eine zusätzliche Regel bereit, die das Problem der Extensionalität auf Wahrheitswertswerten attackiert. Unter der Annahme, daß [Koh94b] bereits das funktionale Extensionalitätsprinzip generell berücksichtigt, geht [Koh95] deshalb davon aus, erstmals einen Henkin-vollständigen Widerlegungsansatz gefunden zu haben, der ohne zusätzliche Axiome auskommt.

Das Zusammenspiel beider Prinzipien bildet aber gerade die entscheidende Herausforderung zum Erreichen der Henkin-Vollständigkeit und aufgrund der Schwäche von [Koh94b] versagt hier folglich auch der in [Koh95] vorgestellte erweiterte Tableauxkalkül.

Anhand des folgenden Beispiels soll die Schwäche sämtlicher vor dieser Arbeit verfügbaren Widerlegungskalküle für die klassische Typtheorie erläutert werden und die Bedeutung für die Mechanisierung einfacher Probleme aus der Mathematik, z.B. triviale Aussagen über Mengen, herausgestellt werden:

$$(\lambda X_\iota .\ \mathrm{rot}_{\iota\to o}\ X \land \mathrm{kreis}_{\iota\to o}\ X) = (\lambda X_\iota .\ \mathrm{kreis}_{\iota\to o}\ X \land \mathrm{rot}_{\iota\to o}\ X)$$

Die linke Seite dieser Gleichung beschreibt die Menge der roten Kreise ($\{x|\mathrm{rot}\ X \land \mathrm{kreis}\ X\}$) und die rechte Seite die Menge der Kreise, die rot sind ($\{x|\mathrm{kreis}\ X \land \mathrm{rot}\ X\}$). Obwohl die Gleichheit dieser beiden Mengen dem Leser trivial erscheint, so ist dieser Satz für automatische Beweiser in der Typtheorie eine große Herausforderung (noch weitaus größer ist aber die Herausforderung an einen Beweiser erster Stufe bei einer entsprechenden Kodierung dieses Problems z.B. in Tarski Grothendieck Mengentheorie [Try89] oder Zermelo-Fränkel Mengentheorie [Zer08, Frä22b, Frä28]). Die Anwendung beider Extensionalitätsprinzipien der Leibnizgleichheit ist nämlich zunächst erforderlich, um diese Gleichung in folgende allquantifizierte Äquivalenz zu transformieren:

$$\forall Y_\iota .\ (\mathrm{rot}\ Y \land \mathrm{kreis}\ Y) \equiv (\mathrm{kreis}\ Y \land \mathrm{rot}\ Y)$$

Die transformierte Aussage enthält keine weiteren Anteile höherer Stufe und ist in wenigen Schritten beweisbar.

Obiges Beispiel unterstreicht die Bedeutung der beiden Extensionalitätsprinzipien selbst für die Mechanisierung trivialer Probleme in der Mathematik oder auch der Sprachverarbeitung (konkrete Beispiele werden in [KK98b] diskutiert), da in beiden Domänen Vergleiche von Mengen oder Aussagen eine zentrale Rolle spielen.

Kurz zusammengefaßt stellte sich für diese Doktorarbeit in der Anfangsphase also folgendes Problem: Sämtliche verfügbaren Widerlegungsansätze für die klassische Typtheorie stellen keine zufriedenstellende Handhabung der Extensionalitätsprinzipien der Gleichheit bereit, sondern setzen die Hinzunahme unendlich vieler Extensionalitätsaxiome in den Suchraum voraus.

Es war also zunächst von hoher Priorität einen Henkin vollständigen Ansatz für definierte Gleichheit zu entwickeln, der die Extensionalitätsaxiome im Suchraum vermeidet, um dann darauf aufbauend die Entwicklung geeigneter Ansätze zur primitiven Gleichheitsbehandlung anzustreben.

Extensionale Resolution höherer Stufe Im technischen Bericht [Ben97] wird erstmals ein Kalkül für *Extensionale Resolution höherer Stufe* ($\mathcal{ER}$) diskutiert, der die Behandlung beider Extensionsionalitätsprinzipien in geeigneter Weise miteinander kombiniert. Die Vollständigkeit des Kalküls $\mathcal{ER}$ wurde im Anschluß in [BK98a] gezeigt. Der im Rahmen dieser Doktorarbeit entwickelte Kalkül $\mathcal{ER}$ ist demnach der erste Kalkül für klassische Typtheorie, der ohne zusätzliche Axiome Henkin-Vollständigkeit garantiert.

Wegen der Unentscheidbarkeit der Unifikation für Logiken höherer Stufe besteht die Anforderung an einen Resolutionskalkül, die Suche nach Unifikatoren auf einer gemeinsamen Ebene mit der generellen Beweissuche zu verschmelzen, bzw. Unifikationsprobleme als grundsätzlich unentscheidbare Constraints zu betrachten. Der Kalkül $\mathcal{ER}$ geht noch einen Schritt weiter: Er erlaubt rekursive Aufrufe an den gesamten Beweisprozeß aus der Unifikation heraus, das heißt, er verschmilzt Unifikation und Resolution noch enger. Der Kalkül $\mathcal{ER}$ wird in Kapitel 4 präsentiert und die Unterschiede zu [Koh94b] und [Koh95] werden dort detailliert diskutiert. $\mathcal{ER}$ ist zudem in [BK98a] und [Ben97] beschrieben.

An dieser Stelle soll die zentrale Idee des Kalküls $\mathcal{ER}$ anhand unseres obigen Beispiels kurz motiviert werden: Ein herkömmlicher Resolutionsansatz, z.B. [Hue72, Koh94b], dem das Negat unserer Ausgangsformel eingegeben wird, erlaubt es sehr rasch die leere Klausel abzuleiten, allerdings unter dem Unifikationsconstraint, daß die beiden Seiten der Gleichung aneinander angeglichen werden können[2]:

$$[p_{(\iota\to o)\to o}\ (\lambda X_\iota.\ (\text{rot}\ X \land \text{kreis}\ X) = p_{(\iota\to o)\to o}\ (\lambda X_\iota.\ \text{kreis}\ X \land \text{rot}\ X)]^F$$

Ganz konkret ist dieses Unifikationsconstraint ableitbar durch Resolution zwischen den beiden einzigen Einerklauseln, die aus der Klauselnormalisierung des vollständig definitionsexpandierten negierten Eingabeproblems resultieren. Die Konstante $p_{o\to o}$ ist dabei entstanden durch die Normalisierung des negierten Allquantors. Bereits nach wenigen Unifikationsschritten (Dekomposition von p und Elimination der λ-Binder — was der Anwendung des funktionalen Extensionalitätsprinzips entspricht und den Skolemterm s einführt) scheitert diese dann an folgendem Unifikationsconstraint

$$[\text{rot}\ s \land \text{kreis}\ s = \text{kreis}\ s \land \text{rot}\ s]^F$$

Der Grund des Scheiterns ist, daß die Dekomposition lediglich syntaktisch vorgeht und nicht die Kommutativität von $\land$ berücksichtigen kann. Der Kalkül $\mathcal{ER}$ löst dieses Problem, indem er an dieser Stelle das Extensionalitätsprinzip auf Wahrheitswerten anwendet, dadurch obige negierte Gleichung, d.h. unser Unifikationsconstraint, durch eine negierte Äquivalenz ersetzt, die dann erneut normalisiert wird. Dadurch wird letztlich ein rekursiver Aufruf des Beweisers aus der Unifikation realisiert, der es erlaubt, die Gleichheit der beiden (modifizierten) Terme innerhalb des gesamten Kalküls — und nicht nur durch rein syntaktische Unifikation — nachzuweisen.[3]

Es ist offensichtlich, daß dem Kalkül $\mathcal{ER}$ die Idee der Differenzreduzierung zugrunde liegt: Einen Resolutionsschritt begründende Gleichungen werden zunächst als Unifikationscons-

[2] Unifikationsconstraints werden in [BK98a] als negierte Gleichungsliterale kodiert. Diese Literale formulieren also die Bedingungen unter denen eine Klausel gilt — in $\mathcal{ER}$ ist die Kodierung als negierte Gleichungsliterale nicht zwingend erforderlich.

[3] Die Unifikation höherer Stufe ist natürlich keine rein syntaktische Unifikation, weil sie, durch die λ-Konversionsregeln unterstützt, das funktionale Extensionalitätsprinzip bereits zu einem gewissen Grade berücksichtigt. Aber eben leider nur zu einem gewissen Grade und insbesondere ermöglicht sie nicht die notwendige Kombination von funktionalen und booleschem Extensionalitätsprinzip, wie unser Beispiel zeigt.

traints kodiert und können durch die Unifikationsregeln zu einem späteren Zeitpunkt entweder gelöst oder so weit wie möglich modifiziert und vereinfacht werden. Die übrig bleibenden Differenzen der modifizierten Unifikationsprobleme können dann durch rekursive Aufrufe an den gesamten Widerlegungsprozeß und nicht nur innerhalb der rein syntaktischen Unifikation analysiert werden.

Alternative Vollständigkeitsbeweise Die Vollständigkeit des Kalküls $\mathcal{ER}$ wurde bereits analysiert in [BK98a]. In dieser Arbeit wird nun ein im Vergleich zu [BK98a] alternatives Vorgehen zur Analyse der Henkin-Vollständigkeit für $\mathcal{ER}$ vorgestellt, das aber leider die zusätzliche, Instantiierungen ratende *FlexFlex*-Unifikationsregel (siehe Abbildung 4.2 oder auch [SG89, Sny91]) im Kalkül $\mathcal{ER}$ voraussetzt. Die Vermutung (siehe Conjecture 4.22), daß diese Regel vermieden werden kann, konnte im Rahmen dieser Arbeit leider nicht mehr bewiesen werden. Motiviert ist das neue Vorgehen und der alternative Beweis wie folgt:

- Das sehr zentrale Liftinglemma in [BK98a] stützt sich letztlich auf das Liftinglemma in [Koh94b] und verwendet den dort eingeführten, recht komplizierten und wenig intuitiven Begriff der Klauselisomorphismen. Ein Ziel dieser Arbeit ist es, diesen komplizierten Beweis des Liftinglemmas zu vereinfachen, dabei auf das (fehleranfällige, weil komplizierte) Konzept der Klauselisomorphismen zu verzichten und sich enger an traditionelle, d.h. aus der ersten Stufe bekannte, Argumentationsweisen anzulehnen. Dies ist möglich, wie in Kapitel 4 gezeigt wird, wenn man den Kalkül $\mathcal{ER}$ zunächst um die *FlexFlex* Regel erweitert und die Liftingeigenschaft im erweiterten Kalkül zeigt (eine weitere kleine Modifikation betrifft die Integration der einzelnen Klauselnormalisierungsregeln in den Kalkül). Konsequenterweise gilt es im Anschluß, die Theoremäquivalenz zwischen dem erweiterten und dem reinen Kalkül $\mathcal{ER}$ zu zeigen, um dadurch auf die Vollständigkeit von $\mathcal{ER}$ schließen zu können.

- Ein zweiter Beweis der Vollständigkeit von $\mathcal{ER}$ vermindert das Fehlerpotential dieser Aussage insgesamt und bietet auch alternative Einblicke in die theoretischen Eigenschaften des Kalküls.

- Der Unterschied zu Huet's traditionellem Ansatz der Constraint Resolution höherer Stufe wird durch die modifizierte Beweisführung besser illustriert: Huet's Ansatz geht davon aus, daß die (Prä-)Unifikation höherer Stufe komplett verzögert werden kann, indem in jedem Resolutions- und Faktorisierungsschritt das zu lösende Unifikationsproblem zunächst explizit als Constraint in der Klausel selbst kodiert wird. Sobald die leere Klausel gefunden ist, gilt es dann die aufgesammelten Constraints zu lösen um die gefundene Ableitung letztendlich zu rechtfertigen. Im Kalkül $\mathcal{ER}$ ist eine grundsätzliche Verzögerung aber nicht mehr möglich. Wie unser obiges Beispiel (sowie viele der Beispiele in Kapitel 8 und selbst der zentrale Vollständigkeitsbeweis 4.16) demonstriert, sind vorzeitige, rekursive Aufrufe des gesamten Widerlegungsansatzes aus der Unifikation heraus zwingend erforderlich, um die Vollständigkeit zu garantieren. Unifikation und Resolution werden in der extensionalen Resolution höherer Stufe eben noch enger als in Huet's Ansatz miteinander verwoben.

- Die wichtigste Motivation ist aber, ein analoges Vorgehen bei der Vollständigkeitsanalyse für alle drei in dieser Arbeit erarbeiteten Kalküle ($\mathcal{ER}$, $\mathcal{EP}$ und $\mathcal{ERUE}$) zu ermöglichen. Weil der Versuch einer direkten Beweisführung aufbauend auf den Ideen von [BK98a] und [Koh94b] mißlang, wurde für diese Arbeit das oben skizzierte, modifizierte Vorgehen gewählt.

Beweisprinzip der abstrakten Konsistenz Der Vollständigkeitsbeweis des Kalküls $\mathcal{ER}$ setzt die Erweiterung der für die Logik erster Stufe durch Smullyan [Smu63] eingeführten Beweismethode der abstrakten Konsistenz voraus. Zwar wurde dieses Beweisprinzip 1971 von Peter Andrews [And71] für die höhere Stufe erweitert, allerdings nur für die sehr schwache Semantik der v-Komplexe. In Zusammenarbeit mit Michael Kohlhase und aufbauend auf den in [Koh93] diskutierten Vorarbeiten, ist es im Rahmen dieser Arbeit gelungen, eine entsprechende Erweiterung der abstrakten Konsistenzmethode für Henkin-Semantik auszuarbeiten und deren Korrektheit zu beweisen. Vorgestellt wird das erweiterte Beweisprinzip in den Kapiteln 2.1 und 3 (siehe aber auch [BK97a]). In Kapitel [BK98a] wird es dann angewendet, um die Henkin Vollständigkeit des Kalküls $\mathcal{ER}$ zu zeigen.

Die theoretischen Betrachtungen in den Kapitel 2.1 und 3 zeigen aber noch mehr: nämlich eine ganze Reihe interessanter Semantik-Konstruktionen für die Logik höherer Stufe, die im Wesentlichen durch unterschiedliche Eigenschaften des Denotats der Leibnizgleichheit motiviert sind. Jedem der ausgearbeiteten Semantikbegriffe wird dann eine entsprechende Menge abstrakter Konsistenzeigenschaften zugeordnet.

Ausgangspunkt der Semantikkonstruktionen sind Σ-Modelle, die im Wesentlichen totalen v-Komplexen entsprechen. Das heißt, daß im Gegensatz zu Andrews' v-Komplexen — die mit partiellen Valuationen arbeiten — in Σ-Modellen jeder Formel ein Denotat zugewiesen wird. Die Funktionsuniversen in Σ-Modellen sind nicht notwendigerweise voll. Ferner sind Σ-Modelle weder funktional[4], noch ist das Denotat der Leibnizgleichheit notwendigerweise die Gleichheitsrelation. Letzteres gilt auch noch für die Klasse der funktionalen Σ-Modelle, in denen η-Gleichheit explizit gefordert wird. Wie Andrews in [And72a] zeigt, gilt erst dann, daß die Leibnizgleichheit tatsächlich die Gleichheitsrelation beschreibt, wenn man sicherstellen kann, daß die Gleichheitsrelationen q^{α} (bzw. einelementige Mengen $\{a\}$ für alle a in $\mathcal{D}_a$) Elemente der Domänen $\mathcal{D}_{\alpha\to\alpha\to o}$ (bzw. $\mathcal{D}_{\alpha\to o}$) sind. Es ist leicht zu überprüfen, daß diese Forderung automatisch erfüllt ist, sobald die konkret betrachtete Logik ein primitives Gleichheitssymbol in der Logik selbst vorsieht.

Ein weiteres wichtiges Charakteristikum für Semantikkonstruktionen in Logik höherer Stufe betrifft die Kardinalität der Menge der Wahrheitswerte $\mathcal{D}_o$. In (funktionalen) Σ-Modellen ist lediglich gefordert, daß eine Abbildung von der konkret gewählten Menge $\mathcal{D}_o$ in die Menge der Wahrheitswerte $\{\mathtt{T},\mathtt{F}\}$ angegeben werden kann, so daß sich die Denotate der logischen Junktoren entsprechend ihrer intuitiven Semantik modulo dieser Abbildung verhalten. Diese Forderung kann man natürlich dahingehend verstärken, daß eine zweiwertige Menge $\{\mathtt{T},\mathtt{F}\}$ von Wahrheitswerten explizit vorgeschrieben wird.

Henkin Modelle kombinieren schließlich diese beiden Forderungen: Die Leibnizgleichheit muß tatsächlich die Gleichheitsrelation charakterisieren und die Menge der Wahrheitswerte muß isomorph sein zu $\{\mathtt{T},\mathtt{F}\}$. Standardsemantik ergibt sich dann aus der Henkin-Semantik durch die zusätzliche Forderung nach vollen Funktionsuniversen. Der wissenschaftliche Beitrag dieser Analyse ist, daß einerseits die oben skizzierten Semantikkonstruktionen formal erarbeitet und analysiert und andererseits diesen Konstruktionen dann jeweils eine Menge abstrakter Konsistenzeigenschaften zugeordnet werden.

[4] Das heißt, das zum Beispiel η-Gleichheit nicht gewährleistet ist.

Dadurch ist es nun möglich, die Verbindung zwischen Syntax und Semantik für unterschiedliche Kalküle der Typtheorie formal in einem einheitlichen Ansatz zu analysieren und diese dann in ihrer Mächtigkeit zu differenzieren. Insbesondere bietet diese Arbeit erstmals ein entsprechendes Beweisprinzip für die in mathematischer Anwendungsdomäne wichtige Henkin-Vollständigkeit.

Primitive Gleichheit: Differenzreduzierender und Termersetzender Ansatz Ein weiteres Ziel dieser Arbeit ist die Erweiterung der Constraint Resolution in [Koh94b] um eine primitive Gleichheitsbehandlung. In Anlehnung an bekannte Verfahren in der Logik erster Stufe soll hierzu ein differenzreduzierender sowie ein termersetzender Ansatz erarbeitet und dann miteinander verglichen werden. Während sich in Logik erster Stufe termersetzende Verfahren (z.B. der Superpositionskalkül [BGLS92]) gegenüber differenzreduzierenden in vielen Anwendungsdomänen — sieht man einmal vom Induktionsbeweisen ab — durchgesetzt zu haben scheinen, ist das Rennen in der Logik höherer Stufe noch völlig offen und wird möglicherweise auch einen anderen Ausgang nehmen.

Sicherlich wurden in den letzten Jahren Fortschritte hinsichtlich des schwierigen Problems der Konstruktion von Reduktionsordnungen für Terme höherer Stufe erreicht, siehe z.B. [JR99, LB98, JR98], die eine Adaption termersetzender Verfahren auf Logiken höherer Stufe nahelegen. Auf der anderen Seite zeigen jedoch die Untersuchungen zur Integration primitiver Gleichheit in den Kalkül $\mathcal{ER}$, daß sich im Kontext des automatischen Beweisens für die Henkin-Semantik im Vergleich zur ersten Stufe neue Probleme ergeben. Wieder sind es die Extensionalitätsprinzipien, die eine Herausforderung an das Beweisverfahren stellen. Konkret stellt sich das Problem, daß man nun grundsätzlich Einerklauseln haben kann, die aus einem einzigen positiven oder negativen Gleichungsliteral bestehen und die bereits widersprüchlich sind. Von der ersten Stufe ist bekannt, daß man in einem Paramodulationsansatz ganz besonders auf die Widerlegbarkeit negativer Gleichungsliterale, wie z.B. $[(f\ X) = (f\ a)]^F$, achten muß. In [RW69] wird dieses Problem gelöst durch die spezielle Reflexivitätsregel, die es erlaubt ein negatives Gleichungsliteral zu eliminieren, sobald beide Seiten der Gleichung unifizierbar sind. In der Logik höherer Stufe gelingt es aufgrund der Extensionalitätsprinzipien nicht mit der rein syntaktischen Unifikation das Problem zu lösen. Ausserdem sind nun auch positive Gleichungen betroffen. Die folgenden vier Einerklauseln sollen dies demonstrieren: $[a = \neg\neg a]^F, [a = \neg a]^T, [\lambda X_\iota.\ \mathrm{rot}\ X = \lambda X_\iota.\ \neg\neg\mathrm{rot}\ X]^F, [\lambda X_\iota.\ \mathrm{rot}\ X = \lambda X_\iota.\ \neg\mathrm{rot}\ X]^T$. Jede dieser Klauseln ist widersprüchlich, doch der Widerspruch kann durch syntaktische Unifikation nicht gezeigt werden. Insbesondere für die beiden letzten Klauseln ist wieder ein geschicktes Zusammenspiel beider Extensionalitätsprinzipien erforderlich, um den Widerspruch formal aufzeigen zu können. Diese Beispiele motivieren also die Notwendigkeit spezieller Extensionalitätsregeln, wie sie im Kalkül $\mathcal{ER}$ bereits für die Unifikationsconstraints eingeführt wurden, um das Problem der Extensionalität zu überwinden. Auch hier sind nun rekursive Aufrufe an den gesamten Widerlegungsprozeß erforderlich, um generell alle widersprüchlichen positiven oder negativen Gleichungsliterale berechnen zu können. Wie aber bereits diskutiert wurde, hat eine solche Extensionalitätsbehandlung einen stark differenzreduzierenden Charakter, der sich in einem Paramodulationsansatz für die höhere Stufe (scheinbar) unüberwindbar mit der Idee der Termersetzung vermischt. Die Praktikabilität eines derart gemischten Ansatzes muß aber zunächst stark in Frage gestellt werden — und deshalb erscheint es sinnvoll, nach einem rein differenzreduzierenden Ansatz zur primitiven Gleichheitsbehandlung in höherer Stufe zu suchen und diesen mit einem gemischten Verfahren nach obiger Idee zu vergleichen. Diese

Fragestellung adressiert diese Arbeit in den Kapiteln 5 und 6 (siehe auch [Ben98, Ben99]). Dort wird neben dem gemischt termersetzenden/differenzreduzierenden extensionalen Paramodulationskalkül $\mathcal{EP}$ der rein differenzreduzierende extensionale RUE-Resolutionskalkül $\mathcal{ERUE}$ entwickelt. Die Henkin-Vollständigkeit beider Ansätze wird wiederum mithilfe der in Kapitel 3 eingeführten abstrakten Konsistenzeigenschaften gezeigt und beide Ansätze werden durch zahlreiche Beispiele in Kapitel 8 verglichen.

Vorarbeiten zur Implementierung: Datenstrukturen und Termindexing höherer Stufe Zunächst wurde ein generischer Algorithmus zur Prä-Unifikation höherer Stufe aufbauend auf der KEIM-Toolbox [HKK$^+$94] realisiert. Eine von Michael Kohlhase betreute Diplomarbeit [Kle97] adaptiert zudem die in erster Ordnung entwickelten Termindexing Techniken [Gra95] für die Logik höherer Stufe. Allgemein erlauben diese Termindexing Techniken a) effizientere Unifikationsalgorithmen, b) effizientere Subsumtionstests und c) eine schnellere Bestimmung von Resolutionspartnern. Dabei stützt sich die ebenfalls auf KEIM aufgesetzte Variante des Termindexing für λ-Terme auf die Simplifikation als entscheidbarem und deterministischen Teil der Unifikation höherer Stufe (HOU) und verwendet diese zur Klassifikation von λ-Termen beim Aufbau eines Index.

Implementierung von LEO Die sich anschließende Implementierung von LEO (siehe [Ben97, BK98b]) bedient sich ebenfalls der KEIM-Toolbox und verwendet die oben erwähnten Vorarbeiten. LEO implementiert den Kalkül $\mathcal{ER}$ und basiert auf einer für die höhere Stufe erweiterten Set of Support Strategie (SOS). Gleichheit wird dabei zunächst ausschließlich als ein durch das Leibniz'sche Prinzip definiertes Konzept realisiert. Implementiert wurden zwei Suchstrategien, die in [Ben97] ausführlich diskutiert werden. In dieser Arbeit wird nur die erste der beiden Strategien diskutiert, da diese sich mittlerweile als Standardlösung für LEO etabliert hat. Diese Strategie bedient sich in erheblichem Maße der Unifikation höherer Stufe und Subsumption als Filter, um möglichst frühzeitig solche Klauseln aussortieren zu können, die offensichtlich nicht zur Widerlegung beitragen können oder dazu nicht benötigt werden. Dabei geht diese Strategie einen erheblichen Rechenaufwand ein.

Ein interessantes Problem ergab sich im Zusammenhang mit der Verwendung der oben erwähnten Indexing-Techniken höherer Stufe im Beweiser LEO: Bei der Realisierung eines extensionalen Resolutionskalküls eignen sich diese lediglich zur Unterstützung der Subsumption und sind größtenteils unbrauchbar zur Bestimmung von Resolutionspartnern oder innerhalb der Unifikation. Dies soll wieder anhand eines Beispiels verdeutlicht werden.

Gegeben seien die folgenden Klauseln: $[empty_{(\iota\to o)\to o}(\lambda X_\iota.\, rot_{\iota\to o}X \land kreis_{\iota\to o}X)]^T$ und $[empty_{(\iota\to o)\to o}(\lambda X_\iota.\, kreis_{\iota\to o}X \land rot_{\iota\to o}X)]^F$. Die erste Klausel beschreibt, daß die Menge der roten Kreise $(\{X|rotX \land kreisX\})$ leer ist, während die zweite behauptet, daß die Menge der Kreise, die rot sind $(\{X|kreisX \land rotX\})$ nicht leer ist. Einziger möglicher Beweisschritt für LEO ist die Resolution zwischen beiden Einerklauseln. Resultat ist das Unifikationsconstraint $[empty_{(\iota\to o)\to o}(\lambda X_\iota.\, rot_{\iota\to o}X \land kreis_{\iota\to o}X) = empty_{(\iota\to o)\to o}(\lambda X_\iota.\, kreis_{\iota\to o}X \land rot_{\iota\to o}X)]^F$. Dieses Beispiel entspricht im Wesentlichen dem Unifikationsconstraint, das bereits zu Beginn dieses Vorworts diskutiert wurde. Mit Hilfe der Extensionalitätsregeln des Kalküls $\mathcal{ER}$ kann die Widerlegung durch den Beweiser LEO leicht gefunden werden. Verwendet LEO allerdings zur Bestimmung aller möglichen Resolutionsschritte zwischen zwei einzelnen Klauseln oder Mengen von Klauseln das für die Logik höhere Stufe erweiterte Indexingverfahren [Kle97], so wird obiger initialer Resolutionsschritt nicht einmal vorgeschlagen und der Beweis kann nicht gefunden werden. Das Problem ist, daß ein Indexing Verfahren, das auf rein syntaktischer Unifikation, bzw. Simplifikation, aufbaut nicht berücksichtigt, daß syntaktisch nicht angleichbare Terme potentiell durch Hinzunahme der Extensionalitätsprinzipien doch noch angeglichen werden können. Dies zeigt, daß ein Beweiser wie LEO zur Bestimmung von Resolutionspartnern auf ein syntaktisches Indexingverfahren verzichten muß, oder ein erweitertes Indexingverfahren entwickelt werden müßte. Das gleiche Problem stellt sich bei Verwendung von Indexingverfahren innerhalb der Unifikation. Lediglich zum Subsumtionstest können Indexingverfahren weiter als Filter eingesetzt werden. Dabei werden natürlich nur solche Klauseln ausgefiltert, die eine andere Klausel hinsichtlich der syntaktischen Unifikation, bzw. Simplifikation, subsumieren. Klauseln, die auf der Grundlage eines extensionalen Subsumtionsbegriffes[5] ausgefiltert werden könnten, bleiben dabei unberücksichtigt.

Fallstudien Der Implementierung von LEO schloß sich eine Fallstudie an, die die Praktikabilität des Kalküls $\mathcal{ER}$ für bestimmte Problemdomänen belegen sollte. Wie bereits die Beispiele in diesem Vorwort andeuten, hat LEO seine Stärken beim Beweisen einfacher Beispiele über Mengen, für die die Extensionalitätsprinzipien von essentieller Bedeutung sind. In diesem Vorwort haben wir bereits mehrfach davon Gebrauch gemacht, daß eine Menge A von Individuen eines Typs α sehr einfach mit Hilfe der zugeordneten charakteristischen Funktion $\lambda X_\alpha.\ a_{\alpha\to o}X$ beschrieben werden kann. Zum Beispiel wurde die Menge aller roten Kreise $\{X|rotX \wedge kreisX\}$ kodiert als $\lambda X_\iota.\ rot_{\iota\to o}X \wedge kreis_{\iota\to o}X$. Konkret untersucht wurden die Beispiele des Artikels *Boolean Properties of Sets* [TS89] aus der Mizar-Bibliothek; Es handelt sich dabei um 97 Theoreme aus der einfachen Mengentheorie. LEO findet mit Ausnahme der Beispiele 56 und 57 alle Beweise, davon die meisten innerhalb einer Sekunde. Damit ist LEO auf diesen Problemen leistungsfähiger als die stärksten automatischen Beweiser erster Stufe (unter `http://www-irm.mathematik.hu-berlin.de/~ilf/miz2atp/mizstat.html` wird das Problemlöseverhalten der Systeme SPASS, OTTER, PROTEIN, SETHEO GANDALF und LEO beschrieben). Die Fallstudie wird zur Zeit fortgesetzt — und ohne weitere Modifikationen an LEO konnten bereits zahlreiche der schwierigeren Beispiel des Mizar-Artikels *Some Basic Properties of Sets* [Byl89] gelöst werden.

Integration von LEO in ΩMEGA LEO wurde gegen Ende dieser Doktorarbeit in das mathematische Assistenzsystem ΩMEGA [BCF+97] integriert. Damit steht ΩMEGA nun neben den bereits zuvor angeschlossenen externen Beweissystemen erster Stufe (OTTER [McC97b], SPASS [WGR96], PROTEIN [BF94] und andere) auch ein Beweiser höherer Stufe zur Verfügung, der auf Extensionalität spezialisiert ist.

Da LEO innerhalb eines gemeinsamen LISP Prozesses mit ΩMEGA arbeitet und zudem auf den gleichen Datenstrukturen (KEIM-Toolbox) operiert, kann LEO im Gegensatz zu einer rein externen Komponente auch von anderen Problemlösekomponenten in ΩMEGA verwendet werden. So benötigt zum Beispiel ΩMEGAs Beweisplaner (siehe z.B. [Mel97], [CrS98]) die extensionale Unifikation höherer Stufe. Für LEO selbst ergibt sich der Vorteil, daß einerseits direkte Zugriffe auf ΩMEGAs zentrale Beweisdatenstruktur (PDS) und auf ΩMEGAs mathematische Wissensbank [Koh98b] möglich sind.

Zur Zeit wird untersucht, inwiefern LEO in ΩMEGAs Agentenmechanismus [BS98a] zur Unterstützung des Benutzers bei interaktiven Beweisen integriert werden kann. Dieser Mechanis-

[5] Ein solcher extensionaler Subsumtionstest wäre zwangsläufig unentscheidbar und sehr rechenaufwendig und erscheint deshalb nicht praktikabel.

mus hat das Ziel, dem Benutzer in interaktiven Sitzungen möglichst geeignete Beweisschritte für das weitere Vorgehen vorzuschlagen. Dazu arbeiten im Hintergrund des ΩMEGA-Systems eine Vielzahl kleiner Agenten, die den aktuellen Beweiszustand analysieren und ihre Einzelanalysen über eine hierarchische Blackboardarchitektur miteinander kommunizieren. Die besten Vorschläge werden ausgefiltert und dann an den Benutzer weitergereicht. LEO kann nun selbst in einen solchen Beweisagenten eingebettet werden. Dieser Agent analysiert anhand heuristischer Kriterien, ob die Anwendung auf ein Unterziel im aktuellen Beweiszustand erfolgversprechend ist und ruft dann LEO auf. Sobald — für den Benutzer zunächst unbemerkt — ein Beweisproblem im Hintergrund durch LEO gelöst wurde, wird das Unterziel in ΩMEGAS PDS automatisch geschlossen und dies dem Benutzer signalisiert.

Parallel zur Integration von LEO wurde auch das an der Carnegie Mellon University entwickelte Beweissystem höherer Stufe TPS [AINP90, ABI$^+$96] an ΩMEGA angebunden (siehe [BS98b]). Im Gegensatz zum Resolutionsbeweiser LEO baut TPS auf dem Matrixverfahren auf. Gerade bei der Extensionalitätsbehandlung hat TPS allerdings Schwächen und kann viele der in diesem Bericht diskutierten Beispiele nicht automatisch (ohne explizite Zugabe der jeweils erforderlichen Extensionalitätsaxiome) lösen. Andererseits ist das seit ca. 20 Jahren entwickelte TPS-System in der Lage, tiefer in große Suchräume abzutauchen, als es derzeit noch im prototypischen LEO-System möglich ist. Zudem stellt TPS einen geschickten Mechanismus zur selektiven Expansion von Definitionen in Beweisproblemen [BA98] bereit. Leistungsvergleiche zwischen beiden Systemen werden nun durch die gemeinsame Verfügbarkeit im ΩMEGA-System erheblich erleichtert. Im Hinblick auf das Fernziel der Mechanisierung mathematischen Problemlösens erscheint es aber wesentlich interessanter, daß nun eine Kooperation der sich ergänzenden Systeme in ΩMEGA möglich wird. Die Idee dabei ist, auf ΩMEGAS Beweisplanungsebene spezielle Beweismethoden zu spezifizieren, um eine geschickte und informierte Kooperation beider Systeme zu realisieren. Diese Idee wird in [BBS99] näher beleuchtet und durch Beispiele motiviert.

Ein langfristige Hoffnung des Autors ist es, daß durch die zielgerichtete Kooperation von TPS, LEO und den zahlreichen verfügbaren Beweisern erster Stufe in ΩMEGA durch ΩMEGA's Beweisplaner gesteuert, weitaus schwierigerere und herausfordernde Beweisprobleme gelöst werden können und dies damit einer Mechanisierung mathematischer Beweisens näher bringt.

Chapter 1

Introduction

1.1 Motivation

The dream of formalising and mechanising mathematical reasoning — which also motivates this thesis — reaches back to Gottfried Wilhelm Leibniz. He conceived a *lingua characteristica* and a *calculus ratiocinator*, i.e., a most general framework to mechanise human reasoning, which was certainly inspired by his own contributions to the mechanisation of simple arithmetical operations: He developed and realised a mechanical calculator capable of multiplication. But Leibniz' contributions to the mechanisation of mathematical reasoning in general did not go beyond elemetnary stages and it was not until the end of the 19th century that the field of modern mathematical logic was born. Taking the important contribution of the ancient Greeks (mostly Aristoteles) into account, who already investigated the laws of human thought and developed a theory of well-chosen axioms and rules, it is probably better to say 'It was not until the 19th century that modern mathematical logic was reborn'. It was mainly the work of George Boole (1815-1864), Gottlob Frege (1848-1925) and Bertrand Russel (1872-1970) that stimulated the new and deep interest of many researchers in the field of mathematical logic. Frege's fundamental *Begriffsschrift* is described by Davis [Dav83] *not only as the direct ancestor of contemporary systems of mathematical logic but also as the ancestor of all formal languages, including computer programming languages.*

Another milestone in the formalisation of mathematics to be mentioned is Hilbert's ambitious program at the very beginning of this century [Hil04, Hil27], which aimed at the complete development of modern mathematics in a formal system. In the early 30's results came fast: Whereas Kurt Gödel, Jacques Herbrand and Thoralf Skolem proved the completeness of the (first-order) predicate calculus in 1930 [Göd30, Her30, Sko28] — i.e., every valid formula in the language of the predicate calculus is derivable from its axioms — it was Gödel who showed in his famous incompleteness theorems [Göd31] that it is impossible to develop a generally complete calculus that mechanises mathematical reasoning. More precisely, Gödel showed that as soon as a system is rich enough to encode peano arithmetic, one can construct sentences that are valid in peano arithmetic but which are not derivable in the system itself.

Caused by the development of electronic computers in the 40's and 50's disappointment gradually gave away to attempts of developing and implementing proof procedures in practice. It took quite a few years until J.A. Robinson achieved a first important break-through with his resolution approach in 1965 [Rob65]. The most important improvement of this approach compared to former ones is that in order to prove a theorem it tries to refute the negated theorem in a goal directed way, thereby employing first-order unification as a powerful filter instead of simply enumerating the Herbrand universe like most earlier methods. Robinson's ideas are still employed in many state of the art theorem provers such as OTTER [MW97, McC94], EQP (which recently solved the Robbins Problem [McC97b]), or the superposition based prover SPASS [WGR96]. Even tableaux based provers like PROTEIN [BF94] or SETHEO [GLMS94] are rather closely related to the resolution approach and unification became an essential (filtering) tool for the whole field.

Unfortunately rather few pioneers dared to tackle the mechanisation of higher-order logic based on the simply typed λ-calculus — also called *classical type theory*. For instance, Robinson presents in [Rob68, Rob69] a higher-order proof procedure based on the tableaux idea that itself employs many ideas from the calculi given in [Sch60] and [Tak53]. The most important works to be mentioned are Peter Andrews' investigation of higher-order resolution [And71], Jensen and Pietrowski's approach [JP72] and especially Gerard Huet's constrained resolution approach [Hue72, Hue73a]. It is well known that one of the great challenges for the mechanisation of classical type theory is the undecidability of higher-order unification [Luc72, Hue73b, Gol81]. Whereas Andrews' resolution approach still avoids unification (and instead employs an enumeration of the universe), Huet's constrained resolution approach [Hue72] solves the problem by encoding the particular unification problems as unification constraints and by delaying the application of higher-order unification until the end of a refutation. Huet's approach additionally gains from the higher-order pre-unification algorithm [Hue75] which avoids the guessing aspects of full higher-order unification (for early non-complete approaches to higher-order unification see [Gou66, Dar71, Ern71]; the first complete approaches are presented in [Hue72] and [JP72]) and which fortunately turned out to be sufficient within a refutation approach.

At present, the most powerful higher-order theorem prover is the TPS-system [ABI$^+$96, AINP90] developed at Carnegie Mellon University which is based on the mating approach [And76, And81, And80, Bib83]. This system and the LEO-prover described in this thesis demonstrate the practical feasibility of automated higher-order theorem proving — at least for simple mathematical theorems.

All approaches to automated higher-order theorem proving mentioned above necessarily lack completeness with respect to the intuitive notion of standard semantics as shown by Gödel in 1931 [Göd31]. Instead modern calculi (and systems) aim at completeness with respect to Henkin semantics, which has been invented by Leon Henkin [Hen50, Hen96] and is known as the most general notion of semantics for classical type theory so that complete calculi are possible. Henkin semantics, in [And71] also called general models, thus became the theoretical surveyor's wooden rod for all calculi in this field.

An interesting aspect of classical type theory is that equality is definable in a very natural way (if the underlying notion of semantics is strong enough). For instance, one can express the *Leibniz principle* of equality — *two things are equal, iff they have the same properties* — very easily in classical type theory:[1]

$$\doteq^{\alpha} \; := (\lambda X_\alpha Y_\alpha.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y)$$

In this sense, all approaches mentioned above provide for a very natural equality treatment. But in order to ensure Henkin completeness, they all (including the more recent

[1] Note that this formula describes an implication instead of an equivalence as one may expect. This is sufficient as one easily gets the other direction by contraposition when instantiating P with $\neg P$.

calculi [Wol93], [Koh94b] and [Koh95]) require the extensionality axioms of Leibniz equality (i.e. the infinitely many *functional extensionality axioms*[2] and the *axiom for Boolean extensionality*[3]) to be added to the search space. Furthermore, there are no special techniques for the mechanisation of equality reasoning as in first-order theorem proving.

We can now precisely formulate the goals of this thesis, which are:

1. The clarification of the role of equality and extensionality in automated higher-order theorem proving.
2. The development of proof techniques for analysing higher-order calculi with respect to Henkin completeness.
3. The development of a Henkin complete resolution calculus that avoids the extensionality axioms in the search space.
4. The development of Henkin complete approaches to primitive equality in higher-order resolution based theorem proving.
5. The demonstration of the practicability of these approaches via an implementation.

In the remaining parts of the introduction we will illustrate the different aspects of this thesis in more detail.

1.2 Higher-Order Logic

A higher-order logic is any simply typed logical system that allows quantification over function and predicate variables. It was Bertrand Russel [Rus02, Rus03] who first pointed out in 1902 that in connection with the comprehension principles[4] this may allow for *paradoxes*. The most prominent example is the set of all non-self-containing sets (also called *Russel's paradox*). As a possible solution Russel suggested a few years later in [Rus08] a theory of types as a basis for the formalisation of mathematics that differentiates between objects and sets (or functions) consisting of these kinds of objects. This idea was also taken up by Alonzo Church in 1940, who invented the *simply typed λ-calculus* [Chu40] in order to prevent such paradoxes in the untyped λ-calculus, which he developed with Schönfinkel and Curry ten years earlier. Needless to mention that typed and untyped λ-calculi play an important or even central role in many research fields of modern computer science. Consequently there are several modern textbooks for the typed and untyped λ-calculus available and we refer to [Bar92, And86, HS86, Bar84] for details.

The avoidance of paradoxes like Russel's paradox is also a main reason why we employ a logic based on Church's simply typed λ-calculus [Chu40] — i.e., *classical type theory* — in this thesis. There are certainly other approaches, e.g., Zermelo-Fränkel [Zer08, Frä22b, Frä28] or

[2] The functional extensionality axioms express that two functions are equal, iff they are equal on all arguments, and they are of form: $\forall F_{\alpha\rightarrow\beta}.\ \forall G_{\alpha\rightarrow\beta}(\forall X_{\beta}.\ F\ X \doteq G\ X) \Rightarrow F =^{\beta} G$. We need one axiom for each pair of types α and β.

[3] The Boolean extensionality axiom expresses that on the domain of truth values, which contains exactly two truth values in standard or Henkin semantics, the equality relation and equivalence relation coincide: $\forall A_o, B_o.\ (A \doteq^{o} B) \equiv (A \equiv B)$.

[4] These principles assure the existence of certain functions, cf. the type restricted comprehension axioms below.

von Neumann's [Neu25] set theory, that solve the paradox problems and they are often more popular among mathematicians as a basis for the formalisation of mathematics.

An argument for type theory, however, is that it allows for more natural and intuitive problem formulations as well as solutions (see the introduction in Andrews' textbook [And86]). Furthermore, the choice of classical type theory has additional advantages as the λ-binding construct in combination with the λ-conversion rules in these languages have the effect that the type restricted *comprehension axioms* become derivable (see also [And86]). The comprehension axioms are of the form $\exists U_{\overline{\alpha^n} \to \beta} . \forall \overline{X^n} . (U\ \overline{X^n}) = \mathbf{B}_\beta$ (for an arbitrary term $\mathbf{B}$ such that the variable U does not occur free in $\mathbf{B}$, types $\overline{\alpha^n}$ and β, and $n \geq 0$) and their intention is to guarantee for each expression $\mathbf{B}$ the existence of the functions (or sets) referred by $U_{\overline{\alpha^n} \to \beta}$. For instance, if β is the type o of truth values the comprehension principle asserts that for any respective formula $\mathbf{B}$ there exists a set u (referred to by the variable U) which contains exactly all those elements $\overline{v^n}$ (referred to by variables $\overline{V^n}$) for which $\mathbf{B}$ evaluates to true. But fortunately the required sets and functions u can easily be directly described in the λ-calculus by the term $\lambda \overline{X^n} . \mathbf{B}$, so that the implicit requirement that each term in our language indeed has a denotation (which we call *Denotatpflicht*), already ensures their existence.

Another advantage of classical type theory is that the functional extensionality principle is, at least to some extent, automatically built into the language. This is due to the λ-conversion rules — especially the η-rule which expresses the convertibility of all terms of the form $(\lambda X. \mathbf{A}\ X)$ to $\mathbf{A}$ in case variable X does not occur free in $\mathbf{A}$ — and the existence of respective normal forms allows us to reduce all terms, e.g., to $\beta\eta$-normal form or $\beta\eta$-head-normal-form (see [Bar92, Bar84]). Also the standard higher-order unification or pre-unification algorithm [Hue75, SG89, Sny91] already applies the functional extensionality principle in a straightforward way in case at least one of the two terms to be unified is a λ-abstraction.[5]

Unfortunately the importance of the Boolean extensionality principle for the mechanisation of classical type theory has widely been overlooked. If one is interested only in *syntactical term rewriting or narrowing* in the simply typed λ-calculus (see for instance [Pre98, NP98, NM98b, Pre95, Pre94, vO94, Wol93]) then the Boolean extensionality principle certainly is not of importance as the special type o denoting the (exactly two valued) set of truth values and the logical connectives, which have fixed instead of arbitrary denotations, do not occur in the language. We clarify our notion of *syntactical term rewriting*: As already mentioned, higher-order (pre-)unification (as well as higher-order matching) indeed unifies terms modulo the functional extensionality principles. And in this sense unification and matching in the simply typed λ-calculus can certainly be seen as an E-unification algorithm, where E is the theory defined by the functional extensionality axioms. But the functional extensionality principles are quite naturally and without much effort built-in into the λ-calculus as well as the traditional unification and matching algorithms. Thus, when we speak of *syntactical* higher-order unification or term rewriting we mean this in the latter sense and consider functional extensionality as automatically built-in.

The interest in syntactical higher-order term rewriting is motivated by potential applications in different fields such as functional programming or program verification. The main challenge recently is to develop suitable term-orderings in order to orientate and complete

[5] Obviously, if one requires the unification terms to be in $\beta\eta$-normal form, any two terms of functional type must be λ-abstractions. If instead one considers only $\beta\eta$-head-normal form terms they not necessarily have to be λ-abstractions. In the latter case additional rules are probably needed in the unification calculus (see for instance the sorted higher-order unification calculus discussed in [Koh94b]).

the equations of a rewriting system. All results on syntactical rewriting are therefore of great interest for the field of automated (or interactive) higher-order theorem proving, too, as there are many application domains in this fields in which syntactical rewriting can fruitfully be employed.

But in this thesis we will illustrate that equational theorem proving in classical type theory is generally much more challenging and complicated as pure syntactical term rewriting. Apart from the functional extensionality principle — which is to some extent addressed by syntactical term rewriting — the Boolean extensionality principle is of importance, too. For example, in equational theorem proving for classical type theory one is also interested in unifying — and rewriting — on extensionally equal terms like $(\lambda X.\ red\ X \land circle\ X)$ and $(\lambda X.\ circle\ X \land red\ X)$ or $(\lambda X.\ \mathbf{A} \lor \neg\mathbf{A})$ and $(\lambda X.\ \mathbf{B} = \mathbf{B})$. It is quite obvious that the pure syntactical higher-order unification approach (i.e., syntactical modulo functional extensionality only) is much too weak to solve such unification problems. This example also illustrates that higher-order unification with respect to both extensionality principles requires the application of a general higher-order theorem prover in the sense of a embedded subsystem within higher-order unification in order to examine whether two terms of type o are equal. This illustrates the challenging difference between simple syntactical rewriting, where at most the functional extensionality principle is taken into account, and equational reasoning in classical type theory where both extensionality principles are of importance.

Another aspect of higher-order logic to be clarified concerns the *axiom of choice* and the *description operator*. It is well known (see [Frä22a, Göd40]) that the axiom of choice is independent and consistent in set theory. A respective result for classical type theory is shown in [And72b]. The axiom of choice expresses that there exists a function f that chooses exactly one element out of each set belonging to an arbitrary family of sets. This axiom — more precisely this axiom scheme — has a non-constructive character and, e.g., allows a very simple but non-constructive proof, first presented by Zermelo [Zer04] in 1904, for Cantor's conjecture that every set can be well ordered (for a modern discussion of the axiom of choice see also [Jec77]). As many mathematicians have objections to the non-constructive character of the axiom of choice we will not treat it as an imperative in our logic. Analogously we do not treat the description operator as automatically built-in. The description operator, which allows for the description of possibly non-existing objects like Santa Claus (note that introducing a constant in a signature for Santa Clause automatically requires his existence by the Denotatpflicht), was first examined by Alfred N. Whitehead and Bertrand Russel [WR10] (see also [Sco67, And72b] for a discussion).

We want to point out that although the axiom of choice is not built-in, non-constructive proofs are possible in our framework. We still consider a classical notion of higher-order logic in which, e.g., the tertium-non-datur principle is valid and proofs can be carried out indirectly.

This is in contrast to intuitionistic type theory [ML94, GJ98] which was invented by Martin Löf in the early 70's to provide a formal foundation for constructive mathematics. Intuitionistic type theory has become a very important and active field in computer science and is successfully applied, for instance, for the verification of computer programs and languages. Hofmann discusses in [Hof97] that the extensionality principles are quite important even in intuitionistic type theory for the formalisation of mathematics, but they unfortunately cause major problems in the mechanisation.

Summing up, we employ in this thesis a classical higher-order logic based on the simply typed λ-calculus (classical type theory) and treat the axiom of choice and the description operator as optional. With respect to semantics we are interested in standard semantics and with respect to the completeness of our calculi we shall refer to Henkin semantics. In standard as well as in Henkin semantics both extensionality principles (the functional as well as the Boolean) are valid and for equational reasoning in classical type theory both principles need to be mechanised in an appropriate way. This differentiates higher-order equational reasoning, which is the topic of this thesis, from syntactical higher-order term rewriting.

In this thesis we shall differentiate between five notions of equality: If we *define* a concept we use $:=$ (e.g., let $\mathcal{D} := \{\mathtt{T}, \mathtt{F}\}$) and $\equiv$ represents *meta-equality*. We denote the equality relation as an *object* of our *semantical domains* with q; note that there is at most one q^α in each domain $\mathcal{D}_\alpha$. The remaining two notions, $=$ and $\doteq$, are related to syntax, where $=^\alpha$ may occur as a *constant symbol* of type α in a signature Σ and finally $\doteq^\alpha$ as an abbreviation for the respective formula expressing *Leibniz equality*. Whereas $\doteq^\alpha$ only denotes semantical equality relation q^α if the underlying semantical notion is strong enough (e.g., in Henkin semantics), $=^\alpha$ always denotes the respective semantical equality relation.

1.3 Higher-Order Model Existence

In classical first-order predicate logic, it is straightforward to assess the deductive power of a calculus: first-order logic has a well-established and intuitive set-theoretic semantics, relative to which completeness can be verified using, for instance, the abstract consistency method (see the introductory textbooks [And86, Fit96]). This well-understood meta-theory supported the development of different calculi well.

In higher-order logics, the situation is rather different: the intuitive set-theoretic standard semantics cannot give a sensible notion of completeness [Göd31]. However, there is a more general notion of semantics (the so-called Henkin semantics [Hen50]), that allows complete calculi and which sets the standard today for deductive power of calculi.

Peter Andrews' *Unifying Principle for Type Theory* [And71] provides a method of higher-order abstract consistency that has become the standard tool for completeness proofs in higher-order logic, even though it can only be used to show completeness relative to a certain Hilbert style calculus $\mathfrak{T}$. A calculus $\mathcal{C}$ is called complete relative to a calculus $\mathfrak{T}$, iff $\mathcal{C}$ proves all theorems of $\mathfrak{T}$. Since $\mathfrak{T}$ is not necessarily complete with respect to Henkin models, the notion of completeness that can be established by this method is a strictly weaker notion than Henkin completeness.

As a consequence, the calculi developed for higher-order automated theorem proving [And71, Hue72, Hue73a, JP72, Mil83, Koh94b, Koh95] and the corresponding theorem proving systems such as Tps [ABI$^+$96], are not complete with respect to Henkin models. Moreover, they are not even sound with respect to $\mathfrak{T}$, since all of them utilise η-conversion, which is not admissible in $\mathfrak{T}$. In other words, their deductive power lies somewhere between $\mathfrak{T}$ and Henkin models.

Here the aim of this thesis is to provide a semantical meta-theory that will support the development of higher-order calculi for automated theorem proving just as the corresponding methodology does in first-order logic. To reach this goal, we establish

- classes of models that adequately characterise the deductive power of existing theorem-proving calculi (making them sound and complete), and
- a standard methodology of abstract consistency proof methods (by providing the necessary model existence theorems, which extend Andrews' Unifying Principle), so that

the completeness analysis for higher-order calculi will become as simple an exercise as in first-order logic.

Due to the inherent complexity of higher-order semantics we give an informal exposition of the issues covered and the techniques applied.

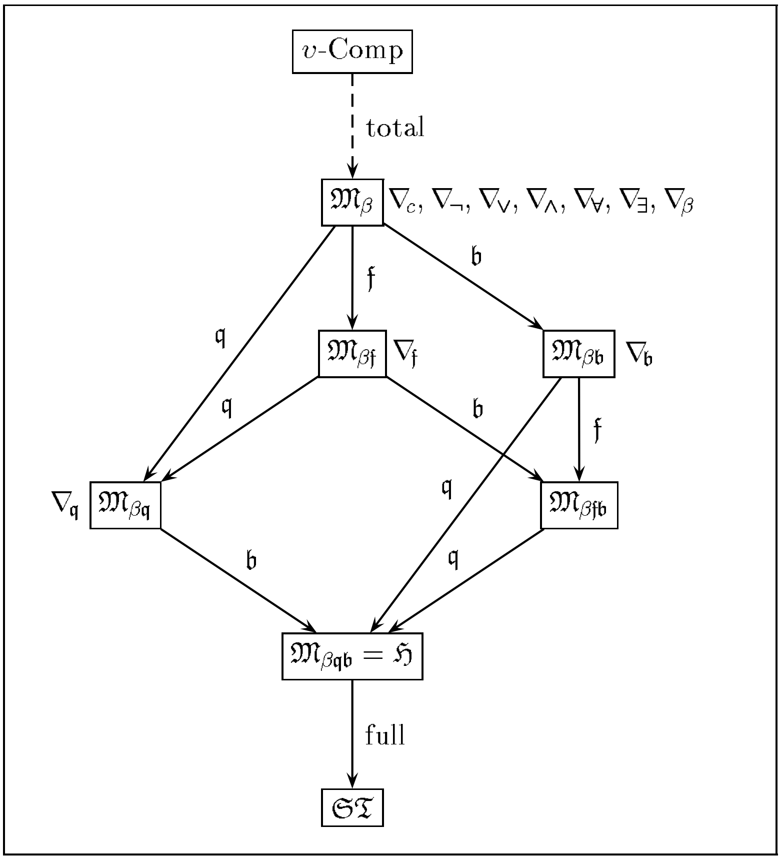

υ-Comp
total
$\mathfrak{M}_\beta$
∇_c, $\nabla_\neg$, $\nabla_\vee$, $\nabla_\wedge$, $\nabla_\forall$, $\nabla_\exists$, ∇_β
𝔟
𝔣
𝔮
$\mathfrak{M}_{\beta\mathfrak{f}}$
$\nabla_\mathfrak{f}$
$\mathfrak{M}_{\beta\mathfrak{b}}$
$\nabla_\mathfrak{b}$
𝔮
𝔟
𝔣
$\nabla_\mathfrak{q}$
$\mathfrak{M}_{\beta\mathfrak{q}}$
𝔮
$\mathfrak{M}_{\beta\mathfrak{fb}}$
𝔟
𝔮
$\mathfrak{M}_{\beta\mathfrak{qb}} = \mathfrak{H}$
full
$\mathfrak{ST}$

Figure 1.1: The landscape of Higher-Order Semantics

Let us now explore the particular semantic notions (see Figure 1.1). We will discuss the model classes from bottom to top, from the most specific notion of standard models ($\mathfrak{ST}$) to the most general notion of v-complexes, motivating the respective generalisations as we go along. In Chapter 2.1, we will proceed the other way round, specialising the notion of a Σ-model ($\mathfrak{M}_\beta$) more and more.

The symbols in the boxes in Figure 1.1 denote model classes, the symbols labelling the arrows indicate the properties inducing the corresponding specialisation, and the ∇-symbols next to the boxes indicate the clauses in the definition of the corresponding abstract consistency class (cf. 3.4) that are needed to establish a model existence theorem for this class of models.

A *standard model* ($\mathfrak{ST}$, cf. Definition 2.31) for our higher-order logic provides a fixed set $\mathcal{D}_\iota$ of individuals, and a set $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ of truth values. All the domains for the complex types are defined inductively: $\mathcal{D}_{\alpha\to\beta}$ is the set of all functions $\{f | f : \mathcal{D}_\alpha \to \mathcal{D}_\beta\}$. The evaluation function $\mathcal{I}_\varphi$ with respect to an interpretation $\mathcal{I} : \Sigma \to \mathcal{D}$ of constants and an assignment

φ of variables is obtained by the standard homomorphic construction that evaluates a λ-abstraction with a function, whose operational semantics is specified by β-reduction.

The notion of *Henkin models* ($\mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$ or $\mathfrak{H}$) generalises that of standard models in the sense that instead of requiring $\mathcal{D}_{\alpha\rightarrow\beta}$ to be the full set of functions it only requires that $\mathcal{D}_{\alpha\rightarrow\beta}$ has enough members such that any well-formed formula can be evaluated.[6] Note that with this generalised notion of a model, there are fewer formulae that are valid in all models (intuitively, for any given formulae there are more possibilities for counter-models). In fact the generalisation to Henkin models restricts the set of valid formulae sufficiently, so that all of them can be proven by a Hilbert-style calculus [Hen50]. Thus, whereas standard semantics does not allow complete calculi due to Gödels result [Göd31], Henkin semantics does [Hen50, Hen96].

It is matter of folklore that a primitive notion of equality (expressed by a primitive equality constant $= \in \Sigma$) is not strictly needed, since it can be expressed by the Leibniz formula. However, the Leibniz formula only denotes the semantic equality relation if $\mathcal{D}_{\alpha\rightarrow o}$ contains enough properties to discern members of $\mathcal{D}_\alpha$; in fact, we have to ensure that for all $a \in \mathcal{D}_\alpha$, the singleton set $\{a\}$ is in $\mathcal{D}_{\alpha\rightarrow o}$ (see the proof of Lemma 2.36).[7] In other words, we are in the somewhat paradoxical situation, that Leibniz equality (which is commonly used as a substitute for primitive equality) will only denote semantical equality, if we can guarantee that the identity relation is already present in the model (we call this property $\mathfrak{q}$, cf. Definition 2.27). Hence we introduce the corresponding semantical structures, namely Henkin models without property $\mathfrak{q}$ ($\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$), in which property $\mathfrak{q}$ is not necessarily valid and thus Leibniz equality does not necessarily denote the equality relation. An example of a theorem which is valid within the class of Henkin models but is not in the class of $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$'s is given by the axiom of functional extensionality for Leibniz equality ($\forall F_{\alpha\rightarrow\beta}. \forall G_{\alpha\rightarrow\beta} (\forall X_\beta. FX \doteq GX) \Rightarrow F \doteq^\beta G$), cf. Lemma 2.37.

The next generalisation of model classes derives from the fact that we want to characterise the deductive power of higher-order theorem provers at a semantic level (we will take Tps [ABI$^+$96] as an example). Note that Tps cannot be complete with respect to Henkin models and is even not generally complete for $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$'s, although there is some 'extensionality treatment' built into the proof procedure. The incompleteness of Tps for Henkin models[8] can be seen from the fact that it fails to refute formulae such as $c\mathbf{A}_o \wedge \neg c(\neg\neg\mathbf{A})$, where c is a constant of type $o \rightarrow o$ or $c(\lambda X_\alpha. \mathbf{B}X \wedge \mathbf{A}X) \Rightarrow c(\lambda X_\alpha. \mathbf{A}X \wedge \mathbf{B}X)$, where c is a constant of type $(\alpha \rightarrow o) \rightarrow o$. The problem in the former example is that the higher-order unification algorithm employed by Tps cannot determine that $\mathbf{A}$ and $\neg\neg\mathbf{A}$ denote identical semantic objects (by Boolean extensionality as already mentioned before), and thus returns failure instead of success. In the second example the principle of functional extensionality is needed in addition in order to prove the theorem.

The lack of completeness of refutation procedures like Tps occurs especially in a situation, where formulae contain occurrences of propositional formulae dominated by uninterpreted constants or variables or where this problem is mixed with the problem of functional exten-

[6] In other words: the functional universes are rich enough to satisfy the comprehension axioms.

[7] On a similar note, Peter Andrews remarked in [And72a] that if the set $\mathcal{D}_{\alpha\rightarrow\alpha\rightarrow o}$ is so sparse that the semantic identity relation is not present, it is then possible to construct a Henkin model where Leibniz equality is non-extensional.

[8] In case the extensionality axioms are not available in the search space. Note that one can add extensionality axioms to the calculus in order to achieve, at least in theory, Henkin completeness. But this increases the search space drastically and is not feasible in practice.

sionality; in our examples the function constant c dominates the proposition $\mathbf{A}_o$ or the set expression $\lambda X_\alpha. \mathbf{B}X \land \mathbf{A}X$. To give a semantical characterisation of the deductive power of the TPS procedure, we have to generalise the class of Henkin models further, so that there are counter-models to the examples above. Obviously, this involves weakening the assumption that $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$ (we call this assumption for Henkin models property $\mathfrak{b}$), since this entails that the values of $\mathbf{A}$ and $\neg\neg\mathbf{A}$ are identical: In *functional* Σ*-models* ($\mathfrak{M}_{\beta\mathfrak{f}}, \mathfrak{M}_{\beta\mathfrak{fb}}, \mathfrak{M}_{\beta\mathfrak{q}}$, cf. Definitions 2.28 and 2.24) we only insist that there is a valuation v of $\mathcal{D}_o$, i.e., a function $v\colon \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ that is coordinated with the functions $\mathcal{I}(\neg)$, $\mathcal{I}(\land)$, $\mathcal{I}(\Pi^\alpha)$ and (possibly) $\mathcal{I}(=^\alpha)$, where $\mathcal{I}$ is the interpretation function (i.e., a family of interpretation functions) for the constants given in the signature. Thus we have a notion of validity for Σ: we call a proposition $\mathbf{A}$ valid in $\mathcal{M} := (\mathcal{D}, \mathcal{I}, v)$ under an assignment φ, iff $v(\mathcal{I}_\varphi(\mathbf{A})) \equiv \mathtt{T}$. In our first example, there is a Σ-model structure $\mathcal{M} \equiv (\mathcal{D}, \mathcal{I}, v)$, where $\mathcal{I}_\varphi(\mathbf{A}) \not\equiv \mathcal{I}_\varphi(\neg\neg\mathbf{A})$, and therefore $\mathcal{I}_\varphi(c\mathbf{A}) \not\equiv \mathcal{I}_\varphi(c(\neg\neg\mathbf{A}))$ if we take $\mathcal{I}(c)$ to be the identity function on $\mathcal{D}_\alpha$. In particular, we can have $v(\mathcal{I}_\varphi(c\mathbf{A})) \not\equiv v(\mathcal{I}_\varphi(c(\neg\neg\mathbf{A})))$, and therefore $v(\mathcal{I}_\varphi(c\mathbf{A}_o \land \neg c(\neg\neg\mathbf{A}))) \equiv \mathtt{F}$, since v is a valuation.

Clearly, for functional Σ-models we have the same choices concerning the role of equality, therefore, we distinguish the classes $\mathfrak{M}_{\beta\mathfrak{f}}$ and $\mathfrak{M}_{\beta\mathfrak{q}}$ of functional Σ-models without/with property $\mathfrak{q}$. Furthermore, we have the class $\mathfrak{M}_{\beta\mathfrak{fb}}$ of functional Σ-models with (only) property $\mathfrak{b}$, which expresses that the set of truth values contains exactly two elements, i.e., $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$. Since functional Σ-models with properties $\mathfrak{b}$ and $\mathfrak{q}$ are defined to be Σ-Henkin models, we can also view $\mathfrak{M}_{\beta\mathfrak{fb}}$ as "Henkin models without property $\mathfrak{q}$".

Finally, we drop the requirement of functional extensionality for **Σ-models**, which is encoded as property $\mathfrak{f}$ (cf. Definition 2.24). This is the most general notion of semantics we will discuss in this thesis; we only insist that the evaluation function is a homomorphism which respects instantiation. In such models, a function is not uniquely determined by its behaviour on all possible arguments. For the construction of such models we therefore need labelings for functions (e.g., a green and a red version of a function f) in order to differentiate between them, even though they are functionally equivalent. As already done for functional Σ-models we analyse properties $\mathfrak{q}$ and $\mathfrak{b}$ for non-functional Σ-Models. Whereas $\mathfrak{b}$ may or may not hold for non-functional Σ-Models, it turns out that property $\mathfrak{q}$, i.e., the requirement that the intended equality relations are provided in the respective domains, only makes sense in connection with functionality. More precisely, property $\mathfrak{q}$ is not independent from functionality.

Peter Andrews has pioneered the construction of non-functional models with his v-complexes in [And71]. These are even more general constructions than our Σ-models, since totality of the evaluation function is not assumed. His construction is based on Schütte's semi-valuation method [Sch60], which only needs partial valuations to construct a model for a given Hintikka set.

In this thesis, we concentrate on the other aspects of higher-order models and ensure totality of our evaluation functions by a saturation condition (cf. 3.9) in our abstract consistency classes. This does not restrict the applicability of our model existence theorems, since saturation is relatively simple to prove for a given calculus (see [Koh94b, Koh98a, BK97b]). For all the notions of a model we present model existence theorems tying the differentiating conditions of the models to suitable conditions in the abstract consistency classes (see Chapter 3). We can use the classical construction in all cases: abstract consistent sets are extended to Hintikka sets 3.13, which induce a valuation on a term structure (see Definition 2.14). In some cases, we have to take a quotient structure (see Definition 2.12) to ensure that the set of truth values is exactly $\{\mathtt{T}, \mathtt{F}\}$ for property $\mathfrak{b}$.

The simplest way to ensure property $\mathfrak{q}$ is by assuming that the signature contains a primitive connective for equality, which is evaluated as semantical identity (we call this property $\mathfrak{e}$). We will study the case in Section 3.3. On the one hand, the semantical situation becomes much simpler now (see Figure 2.1), since $\mathfrak{M}_{\beta}$, $\mathfrak{M}_{\beta\mathfrak{f}}$ and $\mathfrak{M}_{\beta\mathfrak{q}}$ are identified, just as $\mathfrak{M}_{\beta\mathfrak{b}}$, $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ and $\mathfrak{H}$, on the other hand, the existence of another logical constant induces further conditions in the definition of the abstract consistency classes.

1.4 Traditional and Extensional Higher-Order Resolution

As we have seen, traditional higher-order resolution approaches [And71], [JP72], [Hue72] as well as the Tps-system generally need to add all extensionality axioms to the search space in order to ensure Henkin completeness. And this holds for more recent approaches like [Koh94b] and [Wol93] as well. Although the functional extensionality principles taken alone is integrated into syntactical higher-order unification both extensionality principles still have to be additionally axiomatised in the search space in order to reach Henkin completeness. This can be seen for problems like $(\lambda X_{\iota}.\ red\ X \wedge (circle\ X \wedge (large\ X \vee \neg(large\ X)))) = (\lambda Y_{\iota}.\ circle\ X \wedge red\ X)$. It does not help at all that functional extensionality is already integrated in higher-order unification as we here obviously need an appropriate combination of both principles in order to equalise the terms. And in the above approaches — which all follow the idea of generally delaying higher-order unification in order to overcome the undecidability problem — such a combination can only be achieved by adding *all* extensionality axioms to the search space.

Wolfram [Wol93] applies higher-order E-unification instead of pure syntactical higher-order unification, but he does not provide an explicit account of theories E containing both extensionality principles: they require potential recursive calls to an (again) Henkin complete higher-order theorem prover. And also the higher-order E-unification algorithm of Snyder [Sny90] provides no solution to the problem with the extensionality principles, as the suggested approach is restricted to first-order theories and, e.g., does not take the Boolean extensionality principle into account.

The idea of such recursive calls was wirst mentioned in [Koh95]. But unfortunately this approach still lacks Henkin completeness as the single rule added to the calculus (which is analogous to the rule *Equiv* in our approach; see Figure 4.2) is not strong enough to realize all the necessary aspects of a suitable interaction of Boolean and functional extensionality.

Another unfortunate aspect about [Koh94b] and [Koh95] is that both approaches not only lack a general extensionality treatment but also lack soundness. E.g., in both approaches it is possible to prove that each function has a fixed point, which can be formulated as follows: $\forall F_{\alpha\to\alpha}.\ \exists X_{\alpha}.\ F\ X = X$. This is caused by the extra logical treatment of Skolemisation with the so called *variable conditions* employed in both approaches. But the conditions added with each eliminated existential quantifier are not strong enough to prevent a proof of the above statement. As the author was not able to suitably fix this problem we employ in this thesis traditional Skolemisation again and avoid the usage of variable conditions.[9] More precisely, we employ Miller's sound adaptation of traditional first-order Skolemisation [Mil83], which associates with each Skolem constant the minimum number of arguments the constant has to

[9] Michael Kohlhase remarked that he is currently working on a solution to the problem. If this solution turns out to be sound, it can an probably should be employed instead of traditional Skolemisation within the calculi presented in this thesis as well.

be applied to. Higher-order Skolemisation becomes sound[10], if any Skolem function f^n only occurs in a *Skolem term*, i.e., a formula $\mathbf{S} \equiv f^n\overline{\mathbf{A}^n}$, where none of the $\mathbf{A}^i$ contains a bound variable. Thus the Skolem terms only serve as descriptions of the existential witnesses and never appear as functions proper.

As already mentioned, a main motivation for this thesis is to get rid of the drawback of the above approaches, which have to add the (generally infinitely many) extensionality axioms to the search space in order to reach Henkin completeness. In fact, none of the currently available systems for classical type theory actually adds the extensionality to the search space but they instead accept incompleteness. The Tps-system [ABI+96] probably offers the most practicable solution to the situation: It avoids the extensionality axioms and instead analyses the input problems in order to try to modify them in an appropriate way (e.g., by applying the functional extensionality principles to input equations of functional type or by replacing equations on type o by equivalences). But Example $\mathbf{E}_3^{ext}$ in Section 8.1 demonstrates that there are many examples which can not be appropriately modified this way before being passed to the refutation process. Also the Hol-system [GM93] and the Isabelle-system, which are other prominent theorem provers for higher-order logic, do not automatically add all extensionality axioms to the search space when proving subgoals automatically.

In order to back up the motivation for this thesis we will briefly sketch the drawbacks of the option to add the extensionality axioms to the search space. This will also clarify why the available systems indeed avoid this option. First, we remember the definition of the functional and Boolean extensionality principles (for Leibniz equality):

$$\mathrm{EXT}_L^{\alpha\to\beta} := \forall F_{\alpha\to\beta}.\ \forall G_{\alpha\to\beta}.\ (\forall X_\beta.\ F\ X \doteq G\ X) \Rightarrow F \doteq^{\beta} G$$

$$\mathrm{EXT}_L^{o} := \forall A_o.\ \forall B_o.\ (A =^{o} B) \equiv (A \equiv B)$$

One challenging problem for the mechanisation of these principles, namely the general need for infinitely many axioms $\mathrm{EXT}_L^{\alpha\to\beta}$, has already been mentioned. But even apart from this problem these axioms would cause an enormous explosion of the search space. This becomes immediately clear if one considers the clauses obtained by the normalisation of just one single extensionality axiom. Note that Leibniz Equality $\doteq$ is just an abbreviation for a higher-order term and that the expanded axioms are thus of form:

$$\begin{aligned}&\mathrm{EXT}_L^{\alpha\to\beta} :=\\ &\quad \forall F_{\alpha\to\beta}.\ \forall G_{\alpha\to\beta}.\ (\forall X_\beta.\ (\forall P_{\beta\to o}.\ P\ (F\ X) \Rightarrow P\ (G\ X))) \Rightarrow (\forall Q_{(\alpha\to\beta)\to o}.\ Q\ F \Rightarrow^{\beta} Q\ G)\end{aligned}$$

$$\mathrm{EXT}_L^{o} :=\ \ \forall A_o.\ \forall B_o.\ (\forall P_{o\to o}.\ (P\ A) \Rightarrow^{o} (P\ B)) \equiv (A \equiv B)$$

Note that normalising these formulas introduces many flexible variables into the search space.[11] And the problem with the free predicate variables at head position in literals — which are also called *flexible literal heads* — is that the *primitive substitution rule Prim* becomes applicable to them (cf. [And71] or rule *Prim* in Figure 4.2 in Chapter 4). This

[10] Without this additional restriction the calculus does not really become unsound, but one can prove an instance of the axiom of choice, which we want to be optional in our approach.

[11] In Chapter 2.8 we will illustrate that there are infinitely many more or less natural ways of defining equality in classical type theory apart from Leibniz equality. Thus it may be possible to find some slightly more appropriate formulations of the extensionality principles for defined equality as the ones presented here, even though this seems to be rather unlikely. It will certainly not be possible, though, to avoid all the free predicate variables introduced here.

rule, which is very important in higher-order theorem proving for reaching completeness, blindly instantiates each flexible literal head with a most general binding (partial binding) that imitates a logical connective, i.e., with a most general formula that introduces a logical connective at head position. Note that there are infinitely many universal quantifiers (one for each type) and thus the primitive substitution rule is infinitely branching. And as this principle aims at introducing most general terms apart from the new logical connective at head position, new free predicate variables are generated which subsequently become — after the necessary normalisation of the modified clauses — new flexible literal heads. And not enough, the primitive substitution rule can thereby even duplicate flexible literals. It is thus obvious that each single clause derived from an extensionality axiom with a flexible literal leads to explosion of the search space that is awkward to handle in practice. Thus, in principle a higher-order theorem prover can spend an arbitrary amount of time just in applying primitive substitution to the clauses belonging to the extensionality principles. And unfortunately the primitive substitution principle cannot be generally avoided as otherwise a higher-order resolution approach even fails to prove such trivial theorems like $\exists X_o.\ X_o$ or $\exists P_{\iota \to o}.\ \forall Y_\iota.\ P\ Y$.

The illustrated serious drawbacks of the option to add all extensionality axioms to the search space was the main motivation for the development of the extensional higher-order resolution calculus $\mathcal{ER}$ presented in Chapter 4, which instead of adding these axioms employs the idea of recursive calls to the overall refutation process from within syntactical higher-order pre-unification as first mentioned in [Koh95]. But in contrast to [Koh95] the approach presented here (which is also described in [BK98a]) realises the necessary interaction of both extensionality principles in a suitable way, and this finally makes it the first Henkin complete refutation approach for classical type theory. Note that this approach can also be viewed as a *test calculus* for general higher-order E-unification in the following sense: If we pass the conditional equations $(\forall \overline{X^n}.\ (\mathbf{L}_1 = \mathbf{R}_1) \land \ldots \land (\mathbf{L}_n = \mathbf{R}_2)) \Rightarrow (\exists \overline{Y^m}.\ (\mathbf{L} = \mathbf{R}))$ to our extensional higher-order resolution calculus, then this calculus tests for the E-unifiability of the term $(\mathbf{L} = \mathbf{R})$, where E is the theory defined by the equations $(\forall \overline{X^n}.\ (\mathbf{L}_1 = \mathbf{R}_1) \land \ldots \land (\mathbf{L}_n = \mathbf{R}_2))$ and the extensionality principles.

1.5 Adding Primitive Equality

Whereas the research in the field of higher-order term rewriting is very active (see for instance [Pre98, NP98, NM98b, Nip95, Pre95, Pre94, vO94, Wol93] for higher-order term rewriting and narrowing or [JR99, JR98, LP95] for recent work on higher-order term orderings) the integration of primitive equality and the application of term rewriting techniques in a refutation based higher-order theorem proving context is still rather unexamined. At a first glance one might assume that as soon as suitable higher-order rewriting techniques are available — the currently most challenging problem is to find suitable term-orderings in higher-order logic — they can be successfully employed in an automated higher-order theorem proving context as well. And furthermore one may argue that because automated first-order provers that are heavily based on term rewriting techniques (such as WALDMEISTER [HBVL97], BLIKSEM or SPASS [Wei97]) seem to have taken the lead in many application domains over systems that employ difference reducing techniques, an analogous situation will arise in higher-order logic as soon as the available higher-order term rewriting approaches become strong enough. Whereas this may indeed happen in specific domains, this thesis provides a counterargument for the overall success of term rewriting techniques in automated higher-order theorem proving by pointing out new serious problems that in addition to the term-ordering problem need to be solved:

- As already motivated higher-order unification takes only the functional but not the Boolean extensionality principle into account.

- In Chapter 2.8 we illustrate that in classical type theory infinitely many different terms (apart from the Leibniz definition) define equality and that we cannot decide whether a (probably automatically generated) proof problem contains a defined equation at some subterm position. Thus, even if we consider a higher-logic with primitive equality and try to employ term rewriting techniques, we cannot generally restrict equality handling in a calculus that aims at Henkin completeness only to primitive equality as our input problem may still contain some defined equations we cannot even detect. And Examples $\mathbf{E}_2^{ext}$ and $\mathbf{E}_3^{ext}$ in Section 8.1 illustrate that there are even theorems which neither contain a defined nor a primitive equation but where the extensionality principles are nevertheless of central importance for the proof. Furthermore, in classical type theory with a primitive notion of equality one unfortunately has to take care of both kinds of extensionality principles, those for Leibniz equality as well as those for primitive equality. And a Henkin complete approach therefore also needs to realise a general interleaving between both concepts of equality. This sharply contrasts with the situation in first-order theorem proving where one can choose between defining equality (e.g., by axiomatising it) or considering a primitive notion of equality and providing new calculus rules. In classical type theory one simply does not have this choice: Defined equality is always built-in.

- In Chapter 5 we adapt traditional first-order paramodulation [RW69] to higher-order logic and define a higher-order paramodulation approach $\mathcal{EP}$ based on the extensional higher-order resolution calculus $\mathcal{ER}$. Whereas the important reflexivity rule (or axiom) known from first-order paramodulation is naturally built-in in our approach, we will show that in order to ensure Henkin completeness additional extensionality rules (or axioms) for primitive equality are needed. The problem is that in classical type theory even single positive equations can be contradictory, which again contrasts with the situation in first-order logic. Unfortunately the needed additional extensionality rules further strengthen (as would the respective axioms) the already strong difference-reducing character of the underlying calculus $\mathcal{ER}$. It will be motivated by examples in Chapter 8 that a proper term rewriting approach will be quite hard to achieve as the realisation of a suitable interaction of the functional and Boolean extensionality principles has a rather natural difference-reducing character.

As the development of suitable heuristics for the higher-order paramodulation approach $\mathcal{EP}$ with its intrinsic mixed term rewriting and difference reducing character seems to be quite difficult (if possible at all) we additionally develop in Chapter 6 the difference-reducing approach $\mathcal{ERUE}$ which adapts the ideas of first-order RUE-resolution [Dig79] to our higher-order setting. This approach faces the same problem about primitive equality as the paramodulation approach $\mathcal{EP}$ and has to add new extensionality for primitive equality in order to reach Henkin completeness. The only difference between $\mathcal{ER}$ and $\mathcal{ERUE}$ is actually that the latter avoids the paramodulation rule and instead allows to resolve and factorise on unification constraints.

Thereby $\mathcal{ERUE}$ gains a pure difference-reducing character that may be easier to guide and handle in practical applications than the intrinsic mixed term rewriting/difference-reducing character of $\mathcal{EP}$. This aspect is illustrated by examples $\mathbf{E}_1^{set}$ and $\mathbf{E}_2^{set}$ in Section 8.6.

1.6 The Leo System

Leo realises the calculus $\mathcal{ER}$ and is based on an extended set of support architecture, that adapts this well known technique from first-order theorem proving (e.g., see [McC94]) with respect to the very specific requirements of extensional higher-order theorem proving. The system is implemented in Common Lisp [Ste90] and employs many data structures and basic algorithms offered by the Keim-toolbox [HKK$^+$94] for deduction systems. This toolbox, e.g., provides the higher-order term indexing module described in [Kle97] that adapts the first-order term indexing techniques of [Gra95] to the higher-order setting.

The prototypical prover Leo has been implemented mainly during a 5 months stay at Carnegie Mellon University, Pittsburgh, USA, and is discussed in Chapter 7 in detail in (see also [Ben97]).

Originally two contrary search strategies have been developed. In this thesis we only discuss the one that has become the standard strategie in Leo. This strategy employs like most first-order approaches unification and subsumption as filter in order to sort out clauses which are either superflous or which cannot contribute to the refutation. The main difference to first-order theorem provers is that these filtering side-computations are computatationally much more expensive in the higher-order setting and generally even undecidable. Thus, Leo artificially restricts and interrupts these side-computations with the result that these filter become quite imperfect. Especially the employed subsumption check is based only on the simplification part of higher-order unification and we do not develop and realise a notion of extensional higher-order subsumption, which would lead to a much stronger and more appropriate (but undecidable) filter for our purposes.

Anyhow the extensionality principles cause new practical problems for Leo. E.g., term indexing techniques cannot be employed to the same extent and for the same purposes as in first-order automated theorem proving. This is illustrated by the following example, which shows that the usage of term indexing techniques within the computation of resolution partners causes incompleteness: Assume we want to prove

$$\forall P_{(\iota\to o)\to o}.\ \{X_\iota | red\ X \land circle\ X\} \in P \Rightarrow \{X_\iota | circle\ X \land red\ X\} \in P$$

while coding sets as characteristic functions and defining $\in\ := \lambda X_\alpha.\ \lambda M_{\alpha\to o}.\ M\ X$. Definition expansion and clause normalisation leads to the clauses

$$\mathcal{C}_1 : [p_{(\iota\to o)\to o}\ (\lambda X_\iota.\ red_{\iota\to o}\ X \land circle_{\iota\to o}\ X)]^T \qquad \mathcal{C}_2 : [p_{(\iota\to o)\to o}\ (\lambda X_\iota.\ circle_{\iota\to o}\ X \land red_{\iota\to o}\ X)]^F$$

where $p_{(\iota\to o)\to o}$ is a new Skolem constant. Leo can easily solve this problem by first resolving between $\mathcal{C}_1$ and $\mathcal{C}_2$, decomposing $p_{(\iota\to o)\to o}$ in the resulting unification constraint and employing the functional extensionality principle, thereby deriving the constraint

$$[red_{\iota\to o}\ s \land circle_{\iota\to o}\ s = circle_{\iota\to o}\ s \land red_{\iota\to o}\ s]^F$$

Now Leo employs the Boolean extensionality principle, i.e., replaces $=$ by $\equiv$, and performs a recursive call to the overall refutation search, thereby deriving the empty clause in a quite straightforward way. To come to the point, the sketched goal-directed refutation is not possible when employing syntactical term indexing techniques within the computation of resolution partners as the first, essential resolution step between $\mathcal{C}_1$ and $\mathcal{C}_2$ would not even be suggested. The problem is that syntactical term indexing lacks the extensionality principles, and for the same reason LEO can generally not employ term indexing techniques in unification.

We want to point out that the prototypical implementation of LEO is not a complete refutation procedure and in Chapter 7 we will discuss the different sources of incompleteness in detail. But we will also sketch possible solutions to the incompleteness problems.

A case study, which is illustrated in chapter 7, has demonstrated that LEO is indeed capable of solving simple theorems about sets, which require the application of the extensionality principles. In this experiment LEO could solve 95 of 97 theorems from the article *Boolean Properties of Sets* [TS89] of the Mizar library [Rud92]. This experiment also showed that on this domain LEO outperforms well known first-order theorem provers, which cannot exploit the expressiveness of classical type theory and encode this examples in the case study in Tar-ski Grothendieck set theory [Try89] (see the results of this case study at `http://www-irm.mathematik.hu-berlin.de/~ilf/miz2atp/mizstat.html`. At present the case study with LEO is continued with the slightly more challenging examples from the Mizar-article *Some Basic Properties of Sets* [Byl89], and without any modifications LEO can already solve about 40% of them.

In Chapter 7 we also sketch further aspects of LEO like its features as an interactive theorem prover or its integration to ΩMEGA [BCF$^+$97].

Chapter 2

Syntax and Semantics of Higher-Order Logic

2.1 Syntax and Preliminaries

In this section we introduce the preliminaries in a quite compact form.[1] Apart from the notational conventions most of the introduced concept are as defined as usual.

We start with a higher-order logic based on Church's simply typed lambda calculus [Chu40] and choose the set of base types $\mathcal{BT}$ to consist of the types ι and o, where o denotes the set of truth values and ι the set of individuals. The set of all types $\mathcal{T}$ is inductively defined over $\mathcal{BT}$ and the type constructor $\rightarrow$.

We define the order of types and λ-terms as in [SG89, Sny91].

We assume that our signature Σ contains a countably infinite set of variables $\mathcal{V}_\tau$ and constants $\mathcal{C}_\tau$ for every type $\tau \in \mathcal{T}$. Additionally we postulate the existence of the logical connectives $\neg_{o\rightarrow o}$, $\vee_{o\rightarrow o\rightarrow o}$, $\Pi_{(\alpha\rightarrow o)\rightarrow o}$ (in short Π^α) for every type $\alpha \in \mathcal{T}$ in Σ. A signature that contains additionally the logical connectives $=_{\alpha\rightarrow\alpha\rightarrow o}$ (in short $=^\alpha$) for all types $\alpha \in \mathcal{T}$ is noted by $\Sigma^=$. All the logical connectives in Σ or $\Sigma^=$ denote their intuitive semantical counterparts.

The remaining logical connectives are defined as abbreviations of the given ones: $\mathbf{A} \wedge \mathbf{B} := \neg(\neg\mathbf{A} \vee \neg\mathbf{B}), \mathbf{A} \Rightarrow \mathbf{B} := \neg\mathbf{A} \vee \mathbf{B}, \mathbf{A} \Leftrightarrow \mathbf{B} := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A}), \forall X_\alpha.\ \mathbf{A}_o := \Pi_{(\alpha\rightarrow o)\rightarrow o}(\lambda X_\alpha.\ \mathbf{A}), \exists X_\alpha.\ \mathbf{A}_o := \neg\forall X_\alpha.\ \neg\mathbf{A}_o$. All other constants are called *parameters*, since the argumentation in this paper is parametric in their choice[2].

Unlike stated otherwise, variables are printed as upper-case (e.g., X_α), constants as lower-case letters (e.g., c_α), and arbitrary terms appear as bold capital letters (e.g., $\mathbf{T}_\alpha$). If the type of a symbol or term is either not of importance or uniquely determined by the given context, we do not explicitly mention it.

We denote the set of all terms over a signature Σ ($\Sigma^=$) by $\mathit{wff}(\Sigma)$ (resp. $\mathit{wff}(\Sigma^=)$), and $\mathit{wff}_\alpha(\Sigma)$ (resp. $\mathit{wff}_\alpha(\Sigma^=)$) denotes the set of all Σ-terms of type α. Terms of type o are also called *propositions* or *formulae*, and closed propositions are also called *sentences*. The set of propositions is abbreviated as $\mathit{wff}_o(\Sigma)$ and the set of all sentences as $\mathit{cwff}_o(\Sigma)$. $\mathbf{A}$ is called

[1] We apologise that all the important defintions are introduced here at once, instead of introducing them when needed. The motivation is to provide a compact reference.

[2] In particular, we do not assume the existence of description or choice operators. For a detailed discussion of the semantic issues raised by the presence of these logical constants see [And72b].

atomic, if its $\beta\eta$-normal form (cf. below) does not have a logical connective at head position.

We will take the *order* of a formula to be the highest order of the type of any of its subterms, and the order of a set of formulae to be the maximum of the orders of its members.

In order to avoid confusion we clarify the meaning of the different equality symbols used in this paper. $=^{\alpha}\in\Sigma$ is the syntactic equality constant in our higher-order language. We will illustrate below that equality can also be defined in higher-order logic and we refer to this definition by $\doteq$. The intuitive semantical equality relations in $\mathcal{D}_{\alpha\rightarrow\alpha\rightarrow o}$ are denoted by q^{α}. For the meta-level argumentation we use $\equiv$ and $:=$ for definitions.

To ease readability we assume right-associativity for the type constructor $\rightarrow$ and left-associativity of function application: $\mathbf{A}_{\alpha\rightarrow\beta\rightarrow\gamma}\ \mathbf{B}_{\alpha}\ \mathbf{C}_{\beta} := ((\mathbf{A}_{\alpha\rightarrow\beta\rightarrow\gamma}\ \mathbf{B}_{\alpha})\ \mathbf{C}_{\beta})$. Furthermore, we sometimes abbreviate function applications by $h_{\alpha_1\rightarrow\dots\rightarrow\alpha_n\rightarrow\beta}\ \overline{\mathbf{U}^{n}_{\alpha_n}}$, which stands for $(\cdots(h_{\alpha_1\rightarrow\dots\rightarrow\alpha_n\rightarrow\beta}\ \mathbf{U}^{1}_{\alpha_1})\cdots\ \mathbf{U}^{n}_{\alpha_n})$. A dot ". " occurring in a λ-term stands for a left bracket whose mate is as far to the right as consistent with all other brackets and the construction of the term. We avoid brackets in every case where the construction of an expression is uniquely determined by the context.

The structural equality relation in our higher-order logic is induced by *$\beta\eta$-reduction*

$$(\lambda X.\ \mathbf{A})\ \mathbf{B} \longrightarrow_{\beta} [\mathbf{B}/X]\mathbf{A} \qquad\qquad (\lambda X.\ \mathbf{C}\ X) \longrightarrow_{\eta} \mathbf{C}$$

where X is not free in $\mathbf{C}$. It is well-known (c.f. [Bar84]), that the reduction relations β, η, and $\beta\eta$ are terminating and confluent, so that there are unique normal forms for each term $\mathbf{T}$ denoted by $\mathbf{T}\!\downarrow_{\beta\eta}$, $\mathbf{T}\!\downarrow_{\beta}$, $\mathbf{T}\!\downarrow_{\eta}$. The induced structural equality relations are denoted by $\equiv_{\beta\eta}$, $\equiv_{\beta}$, $\equiv_{\eta}$ (and $\equiv_{\alpha}$ for the equality relation induced by the renaming of bound variables). Another important normal form used in this paper is the *head-normal form* (which is unique only with respect to $\beta\eta$-equality): a term $\lambda\overline{X^{n}}.\ h\ \overline{\mathbf{U}^{m}}$ is in head-normal form, iff h is a variable or a constant. The head-normal form of a term $\mathbf{T}$ is denoted by $\mathbf{T}\!\downarrow_{h}$.

The definitions of *free* and *bound* variables, *substitutions* and the *application of substitution* are as usual (see [Bar84]). In this paper we denote the set of free variables of a term $\mathbf{T}$ (analogously for literals and clauses) by free($\mathbf{T}$). Whereas the usual application of a substitution $[\mathbf{A}/X]$ to $\mathbf{T}$ is denoted by $[\mathbf{A}/X]\mathbf{T}$, we denote with $\mathbf{T}_{[\mathbf{A}/X]}$ the combination of usual substitution with subsequent reduction of the resulting term (literal or clause) to head normal form.

We define *satisfiability*, *unsatisfiability*, and *validity* of a formula $\mathbf{F}$ or set of formulae Φ with respect to a model[3] $\mathcal{M}$ as usual.

When we speak of a *Skolem term s_{α} for a clause $\mathcal{C}$* and free($\mathcal{C}$) $\equiv \{X^{1}_{\alpha^{1}},\dots,X^{n}_{\alpha^{n}}\}$, then s_{α} is an abbreviation for the term $(f^{n}_{\alpha^{1}\rightarrow\dots\rightarrow\alpha^{n}\rightarrow\alpha}X^{1}\cdots X^{n})$, where f is a new constant from $\mathcal{C}_{\alpha^{1}\rightarrow\dots\rightarrow\alpha^{n}\rightarrow\alpha}$ and n specifies the number of necessary arguments for f. The latter is important as a naive treatment of Skolemisation results in a calculus that is not sound with respect to Henkin models, since Skolem functions are special choice functions[4], which are not guaranteed to exist in Henkin models. A solution due to [Mil83] is to associate with each Skolem constant the minimum number of arguments the constant has to be applied to. Skolemisation becomes sound, if any Skolem function f^{n} only occurs in a *Skolem term*, i.e., a formula $\mathbf{S}\equiv f^{n}\overline{\mathbf{A}^{n}}$, where none of the $\mathbf{A}^{i}$ contains a variable that is bound outside of S. Thus the Skolem terms only serve as descriptions of the existential witnesses and never appear as choice functions.

[3] The different notions of models, e.g. Henkin models and standard models will be introduced in Sections 2.2–2.7.

[4] They choose an existential witness from the set of possible witnesses for an existential formula.

Let $\alpha := (\overline{\beta^l} \to \gamma)$ and let h be a constant or variable of type $(\overline{\delta_m} \to \gamma)$ in Σ, then $\mathbf{G} := \lambda \overline{X^l_{\beta^l}}.\ h\ \overline{\mathbf{V}^m}$ is called a *partial binding of type α and head h* (see also [SG89, Sny91]), if $\mathbf{V}^i \equiv H^i\ \overline{X^l_{\beta_l}}$ and the H^i are new variables of types $\overline{\beta^l} \to \delta^i$. It is easy to see that general bindings indeed have the type and head claimed in the name and are most general in the class of all such terms.

Partial bindings, where the head is a bound variable $X^j_{\beta_j}$ are called *projection bindings* (we write them as $\mathcal{G}^j_\alpha$) and *imitation bindings* (written $\mathcal{G}^h_\alpha$) otherwise. Since we need both imitation and projection bindings for higher-order unification, we collect them in the set of *approximating bindings for h and α* ($\mathcal{AB}^h_\alpha := \{\mathcal{G}^h_\alpha\} \cup \{\mathcal{G}^j_\alpha \mid j \leq l\}$).

For a general introduction to higher-order unification we refer to [SG89, Sny91].

The calculi in this paper are defined on *clauses*, which are disjunctions of *literals* (e.g., $[q_{\alpha\to o} X_\alpha]^T \vee [p_{\alpha\to o} X_\alpha]^F \vee [c_\alpha = X_\alpha]^F$). For literals we differentiate between *pre-literals* and *proper literals*. A *pre-literal* consists of a proposition $\mathbf{N}_o$ in head-normal form (*atom*) and a *polarity* T or F which states whether this literal is positive or negative. We call a literal *proper*, iff it contains no logical constant beside $=$ at head position.

We further differentiate between *positive literals*, *negative literals*, and *unification constraints*. Unification constraints refer to negative literals with primitive equations as atoms. For the calculi discussed in this paper unification constraints are handled in one of the following ways:

- As special negative literals with a special head symbol $=^\alpha \notin \Sigma$, i.e., $=^\alpha$ is not a logical constant. By special literal we mean that this literal is treated as a unification constraint only, such that no rule but the unification rules are allowed to operate on them. This will be the case in the extensional higher-order resolution calculus $\mathcal{ER}$ discussed in Chapter 4. E.g., consider the clauses $\mathcal{C}_1 : [H_{\iota\to o} a_\iota]^T$ and $\mathcal{C}_2 : [X = \mathbf{A}]^F$. Clause $\mathcal{C}_2$ consists only of a unification constraint, which is encoded as a negative primitive equation. This literal is not treated as usual negative literal in $\mathcal{ER}$, such that resolving between $\mathcal{C}_1$ and $\mathcal{C}_2$ is forbidden and positive primitive literals like $[f = g]^T$ are not allowed.

- As special negative literals with a logical constant $=^\alpha \in \Sigma$ as head symbol. Special negative literal means that despite the fact that $=^\alpha$ is a logical constant provided by the signature, no rule other than the unification rules are allowed to operate on them. This will be the case in the extensional higher-order paramodulation calculus $\mathcal{EP}$ in Chapter 5. Now positive primitive equations like $[f = g]^T$ are allowed (since $=^\alpha \in \Sigma$), but resolution (e.g., between $\mathcal{C}_1$ and $\mathcal{C}_2$ above), factorisation, etc. on unification constraints is still forbidden.

- As ordinary negative literals with the logical constant $=^\alpha \in \Sigma$ as head symbol. In this case all rules, e.g., the resolution and factorisation rules, are allowed to operate on these literals. This holds for the extensional higher-order paramodulation calculus $\mathcal{ERUE}$ in Chapter 6. Now resolution beetween clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ from above is allowed.

A clause $\mathcal{C}$ is called a *proper clause*, iff it is in *clause normal form*, i.e., if all literals of $\mathcal{C}$ are proper. Otherwise we call $\mathcal{C}$ a *pre-clause*. Similarly a clause $\mathcal{C}$ is in *head-normal form*, iff all its literals are.

An unification constraint $U := [X_\alpha = \mathbf{N}_\alpha]^F$ or $U := [\mathbf{N}_\alpha = X_\alpha]^F$ is called *solved*, iff $X_\alpha \notin \text{free}(\mathbf{N}_\alpha)$. In this case X is called the *solved variable* of U. Furthermore, a unification constraint in *head-normal form* $[(H\ \overline{\mathbf{U}^n}) = (G\ \overline{\mathbf{V}^m})]^F$ for $n, m \geq 1$ is called a *flex-flex constraint* (*flex-rigid constraint*), iff H and G are variables (either H or G is a variable).

We define a clause $\mathcal{C}$ to be *empty* (denoted by $\Box$), iff $\mathcal{C}$ consists only of *flex-flex constraints*. As it is well known that any set of *flex-flex constraints* is unifiable, such that they evaluate to F, we know that $\Box$ is unsatisfiable (cf. [Hue75]).

Let $\mathcal{C} := L^1 \vee \cdots \vee L^n \vee U^1 \vee \cdots \vee U^m$ be a clause with unification constraints $U^1, \ldots, U^m$ $(1 \leq m)$. Then a disjunction $U^{i_1} \vee \cdots \vee U^{i_k}$ $(i_j \in \{1, \cdots, m\}; 1 \leq j \leq k)$ of solved unification constraints occurring in $\mathcal{C}$ is called *solved for* $\mathcal{C}$, iff for every $U^{i_j} (1 \leq j \leq k)$ holds: the solved variable of U^{i_j} does not occur free in any of the U^{i_l} for $l \neq j; 1 \leq l \leq k$.

Given a calculus R, i.e., a set of rules $r_n (n \geq 0)$ defined on clauses, we define the following derivation relation: $\Phi \vdash^{r_n} \mathcal{C}$ $(\mathcal{C}' \vdash^{r_n} \mathcal{C})$, iff $\mathcal{C}$ is the result of one application of rule $r_n \in R$ to premise clauses $\mathcal{C}'_i \in \Phi$ (to premise clause $\mathcal{C}'$). Multiple step derivations within a calculus R, e.g., $\Phi_1 \vdash^{r_{i_1}} \ldots \vdash^{r_{i_k}} \Phi_k$ (or $\mathcal{C}_1 \vdash^{r_{i_1}} \ldots \vdash^{r_{i_k}} \mathcal{C}_k$) where $k \geq 0$ and $r_{i_j} \in R$ for $1 \leq k \leq n$, are abbreviated by $\Phi_1 \vdash_R \Phi_k$ (or $\mathcal{C}_1 \vdash_R \mathcal{C}_k$). Derivations in a calculus R of exactly n steps are symbolised by $\vdash^n_R$.

A rule r is called *admissible* in one of our resolution calculi R, iff adding rule r to R does not increase the set of refutable formulae. Furthermore, a rule r is called *derivable* in R, iff each application of rule r can be replaced by an alternative derivation in R.

We shall now introduce a variety of semantical constructions for classical type theory motivated by the different of roles equality and extensionality. We will start out by defining Σ-structures (and as an intermediate step pre-Σ-structures) as algebraic semantics for the simply typed λ-calculus and then specialise them to our notions of models by requiring a special treatment of the propositional formulae.

2.2 Pre-Σ-Structures

Definition 2.1 (Pre-Σ-Structure). A collection $\mathcal{D} := \mathcal{D}_\mathcal{T} := \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ of sets $\mathcal{D}_\alpha$, indexed by the set $\mathcal{T}$ of types, is called a *typed collection (of sets)*. Let $\mathcal{D}_\mathcal{T}$ and $\mathcal{E}_\mathcal{T}$ be typed collections, then a collection $\mathcal{I} := \{\mathcal{I}^\alpha : \mathcal{D}_\alpha \to \mathcal{E}_\alpha \mid \alpha \in \mathcal{T}\}$ of mappings is called a *typed mapping* $\mathcal{I}: \mathcal{D}_\mathcal{T} \longrightarrow \mathcal{E}_\mathcal{T}$. We call the triple $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ a *pre-Σ-structure*, iff $\mathcal{D} \equiv \mathcal{D}_\mathcal{T}$ is a typed collection of sets and

$$@ := \{@^{\alpha\beta} : \mathcal{D}_{\alpha\to\beta} \times \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta \mid \alpha, \beta \in \mathcal{T}\}$$

and $\mathcal{I}: \Sigma \longrightarrow \mathcal{D}$ are typed total functions.

The collection $\mathcal{D}$ is called the *frame of* $\mathcal{A}$, the set $\mathcal{D}_\alpha$ the *universe of type* α, the function @ the *application operator*, and the function $\mathcal{I}$ the *interpretation of constants*.

We call a pre-Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ *functional*, iff the following statement holds for all $\mathsf{f}, \mathsf{g} \in \mathcal{D}_{\alpha\to\beta}$: $\mathsf{f} \equiv \mathsf{g}$, if for all $\mathsf{a} \in \mathcal{D}_\alpha$ we have that $\mathsf{f}@\mathsf{a} \equiv \mathsf{g}@\mathsf{a}$. Note that functionality is a restriction on the function universes only.

Remark 2.2. The application operator @ in a pre-Σ-structure is an abstract version of function application. It is no restriction to exclusively use a binary application operator, which corresponds to unary function application, since we can define higher-arity application operators from the binary one by setting ("Currying")

$$\mathsf{f}@(\mathsf{a}^1, \ldots, \mathsf{a}^n) := (\ldots(\mathsf{f}@\mathsf{a}^1)\ldots@\mathsf{a}^n)$$

Example 2.3. If we define $\mathbf{A}@\mathbf{B} := (\mathbf{A}\ \mathbf{B})$ for $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma)$, then $@\colon \mathit{wff}_{\alpha\to\beta}(\Sigma) \times \mathit{wff}_\alpha(\Sigma) \longrightarrow \mathit{wff}_\beta(\Sigma)$ is a total function. Thus $(\mathit{wff}(\Sigma), @, \mathrm{Id}_\Sigma)$ is a pre-Σ-structure. The intuition behind this example is that we can think of the formula $\mathbf{A} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ as a function $\mathbf{A}\colon \mathit{wff}_\alpha(\Sigma) \longrightarrow \mathit{wff}_\beta(\Sigma)$ that maps $\mathbf{B}$ to $(\mathbf{A}\ \mathbf{B})$.

Analogously, we can define the pre-Σ-structure $(\mathit{cwff}(\Sigma), @, \mathrm{Id}_\Sigma)$ of closed formulae.

Example 2.4. The following is a (trivial) example for a functional pre-Σ-structure:

1. $(\{\mathsf{a}\} \times \mathcal{T}, @^{\mathsf{a}}, \mathcal{I}^{\mathsf{a}})$, where $\mathsf{a}@^{\mathsf{a}}\mathsf{a} \equiv \mathsf{a}$ and $\mathcal{I}^{\mathsf{a}}(c) \equiv \mathsf{a}$ for all constants $c \in \Sigma$, is called the **singleton pre-Σ-structure.**

Definition 2.5 (Σ-Homomorphism). Let $\mathcal{A} := (\mathcal{D}, @^{\mathcal{A}}, \mathcal{I})$ and $\mathcal{B} := (\mathcal{E}, @^{\mathcal{B}}, \mathcal{J})$ be pre-Σ-structures. A *Σ-homomorphism* is a typed mapping $\kappa\colon \mathcal{D} \longrightarrow \mathcal{E}$ such that

1. $\kappa \circ \mathcal{I} \equiv \mathcal{J}$.
2. For all types $\alpha, \beta \in \mathcal{T}$, all $\mathsf{f} \in \mathcal{D}_{\alpha\to\beta}$, and $\mathsf{g} \in \mathcal{D}_\alpha$ we have: $\kappa(\mathsf{f})@^{\mathcal{B}}\kappa(\mathsf{g}) \equiv \kappa(\mathsf{f}@^{\mathcal{A}}\mathsf{g})$.

The most important method for constructing Σ-structures with given properties in this thesis is well-known for algebraic structures and consists in building a suitable Σ-congruence and passing it to the quotient structure. We will now develop the formal basis for it.

Definition 2.6 (Σ-Congruence). Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure, then a typed equivalence relation $\sim$ is called a *Σ-congruence on $\mathcal{A}$*, iff $\mathsf{f} \sim \mathsf{f}' \in \mathcal{D}_{\alpha\to\beta}$ and $\mathsf{g} \sim \mathsf{g}' \in \mathcal{D}_\alpha$ imply $\mathsf{f}@\mathsf{g} \sim \mathsf{f}'@\mathsf{g}'$. Let $f \in \mathcal{D}_\alpha$, then the equivalence class of f modulo $\sim$, $[\![f]\!]_\sim$, is the set of all $g \in \mathcal{D}_\alpha$, such that $f \sim g$.

A Σ-congruence is called *functional*, iff for all types α, β and all $\mathsf{f}, \mathsf{g} \in \mathcal{D}_{\alpha\to\beta}$ the fact that $\mathsf{f}@\mathsf{a} \sim \mathsf{g}@\mathsf{a}$ holds for all $\mathsf{a} \in \mathcal{D}_\beta$ implies $\mathsf{f} \sim \mathsf{g}$. Note that, since $\sim$ is a congruence, we also have the other direction, so we have

$$\mathsf{f}@\mathsf{a} \sim \mathsf{g}@\mathsf{a} \text{ for all } \mathsf{a} \in \mathcal{D}_\beta, \text{ iff } \mathsf{f} \sim \mathsf{g}$$

Lemma 2.7. *The β and $\beta\eta$ equality relations $\equiv_\beta$ and $\equiv_{\beta\eta}$ are congruences on the pre-Σ-structures $\mathit{wff}(\Sigma)$ and $\mathit{cwff}(\Sigma)$ by definition. Moreover, $\beta\eta$-equality is functional on $\mathit{wff}(\Sigma)$ and $\mathit{cwff}(\Sigma)$.*

Proof: The congruence properties are a direct consequence of the fact that $\beta\eta$ reduction rules are defined to act on sub-term positions. We will establish functionality of $\equiv_{\beta\eta}$ on $\mathit{wff}(\Sigma)$ first and then use this to obtain the assertion for closed formulae.

Let $(\mathbf{A}_{\gamma\to\alpha}\ \mathbf{C}_\gamma) {\equiv_{\beta\eta}} (\mathbf{B}_{\gamma\to\alpha}\ \mathbf{C})$ for all $\mathbf{C}$, then in particular, for any variable $X \in \mathcal{V}_\gamma$ that is not free in $\mathbf{A}$ or $\mathbf{B}$, we have $(\mathbf{A}\ X) {\equiv_{\beta\eta}} (\mathbf{B}\ X)$ and $(\lambda X.\ \mathbf{A}\ X) {\equiv_{\beta\eta}} (\lambda X.\ \mathbf{B}\ X)$. By definition we have $(\mathbf{A} {\equiv_\eta} (\lambda X_\alpha.\ \mathbf{A}\ X) {\equiv_{\beta\eta}} (\lambda X_\alpha.\ \mathbf{B}\ X) {\equiv_\eta} \mathbf{B}$.

To show functionality of $\beta\eta$ on closed formulae, let $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$, such that $\mathbf{A} \not\equiv_{\beta\eta} \mathbf{B}$. Since $\beta\eta$ is functional on $\mathit{wff}(\Sigma)$, there must be a formula $\mathbf{C}$ with $(\mathbf{A}\ \mathbf{C}) \not\equiv_{\beta\eta} (\mathbf{B}\ \mathbf{C})$. Now let $\mathbf{C}'$ be a ground instance of $\mathbf{C}$, i.e., $\mathbf{C}' := \sigma(\mathbf{C})$, where σ is a closed substitution[5], then we have $(\mathbf{A}\ \mathbf{C}') \not\equiv_{\beta\eta} (\mathbf{B}\ \mathbf{C}')$. Thus we have shown that $\mathbf{A} \not\equiv_{\beta\eta} \mathbf{B}$ entails $\mathbf{A}\ \mathbf{C}' \not\equiv_{\beta\eta} \mathbf{B}\ \mathbf{C}'$, which gives us the assertion. □

[5] This has to exist, since we have assumed infinetely many constants for each type α in our signature Σ, i.e., all types are inhabited.

Definition 2.8 (Quotient Pre-Σ-Structure). Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure, $\sim$ a Σ-congruence on $\mathcal{A}$, $\mathcal{D}^{\sim}_{\alpha} := \{[\![\mathsf{f}]\!]_{\sim} \mid \mathsf{f} \in \mathcal{D}_{\alpha}\}$, and $\mathcal{I}^{\sim}(c_{\alpha}) := [\![\mathcal{I}(c_{\alpha})]\!]_{\sim}$ for all constants $c_{\alpha} \in \Sigma_{\alpha}$. Furthermore let $@^{\sim}$ be defined by $[\![\mathsf{f}]\!]_{\sim} @^{\sim} [\![\mathsf{a}]\!]_{\sim} := [\![\mathsf{f}@\mathsf{a}]\!]_{\sim}$. To see that this definition only depends on equivalence classes of $\sim$, consider $\mathsf{f}' \in [\![\mathsf{f}]\!]_{\sim}$ and $\mathsf{g}' \in [\![\mathsf{g}]\!]_{\sim}$, then $[\![\mathsf{f}@\mathsf{g}]\!]_{\sim} \equiv [\![\mathsf{f}'@\mathsf{g}]\!]_{\sim} \equiv [\![\mathsf{f}'@\mathsf{g}']\!]_{\sim} \equiv [\![\mathsf{f}@\mathsf{g}']\!]_{\sim}$. So $@^{\sim}$ is well-defined and total, thus $\mathcal{A}/_{\sim} := (\mathcal{D}^{\sim}, @^{\sim}, \mathcal{I}^{\sim})$ is also a pre-Σ-structure. We call $\mathcal{A}/_{\sim}$ the *quotient structure of $\mathcal{A}$ for the relation $\sim$* and the typed function $\pi_{\sim}\colon \mathcal{A} \longrightarrow \mathcal{A}/_{\sim}$ that maps f to $[\![\mathsf{f}]\!]_{\sim}$ its *canonical projection.*

This definition is justified by the following theorem.

Theorem 2.9. *Let $\mathcal{A}$ be a pre-Σ-structure and let $\sim$ be an Σ-congruence on $\mathcal{A}$, then the canonical projection $\pi_{\sim}$ is a surjective Σ-homomorphism. Furthermore, $\mathcal{A}/_{\sim}$ is functional, iff $\sim$ is functional.*

Proof: Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure. To convince ourselves that $\pi_{\sim}$ is indeed a surjective Σ-homomorphism, we note that $\pi_{\sim}$ is surjective by definition and $\mathcal{I}^{\sim} \equiv \pi_{\sim} \circ \mathcal{I}$. Now let $\mathsf{f} \in \mathcal{D}_{\beta\to\alpha}$, and $\mathsf{g} \in \mathrm{domain}(\mathsf{f}) \subseteq \mathcal{D}_{\beta}$, then $\mathsf{g}' \in [\![\mathsf{g}]\!]_{\sim}$ for all $\mathsf{g}' \in \mathrm{domain}(\mathsf{f})$ and therefore $[\![\mathsf{g}]\!]_{\sim} \equiv \pi_{\sim}(\mathsf{g}) \in \mathrm{domain}([\![\mathsf{f}]\!]_{\sim}) \equiv \mathrm{domain}(\pi_{\sim}(\mathsf{f}))$ and $\pi_{\sim}(\mathsf{f})@^{\sim}\pi_{\sim}(\mathsf{g}) \equiv [\![\mathsf{f}]\!]_{\sim} @^{\sim} [\![\mathsf{g}]\!]_{\sim} \equiv [\![\mathsf{f}@\mathsf{g}]\!]_{\sim} \equiv \pi_{\sim}(\mathsf{f}@\mathsf{g})$.

The quotient construction trivialises $\sim$ to (meta-)equality, so functionality of $\sim$ is equivalent to functionality of $\mathcal{A}$. Formally we have $[\![\mathsf{f}]\!]_{\sim} \equiv [\![\mathsf{g}]\!]_{\sim}$, iff $\mathsf{f} \sim \mathsf{g}$, iff $f@\mathsf{a} \sim \mathsf{g}@\mathsf{a}$, iff $[\![\mathsf{f}@\mathsf{a}]\!]_{\sim} \equiv [\![\mathsf{g}@\mathsf{a}]\!]_{\sim}$, iff $[\![\mathsf{f}]\!]_{\sim} @^{\sim} [\![\mathsf{a}]\!]_{\sim} \equiv [\![\mathsf{g}]\!]_{\sim} @^{\sim} [\![\mathsf{a}]\!]_{\sim}$ for all $\mathsf{a} \in \mathcal{D}_{\alpha}$ and thus for all $[\![\mathsf{a}]\!]_{\sim} \in \mathcal{D}^{\sim}_{\alpha}$. □

2.3 Σ-Structures

Σ-structures are pre-Σ-structures with a notion of evaluation for *wff*(Σ).

Definition 2.10 (Σ-Structure). Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure. A typed function $\varphi\colon \mathcal{V} \longrightarrow \mathcal{D}$ is called a *variable assignment into $\mathcal{A}$*. We call a total typed mapping[6] $\mathcal{E}\colon \mathcal{F}(\mathcal{V};\mathcal{D}) \times$ *wff*(Σ) $\longrightarrow \mathcal{D}$ an *evaluation function* for $\mathcal{A}$, iff for any assignment φ into $\mathcal{A}$, we have

1. $\mathcal{E}_{\varphi}|_{\Sigma} \equiv \mathcal{I}$ and $\mathcal{E}_{\varphi}|_{\mathcal{V}} \equiv \varphi$
2. $\mathcal{E}_{\varphi}$ is a Σ-homomorphism
3. $\mathcal{E}_{\varphi}(\mathbf{A}) \equiv \mathcal{E}_{\psi}(\mathbf{A})$, whenever φ and ψ coincide on free($\mathbf{A}$)
4. $\mathcal{E}_{\varphi}([\mathbf{B}/X]\mathbf{A}) \equiv \mathcal{E}_{\varphi,[\mathcal{E}_{\varphi}(\mathbf{B})/X]}(\mathbf{A})$

We call $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ a *Σ-structure*, iff $(\mathcal{D}, @, \mathcal{I})$ is a pre-Σ-structure and $\mathcal{E}$ is an evaluation function for $\mathcal{A}$. We call $\mathcal{E}_{\varphi}(\mathbf{A}_{\alpha}) \in \mathcal{D}_{\alpha}$ the *denotation of $\mathbf{A}_{\alpha}$ in $\mathcal{A}$ for φ.*

If $\mathbf{A}$ is a closed formula, then $\mathcal{E}_{\varphi}(\mathbf{A})$ is independent of φ, since free($\mathbf{A}$) $\equiv \emptyset$. In these cases we sometimes drop the reference from $\mathcal{E}_{\varphi}(\mathbf{A})$ and simply write $\mathcal{E}(\mathbf{A})$.

[6] We write $\mathcal{F}(\mathcal{V};\mathcal{D})$ for the set of functions $f\colon \mathcal{V} \to \mathcal{D}$

Example 2.11. The singleton pre-Σ-structure is a Σ-structure if we take $\mathcal{E}(\mathbf{A}) \equiv a$, where a is the (unique) member of $\mathcal{D}_\alpha$.

For a detailed discussion on the closure conditions needed for the function universes to be rich enough, we refer the reader to [And72a, And73].

Note that the pre-Σ-structure $\mathit{wff}(\Sigma)$ from 2.3 cannot be made into a Σ-structure by providing an evaluation function, since there is no formula $\mathbf{C} \equiv \mathcal{I}_\varphi(\lambda X_\alpha.\ \mathbf{B}) \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$, such that $\mathbf{C}@\mathbf{A} \equiv \mathbf{CA} \equiv \mathcal{I}_{\varphi,[\mathbf{A}/X]}(\mathbf{B})$. In particular, the "obvious" choice $\lambda X_\alpha.\ \mathbf{B}$ for $\mathbf{C}$ does not work, since $(\lambda X_\alpha.\ \mathbf{B})\mathbf{A} \not\equiv \mathcal{I}_{\varphi,[\mathbf{A}/X]}(\mathbf{B})$. In fact, if $\mathit{wff}(\Sigma)$ were a Σ-structure, β-equality would have to be valid in $\mathit{wff}(\Sigma)$ (cf. 2.17), which it is clearly not.

Definition 2.12 (Quotient Σ-Structure). Let $\mathcal{A} \equiv (\mathcal{D}, @, \mathcal{E})$ be a Σ-structure, $\sim$ a Σ-congruence on $\mathcal{A}$ and let $\mathcal{A}/_\sim := (\mathcal{D}^\sim, @^\sim, \mathcal{I}^\sim)$ be the quotient pre-Σ-structure of $\mathcal{A}$, where $\mathcal{I} := \mathcal{E}|_\Sigma$.

For any assignment ψ into $\mathcal{A}/_\sim$, there exists an assignment φ into $\mathcal{A}$ such that $\psi \equiv \pi_\sim \circ \varphi$, since $\pi_\sim$ is a surjective Σ-homomorphism. So we can define $\mathcal{E}^\sim_\varphi$ as $\pi_\sim \circ \mathcal{E}_\psi$, and call $\mathcal{A}/_\sim := (\mathcal{D}^\sim, @^\sim, \mathcal{E}^\sim)$ the *quotient Σ-structure* of $\mathcal{A}$ modulo $\sim$.

Theorem 2.13 (Quotient Σ-Structure). *Let $\mathcal{A}$ be a Σ-structure and let $\sim$ be a Σ-congruence on $\mathcal{A}$, then $\mathcal{A}/_\sim$ is a Σ-structure.*

Proof: We prove that $\mathcal{E}^\sim$ is a legal value function by verifying the conditions in 2.10: Let φ and ψ be assignments, such that $\psi \equiv \pi_\sim \circ \varphi$, then

1. $\mathcal{E}^\sim_\varphi|_\Sigma \equiv (\pi_\sim \circ \mathcal{E}_\psi)|_\Sigma \equiv \pi_\sim \circ \mathcal{E}_\psi|_\Sigma \equiv \pi_\sim \circ \mathcal{I} \equiv \mathcal{I}^\sim$ and
   $\mathcal{E}^\sim_\varphi|_\mathcal{V} \equiv (\pi_\sim \circ \mathcal{E}_\psi)|_\mathcal{V} \equiv \pi_\sim \circ \mathcal{E}_\psi|_\mathcal{V} \equiv \pi_\sim \circ \psi \equiv \varphi$

2. $\mathcal{E}^\sim_\varphi \equiv \pi_\sim \circ \mathcal{E}_\psi$ is a Σ-homomorphism, since $\pi_\sim$ and $\mathcal{E}_\psi$ are.

3. $\mathcal{E}^\sim_\varphi(\mathbf{A}) \equiv [\![\mathcal{E}_\psi(\mathbf{A})]\!]_\sim \equiv \left[\!\left[\mathcal{E}_{\psi'}(\mathbf{A})\right]\!\right]_\sim \equiv \mathcal{E}^\sim_{\varphi'}(\mathbf{A})$, iff φ and φ' coincide on free($\mathbf{A}$), since this entails that ψ and ψ' do too.

4. $\mathcal{E}^\sim_\varphi([\mathbf{B}/X]\mathbf{A}) \equiv [\![\mathcal{E}_\psi([\mathbf{B}/X]\mathbf{A})]\!]_\sim \equiv \left[\!\left[\mathcal{E}_{\psi,[\mathcal{E}_\psi(\mathbf{B})/X]}(\mathbf{A})\right]\!\right]_\sim \equiv \mathcal{E}^\sim_{\varphi,[\mathcal{E}^\sim_\varphi(\mathbf{B})/X]}(\mathbf{A})$, since $[\![\mathcal{E}_\psi(\mathbf{B})]\!]_\sim \equiv \mathcal{E}^\sim_\varphi(\mathbf{B})$ and therefore $\pi_\sim \circ \psi, [\mathcal{E}_\psi(\mathbf{B})/X] \equiv \varphi, [\mathcal{E}^\sim_\varphi(\mathbf{B})/X]$

□

Definition 2.14 (Term Structures for Σ). Let $\mathit{cwff}(\Sigma)\!\downarrow_\beta$ be the collection of closed well-formed formulae in β-normal form and $\mathbf{A}@^\beta\mathbf{B}$ be the β-normal form of $(\mathbf{A}\ \mathbf{B})$. For the definition of an evaluation function let φ be an assignment into $\mathit{cwff}(\Sigma)\!\downarrow_\beta$. Note that $\sigma := \varphi|_{\mathrm{free}(\mathbf{A})}$ is a substitution, since free($\mathbf{A}$) is finite. Thus, we can choose $\mathcal{E}^\beta_\varphi(\mathbf{A}) := \sigma(\mathbf{A})\!\downarrow_\beta$, where $\mathbf{A}\!\downarrow_\beta$ is the β-normal form of $\mathbf{A}$. We call $\mathcal{TS}(\Sigma)^\beta := (\mathit{cwff}(\Sigma)\!\downarrow_\beta, @^\beta, \mathcal{E}^\beta)$ the *β-term structure for* Σ.

Analogously, we can define $\mathcal{TS}(\Sigma)^{\beta\eta} := (\mathit{cwff}(\Sigma)\!\downarrow_{\beta\eta}, @^{\beta\eta}, \mathcal{E}^{\beta\eta})$ the *$\beta\eta$-term structure for* Σ.

The name *term structure* in the previous definition is justified by the following lemma.

Lemma 2.15. *$\mathcal{TS}(\Sigma)^\beta$ is a Σ-structure and $\mathcal{TS}(\Sigma)^{\beta\eta}$ is a functional Σ-structure.*

Proof: Note that constants are β-normal forms, therefore $\mathcal{TS}(\Sigma)^{\beta}$ is the quotient structure of *cwff*(Σ) for the congruence $\equiv_{\beta}$. As we have remarked in 2.11, *wff*(Σ) is not a Σ-structure, so we cannot use 2.13, but have to convince ourselves directly that $\mathcal{TS}(\Sigma)^{\beta}$ is a Σ-structure by verifying the conditions of 2.10. The first three are direct consequences of the definition of $\mathcal{E}^{\beta}$ as substitution application.

1. $\mathcal{E}^{\beta}_{\varphi}|_{\Sigma} \equiv \mathcal{I}^{\beta} \equiv \mathrm{Id}_{\Sigma}$ and $\mathcal{E}^{\beta}_{\varphi}|_{\mathcal{V}} \equiv \varphi$
2. $\mathcal{E}^{\beta}_{\varphi}$ is a Σ-homomorphism
3. $\mathcal{E}^{\beta}_{\varphi}(\mathbf{A}) \equiv \sigma(\mathbf{A}) \equiv \sigma'(\mathbf{A}) \equiv \mathcal{E}_{\varphi'}(\mathbf{A})$, iff φ and φ' coincide on free$(\mathbf{A})$
4. $\mathcal{E}^{\beta}_{\varphi}([\mathbf{B}/X]\mathbf{A}) \equiv \sigma([\mathbf{B}/X]\mathbf{A}) \equiv [\sigma(\mathbf{B})/X](\sigma'(\mathbf{A})) \equiv \sigma, [\sigma(\mathbf{B})/X]\mathbf{A} \equiv \mathcal{E}^{\beta}_{\varphi,[\mathcal{E}^{\beta}_{\varphi}(\mathbf{B})/X]}(\mathbf{A})$, where σ' is $\sigma, [X/X]$.

Since $\equiv_{\beta\eta}$ is a super-relation of $\equiv_{\beta}$, a similar argument shows that $\mathcal{TS}(\Sigma)^{\beta\eta}$ is a Σ-structure. Furthermore, $\equiv_{\beta\eta}$ is a functional Σ-congruence on *wff*(Σ) (cf. 2.7), so we know by 2.9 that $\mathcal{TS}(\Sigma)^{\beta\eta}$ is functional. □

Remark 2.16. Note that $\mathcal{TS}(\Sigma)^{\beta}$ is not a functional Σ-structure since, e.g., $(\lambda X_{\gamma}.\, Y_{\gamma\to\delta} X)@^{\beta}\mathbf{C}_{\gamma} \equiv \mathbf{Y}@^{\beta}\mathbf{C}$ for all $\mathbf{C}$ in $\mathcal{TS}_{\gamma}(\Sigma)^{\beta}$ but $(\lambda X.\, Y\ X) \not\equiv Y$.

In a general Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$, the constants are given a meaning by the interpretation function $\mathcal{I}\colon \Sigma \to \mathcal{D}$, and variables get their meaning by assignments $\varphi\colon \mathcal{V} \to \mathcal{D}$. Furthermore, the evaluation function has to respect instantiation as in first-order logic. This is enough to ensure soundness of β-equality. We do not have to show soundness of α-equality, since this is trivial as we have assumed alphabetic variants to be identical.

Lemma 2.17 (Soundness of β-equality). *Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-structure and φ an assignment into $\mathcal{A}$, then $\mathcal{E}_{\varphi}((\lambda X.\, \mathbf{A})\mathbf{B}) \equiv \mathcal{E}_{\varphi}([\mathbf{B}/X]\mathbf{A})$ provided that X is not bound in $\mathbf{A}$.*

Proof: By the definition of Σ-structures, we have $\mathcal{E}_{\varphi}((\lambda X.\, \mathbf{A})\mathbf{B}) \equiv \mathcal{E}_{\varphi}(\lambda X.\, \mathbf{A})@\mathcal{E}_{\varphi}(\mathbf{B}) \equiv \mathcal{E}_{\varphi,[\mathcal{E}_{\varphi}(\mathbf{B})/X]}(\mathbf{A}) \equiv \mathcal{E}_{\varphi}([\mathbf{B}/X]\mathbf{A})$ □

2.4 Functional Σ-structures

For functional Σ-structures, there is another way to define evaluation: Since well-formed formulae are inductively built up from constants and variables we can extend φ and $\mathcal{I}$ to a Σ-homomorphism on well-formed formulae.

Definition 2.18 (Homomorphic Extension). Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a functional pre-Σ-structure and let φ be an assignment into $\mathcal{A}$. Then the *homomorphic extension $\mathcal{I}_{\varphi}$ of φ to* *wff*(Σ) is inductively defined to be a typed partial function $\mathcal{I}_{\varphi}\colon$ *wff*$(\Sigma) \longrightarrow \mathcal{D}$, such that

1. $\mathcal{I}_{\varphi}(X) \equiv \varphi(X)$, if X is a variable,
2. $\mathcal{I}_{\varphi}(c) \equiv \mathcal{I}(c)$, if c is a constant,
3. $\mathcal{I}_{\varphi}(\mathbf{A}\ \mathbf{B}) \equiv \mathcal{I}_{\varphi}(\mathbf{A})@\mathcal{I}_{\varphi}(\mathbf{B})$,

4. $\mathcal{I}_\varphi(\lambda X_\alpha.\ \mathbf{B}_\beta)$ is the function in $\mathcal{D}_{\alpha\to\beta}$, such that $\mathcal{I}_\varphi(\lambda X_\alpha.\ \mathbf{B})@z := \mathcal{I}_{\varphi,[z/X]}(\mathbf{B})$. Note that this function is unique, since we have assumed $\mathcal{A}$ to be functional.

We have to assume that the universes of functions $\mathcal{D}_{\alpha\to\beta}$ are rich enough to contain a value for all $\mathbf{A}_{\alpha\to\beta} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ for this construction to yield a total function.

Lemma 2.19. *Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a functional pre-Σ-structure, then $\mathcal{E}\colon \varphi \mapsto \mathcal{I}_\varphi$ is an evaluation function for $\mathcal{A}$.*

Proof: To prove the assertion, we have to show the conditions of 2.10. The first one is trivially met by construction, the second is a direct consequence of the fact that $\mathcal{I}_\varphi \circ \mathrm{Id}_\Sigma \equiv \mathcal{I} \circ \mathrm{Id}_\Sigma \equiv \mathcal{I}$ on Σ.

For the third condition, we prove that the value of a function depends only on its free variables (by induction on the structure of $\mathbf{A}$). The only interesting case is the one, where $\mathbf{A}$ is an abstraction, since the assertion is trivial for constants and variables, and a simple consequence of the inductive hypothesis for applications. So let $\mathbf{A} := (\lambda X.\ \mathbf{B})$, then $\mathcal{I}_\varphi(\mathbf{A})@\mathsf{a} \equiv \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B}) \equiv \mathcal{I}_{\psi,[\mathsf{a}/X]}(\mathbf{B}) \equiv \mathcal{I}_\psi(\mathbf{A})@\mathsf{a}$ by inductive hypothesis, since $\varphi, [\mathsf{a}/X]$ and $\psi, [\mathsf{a}/X]$ coincide on the free variables of $\mathbf{B}$. Thus we obtain the assertion from the definition of $\mathcal{I}_\varphi$.

Finally, we prove the fourth condition by induction on the structure of $\mathbf{A}$. If $\mathbf{A}$ is a constant or variable, then the assertion is trivial. The case where $\mathbf{A}$ is the application $\mathbf{C}\ \mathbf{D}$ is entailed by the fact, that substitution and homomorphic extension are defined inductively on the structure of applications: We have

$$\begin{aligned}
\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}\ \mathbf{D})) &\equiv \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{C})@\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{D})\\
&\equiv \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{C})@\mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{D})\\
&\equiv \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{C}\ \mathbf{D})
\end{aligned}$$

If $\mathbf{A} \equiv (\lambda Y.\ \mathbf{D})$ and $\psi \equiv \varphi, [\mathsf{a}/Y]$, then

$$\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})@\mathsf{a} \equiv \mathcal{I}_\varphi(\lambda Y.\ [\mathbf{B}/X]\mathbf{D})@\mathsf{a} \equiv \mathcal{I}_\psi([\mathbf{B}/X]\mathbf{D}) \equiv \mathcal{I}_{\psi,[\mathcal{I}_\psi(\mathbf{B})/X]}(\mathbf{D})$$

by inductive hypothesis. Note that ψ and φ coincide on the free variables of $\mathbf{A}$, therefore by the third condition, which we have proven above, we have $\mathcal{I}_{\psi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{D}) \equiv \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\lambda Y.\ \mathbf{D})@\mathsf{a}$, which implies the assertion, since $\mathcal{A}$ is functional. □

In fact, for functional Σ-structures, the two notions of evaluation coincide:

Lemma 2.20 (Evaluation in functional Σ-Structures). *If $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ is a functional Σ-structure, then $\mathcal{E}_\varphi \equiv \mathcal{I}_\varphi$ for any assignment φ into $\mathcal{A}$.*

Proof: Let $\mathbf{A} \in \mathit{wff}(\Sigma)$, we prove the assertion by induction over the size of $\mathbf{A}$. The assertion is trivial, if $\mathbf{A}$ is a constant or variable, and a simple consequence of the inductive hypothesis, if $\mathbf{A}$ is an application. So let $\mathbf{A} := (\lambda X.\ \mathbf{B})$, furthermore let Y be a variable not in free($\mathbf{A}$) and $\psi := \varphi, [\mathsf{a}/Y]$. Then

$$\mathcal{E}_\varphi(\mathbf{A})@\mathsf{a} \equiv \mathcal{E}_\psi(\mathbf{A})@\mathsf{a} \equiv \mathcal{E}_\varphi(\mathbf{A})@\mathcal{E}_\psi(Y) \equiv \mathcal{E}_\psi(\mathbf{A}Y) \equiv \mathcal{E}_\psi([Y/X]\mathbf{B})$$

since β-equality is sound in Σ-structures. Now $[Y/X]\mathbf{B}$ is smaller than $\mathbf{A}$, so we can use the inductive hypothesis to obtain

$$\mathcal{E}_\varphi(\mathbf{A})@\mathsf{a} \equiv \mathcal{I}_\psi([Y/X]\mathbf{B}) \equiv \mathcal{I}_\psi(\mathbf{A}Y) \equiv \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_\psi(Y) \equiv \mathcal{I}_\varphi(\mathbf{A})@\mathsf{a}$$

which entails the assertion since $\mathcal{A}$ is functional. □

Lemma 2.21. *Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-structure and X be a variable that is not free in $\mathbf{A}$, then $\mathcal{E}_\varphi(\lambda X.\ \mathbf{A}X) \equiv \mathcal{E}_\varphi(\mathbf{A})$ for all assignments φ into $\mathcal{A}$.*

Proof: With 2.10.3 and the fact that X is not free in $\mathbf{A}$ we have

$$\mathcal{E}_\varphi(\lambda X.\ \mathbf{A}X)@\mathsf{a} \equiv \mathcal{E}_{\varphi,[\mathsf{a}/X]}(\mathbf{A})@\mathcal{E}_{\varphi,[\mathsf{a}/X]}(X) \ \equiv \mathcal{E}_\varphi(\mathbf{A})@\mathsf{a}$$

which implies the assertion $\mathcal{E}_\varphi(\lambda X.\ \mathbf{A}X) \equiv \mathcal{E}_\varphi(\mathbf{A})$, as $\mathcal{A}$ is functional. □

We now specialise the notion of Σ-structures to the classical general model semantics for $\Lambda^{\rightarrow}$.

Definition 2.22 (Σ-Algebra). A *pre-Σ-algebra* $\mathcal{A} := (\mathcal{D}, \mathcal{I})$ is a pre-Σ-structure $(\mathcal{D}, @, \mathcal{I})$, such that $\mathcal{D}_{\alpha\rightarrow\beta} \subseteq \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$ and $\mathsf{f}@\mathsf{a} \equiv \mathsf{f}(\mathsf{a})$. A pre-Σ-algebra is called *full*, iff $\mathcal{D}_{\alpha\rightarrow\beta} \equiv \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$. We call a pre-Σ-algebra a *Σ-algebra*, iff it is a Σ-structure.

Remark 2.23. Note that pre-Σ-algebras are functional, since they are defined as structures of mathematical functions. On the other hand, for any functional Σ-structure $\mathcal{A}$, we can define an isomorphic Σ-algebra $\mathcal{A}'$.

Proof: For a functional Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ we define a Σ-algebra $\mathcal{A}' := (\mathcal{D}', \mathcal{I}')$ and a bijective Σ-homomorphism $\kappa: \mathcal{A} \longrightarrow \mathcal{A}'$ by an induction on the type:

- $\mathcal{D}'_\alpha := \mathcal{D}_\alpha$ for all $\alpha \in \mathcal{BT}$ and $\kappa := \mathrm{Id}_{\mathcal{D}}$; obviously κ is bijective.
- $\mathcal{D}'_{\alpha\rightarrow\beta} := \kappa(\mathcal{D}_{\alpha\rightarrow\beta})$ and $\kappa(\mathsf{f}) := \kappa \circ (@\mathsf{f}) \circ \kappa^{-1}$ for $\mathsf{f} \in \mathcal{D}_{\alpha\rightarrow\beta}$. Note that with this construction κ is a homomorphism, since

  $$\kappa(\mathsf{f})(\kappa(\mathsf{a})) \equiv \kappa(\mathsf{f}@(\kappa^{-1}(\kappa(\mathsf{a})))) \equiv \kappa(\mathsf{f}@\mathsf{a})$$

  κ is surjective by construction and injective, since $\mathcal{A}$ is functional: If $\mathsf{f} \not\equiv \mathsf{g} \in \mathcal{D}_{\alpha\rightarrow\beta}$, then there is an $\mathsf{a} \in \mathcal{D}_\alpha$, such that $\mathsf{f}(\mathsf{a}) \not\equiv \mathsf{g}(\mathsf{a})$, in particular, we have

  $$\kappa(\mathsf{f}(\mathsf{a})) \equiv \kappa(\mathsf{f})@\kappa(\mathsf{a}) \not\equiv \kappa(\mathsf{f})@\kappa(\mathsf{a}) \equiv \kappa(\mathsf{g}(\mathsf{a}))$$

  since κ is injective on $\mathcal{D}_\beta$. Thus and therefore $\kappa(\mathsf{f}) \not\equiv \kappa(\mathsf{g})$, since $\kappa(\mathsf{f}), \kappa(\mathsf{g}) \in \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$

Now, we only have to choose $\mathcal{I}' := \kappa \circ \mathcal{I}$ to complete the construction of $\mathcal{A}'$. □

As a consequence, we can always consider functional Σ-structures as Σ-algebras.

2.5 Σ-Models

The semantic notions so far are independent of the set of base types. Now, we specialise these to obtain a notion of models by requiring specialised behaviour on the type o of truth values. For this we use the notion of a Σ-valuation, which intuitively gives a truth-value interpretation to the domain $\mathcal{D}_o$ of a Σ-structure, which is consistent with the intuitive interpretations of the logical constants. Since models are semantic entities that are constructed first of all to make a statement about the truth or falsity of a formula, the requirement that there exists a Σ-valuation is perhaps the most general condition under which one wants to speak of a model. Thus we will define our most general notion of semantics as Σ-structures that have Σ-valuations.

Definition 2.24 (Σ-Model). Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-structure, then a surjective total function $v\colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ such that

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

is called a *Σ-valuation for $\mathcal{A}$* and $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, v)$ is called a *Σ-model.* The class of all Σ-models is denoted by $\mathfrak{M}_\beta$.

We say that an assignment φ *satisfies* a formula $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ in $\mathcal{M}$ ($\mathcal{M} \models_\varphi \mathbf{A}$), iff $v(\mathcal{E}_\varphi(\mathbf{A})) \equiv \mathtt{T}$ and that $\mathbf{A}$ is *valid* in $\mathcal{M}$, iff $\mathcal{M} \models_\varphi \mathbf{A}$ for all assignments φ. Finally, we say that $\mathcal{M}$ is a *Σ-model* for a set $H \subseteq \mathit{wff}_o(\Sigma)$ ($\mathcal{M} \models H$) iff $\mathcal{M}$ satisfies all $\mathbf{A} \in H$.

Lemma 2.25 (Truth and Falsity in Σ-Models). *Let $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, v)$ be a Σ-model and φ an assignment. Furthermore let $\mathbf{T}_o := \mathbf{A}_o \vee \neg(\mathbf{A}_o)$ for some $\mathbf{A}_o \in \mathit{wff}_o$ and let $\mathbf{F}_o := \neg\mathbf{T}_o$. Then $v(\mathcal{E}_\varphi(\mathbf{T}_o)) \equiv \mathtt{T}$ and $v(\mathcal{E}_\varphi(\mathbf{F}_o)) \equiv \mathtt{F}$.*

Proof: We have $v(\mathcal{E}_\varphi(\mathbf{T}_o)) \equiv \mathtt{T}$ iff $v(\mathcal{E}_\varphi(\mathbf{A}_o \vee \neg(\mathbf{A}_o))) \equiv \mathtt{T}$. Evaluation shows that this statement is equivalent to $v(\mathcal{E}_\varphi(\mathbf{A})) \equiv \mathtt{T}$ or $v(\mathcal{E}_\varphi(\mathbf{A})) \equiv \mathtt{F}$, which is valid since $\varphi : \mathcal{V}_o \to \mathcal{D}_o$ and $v : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ are total functions.
Note further that $v(\mathcal{E}_\varphi(\mathbf{F}_o)) \equiv \mathtt{F}$ evaluates to $v(\mathcal{E}_\varphi(\mathbf{T}_o)) \equiv \mathtt{T}$, which we already know. □

Remark 2.26. We only constrain the functional behaviour of the values of the logical constants. In particular this does not fully specify these values, since

- $\mathcal{M}$ need not be functional,
- and there can be more than two truth values.

Definition 2.27 (Properties $\mathfrak{q}$, $\mathfrak{b}$, and $\mathfrak{f}$). Given a Σ-model $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, v)$, we say that $\mathcal{M}$ has *property*

$\mathfrak{f}$ iff $\mathcal{M}$ is functional,

$\mathfrak{q}$ iff $\mathcal{M}$ has property $\mathfrak{f}$ and for all $\alpha \in \mathcal{T}$ there is a function $\mathsf{q}^\alpha \in \mathcal{D}_{\alpha\to\alpha\to o}$, such that for all $\mathsf{a}, \mathsf{b} \in \mathcal{D}_\alpha$ holds $v(\mathsf{q}^\alpha@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$ iff $\mathsf{a} \equiv \mathsf{b}$,

$\mathfrak{b}$ iff $\mathcal{D}_o$ has at most two elements. Note that $\mathcal{D}_o$ must always have at least the two elements $\mathcal{E}_\varphi(\mathbf{T}_o)$ and $\mathcal{E}_\varphi(\mathbf{F}_o)$ by Lemma 2.25, so we can assume without loss of generality that $\mathcal{D}_o \equiv \{\mathcal{E}_\varphi(\mathbf{F}_o) \equiv \mathtt{F}, \mathcal{E}_\varphi(\mathbf{T}_o) \equiv \mathtt{T}\}$ and that v is the identity function.

Definition 2.28 (Specialised Model Classes). We define special classes of Σ-models depending on the validity of the properties $\mathfrak{f}$, $\mathfrak{q}$ and $\mathfrak{b}$. Thus we obtain the specialised classes of Σ-models $\mathfrak{M}_{\beta\mathfrak{q}}, \mathfrak{M}_{\beta\mathfrak{b}}, \mathfrak{M}_{\beta\mathfrak{f}}, \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ by requiring that the properties specified in the index are valid.

*Remark 2.29 (**Property** $\mathfrak{q}$).* The idea of property $\mathfrak{q}$ is to ensure for all types α that the intuitive equality relation $\mathfrak{q}_{\alpha\to\alpha\to o}$, i.e., a functional congruence relation, is contained in $\mathcal{D}_{\alpha\to\alpha\to o}$. This ensures the existence of unit sets in the domains $\mathcal{D}_{\alpha\to o}$ which in turn makes Leibniz equality the intended equality relation, as the membership in this unit sets can be used as a strong argument in order to distinguish between different elements of $\mathcal{D}_\alpha$. For a detailed discussion see [And72a].

Property $\mathfrak{q}$ as stated in [BK97a] is not correct. Whereas the motivation for the formulation there was the same as sketched above, this formulation does unfortunately not ensure functionality (although this was intended), which is needed in the proof of Lemma 2.35. We want to thank an unknown referee of the Journal of Symbol Logic for pointing to this problem.

*Remark 2.30 (**Property** $\mathfrak{q}$ **in Henkin models**).* As Peter Andrews has noted in [And72a], Leon Henkin unintendedly introduced $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ in [Hen50] instead of the class of Henkin models in the sense below. An element of $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ does not necessarily have property $\mathfrak{q}$ and as Andrews has shown in [And72a], a consequence is, that such an element may lack the principle of functional extensionality $\mathrm{EXT}_L^{\alpha\to\beta}$, which he corrected by introducing property $\mathfrak{q}$.

Definition 2.31 (Σ-Henkin models). A functional Σ-model is called a *Σ-Henkin model*, iff it has properties $\mathfrak{q}$ and $\mathfrak{b}$. The class of all Σ-Henkin models is denoted by $\mathfrak{H}$ or $\mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$. If furthermore, all domains $D_{\alpha\to\beta}$ are full then we call $\mathcal{H}$ a *Σ-standard model ($\mathfrak{ST}$).*

Now let us extend the notion of a quotient structure to Σ-models.

Definition 2.32 (Quotient Σ-model). Let $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ be a Σ-model, $\sim$ a congruence on the corresponding Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$, and $\mathcal{A}/_{\sim}$ be the quotient Σ-structure of $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ modulo $\sim$ as defined in 2.12.

If $\upsilon(\mathbf{A}) \equiv \upsilon(\mathbf{B})$ for all $\mathbf{A}, \mathbf{B} \in \mathit{wff}_o(\Sigma)$ with $\mathbf{A} \sim \mathbf{B}$, then $\sim$ is called a *congruence for $\mathcal{M}$*. Then $\mathcal{M}/_{\sim} := (\mathcal{D}^{\sim}, @^{\sim}, \mathcal{E}^{\sim}, \upsilon^{\sim})$ is called the *quotient Σ-model of $\mathcal{M}$ modulo $\sim$*, if $\upsilon^{\sim}([\![a]\!]_{\sim}) \equiv \upsilon(a)$ for all $a \in \mathcal{D}_o$.

Remark 2.33. Note the importance of the additional requirement for functional congruence relations stated in 2.32. Without this requirement the quotient Σ-models are not well-defined.

Lemma 2.34. *Let $\mathcal{M}$ be a Σ-model, $H \subseteq \mathit{wff}_o(\Sigma)$, and $\sim$ be a congruence for $\mathcal{M}$, then $\mathcal{M}/_{\sim} \models_{\varphi} H$, iff $\mathcal{M} \models_{\varphi} H$.*

Proof: Let $\mathbf{A}_o \in H$. We have $\upsilon^{\sim}(\mathcal{E}_{\varphi}^{\sim}(\mathbf{A}_o)) \equiv \upsilon^{\sim}([\![\mathcal{E}_{\varphi}(\mathbf{A}_o)]\!]_{\sim}) \equiv \upsilon(\mathcal{E}_{\varphi}(\mathbf{A}_o))$. □

2.6 Leibniz Equality

Definition 2.35 (Extensionality for Leibniz equality). We call the following formula schemata

$$\begin{array}{lcl}\mathrm{EXT}_L^{\alpha\to\beta} & := & \forall F_{\alpha\to\beta}.\ \forall G_{\alpha\to\beta}(\forall X_\beta.\ F\ X \doteq G\ X) \Rightarrow F \doteq^{\beta} G \\ \mathrm{EXT}_L^{o} & := & \forall A_o.\ \forall B_o.\ (A \Leftrightarrow B) \Leftrightarrow A \doteq^{o} B\end{array}$$

the *axioms of full extensionality for Leibniz equality*; we refer to the first as *axiom of functional extensionality* and to the latter formula as the *extensionality axiom for truth values.* Note that $\mathrm{EXT}_L^{\alpha\to\beta}$ specifies functionality of the relation denoted by the Leibniz formula $\doteq$. We will use the terms functionality and extensionality interchangeably.

Lemma 2.36 (Leibniz Equality in Σ-models). *Let $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ be a Σ-model and φ be an assignment.*

1. *If $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$, then $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B}) \equiv \mathtt{T}$.*
2. *If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{b}}$ and $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^o \mathbf{B})) \equiv \mathtt{T}$, then $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$.*
3. *If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$ and $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, then $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$.*

Proof: Let $\mathsf{a}, \mathsf{b} \in \mathcal{D}_\alpha$ and $\psi := \varphi, [\mathsf{a}/X], [\mathsf{b}/Y]$.

1. We show that $\upsilon(\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, if $\mathsf{a} \equiv \mathsf{b}$, which entails the assertion. By definition $\mathcal{E}_\varphi(\doteq^\alpha) \equiv \mathcal{E}_\varphi(\lambda X.\, \lambda Y.\, \forall P.\, P\, X \Rightarrow P\, Y)$ and thus $\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b} \equiv \mathcal{E}_\psi(\forall P.\, P\, X \Rightarrow P\, Y)$. Now let $\mathsf{r} \in \mathcal{D}_{\alpha\to o}$, then $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(P\, X)) \equiv \mathsf{r}@\mathsf{a} \equiv \mathtt{F}$ or $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(P\, Y)) \equiv \mathsf{r}@\mathsf{b} \equiv \mathsf{r}@\mathsf{a} \equiv \mathtt{T}$, since υ is total and $\mathsf{a} \equiv \mathsf{b}$. So we see that $\upsilon(\mathcal{E}_\varphi(\doteq)@\mathsf{a}@\mathsf{b}) \equiv \upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(P\, X \Rightarrow P\, Y)) \equiv \mathtt{T}$ for all $\mathsf{r} \in \mathcal{D}_{\alpha\to o}$, which yields the assertion.
2. First note that by property $\mathfrak{b}$ we have $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$ and υ is the identity function on $\mathcal{D}_o$. Let us assume that $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^o \mathbf{B})) \equiv \mathcal{E}_\psi(\forall P.\, P\, \mathbf{A} \Rightarrow P\, \mathbf{B}) \equiv \mathtt{T}$ but $\mathcal{E}_\varphi(\mathbf{A}) \not\equiv \mathcal{E}_\varphi(\mathbf{B})$, which means that either $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathtt{T}$ and $\mathcal{E}_\varphi(\mathbf{B}) \equiv \mathtt{F}$ or vice a versa. In the first case we choose a predicate $r := \mathcal{E}_\varphi(\lambda X_o.\, X_o)$ and get from the first assumption that $\mathcal{E}_{\varphi,[\mathsf{r}/P]}(P\, \mathbf{A}) \equiv \mathcal{E}_{\varphi,[\mathsf{r}/P]}(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{A}) \equiv \mathtt{F}$ or $\mathcal{E}_{\varphi,[\mathsf{r}/P]}(P\, \mathbf{B}) \equiv \mathcal{E}_{\varphi,[\mathsf{r}/P]}(\mathbf{B}) \equiv \mathcal{E}_\varphi(\mathbf{B}) \equiv \mathtt{T}$, which gives us the contradiction. Note that P does not occur free in $\mathbf{A}$ or $\mathbf{B}$ by definition of $\doteq$.

   The second case is analogous with $r := \mathcal{E}_\varphi(\lambda X_o.\, \neg X_o)$.
3. We show that if $\upsilon(\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, then $\mathsf{a} \equiv \mathsf{b}$, which entails the assertion. Suppose $\mathsf{a} \not\equiv \mathsf{b} \in \mathcal{D}_\alpha$ and $\mathsf{r} \equiv \mathsf{q}^\alpha@\mathsf{a}$, where $\mathsf{q}^\alpha \in \mathcal{D}_{\alpha\to\alpha\to o}$ is the function guaranteed by property $\mathfrak{q}$. We know that $\mathsf{q}^\alpha@\mathsf{a}@\mathsf{a} \equiv \mathtt{T}$ and $\mathsf{q}^\alpha@\mathsf{a}@\mathsf{b} \equiv \mathtt{F}$, since $\mathsf{a} \not\equiv \mathsf{b}$ by assumption. Hence $\upsilon(\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \upsilon(\mathcal{E}_\psi(\forall P.\, P\, X \Rightarrow P\, Y) \equiv \mathtt{F}$ for $\psi := \varphi, [\mathsf{a}/X], [\mathsf{b}/Y]$, since $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(P\, X \Rightarrow PY)) \equiv \mathtt{F}$, as $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(P\, X)) \equiv \mathsf{q}^\alpha@\mathsf{a}@\mathsf{a} \equiv \mathsf{r}@\mathsf{a} \equiv \mathtt{T}$ and $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(P\, Y)) \equiv \mathsf{q}^\alpha@\mathsf{a}@\mathsf{b} \equiv \mathsf{r}@\mathsf{b} \equiv \mathtt{F}$.

□

Lemma 2.37 (Extensionality in Σ-models).

1. *There exists a model $\mathcal{M} \in \mathfrak{M}_\beta$ which is not functional.*
2. *There exists a model $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ for which $\mathrm{EXT}_L^{\alpha\to\beta}$ is not valid.*
3. *There exists a model $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$ for which EXT_L^o is not valid.*
4. *$\mathrm{EXT}_L^{\alpha\to\beta}$ is valid in model $\mathcal{M}$, if $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$.*
5. *EXT_L^o is valid in model $\mathcal{M}$, if $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{b}}$.*

As a consequence the following table characterises the different properties of the introduced semantical structures. If a formula is valid for a certain semantical structure we use a '+' and a '−' otherwise. Each entry is further marked with a justification referring to one of the above statements.

valid in	$\mathfrak{M}_\beta/\mathfrak{M}_{\beta\mathfrak{f}}$	$\mathfrak{M}_{\beta\mathfrak{q}}$	$\mathfrak{M}_{\beta\mathfrak{b}}/\mathfrak{M}_{\beta\mathfrak{fb}}$	$\mathfrak{M}_{\beta\mathfrak{qb}}$
$\mathrm{EXT}_L^{\alpha\to\beta}$	$-_{(2)}$	$+_{(4)}$	$-_{(2)}$	$+_{(4)}$
EXT_L^{o}	$-_{(3)}$	$-_{(3)}$	$+_{(5)}$	$+_{(5)}$

Proof: Let v be a Σ-valuation and let $\mathcal{M}$ be a Σ-model based on the β-termstructure $\mathcal{TS}(\Sigma)^\beta$ for Σ, i.e., $\mathcal{M} := (\mathit{cwff}(\Sigma){\downarrow}_\beta, @^\beta, \mathcal{E}^\beta, v)$. Note that $\mathcal{M}$ need not to be functional by Remark 2.16 and 2.26.

For the proof of 2. we refer to [And72a], where Andrews constructs a functional Σ-Model (actually a $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{fb}}$) that lacks the principle of functional extensionality of Leibniz equality.

For 3. note that EXT_L^o can only be valid if $\mathcal{D}_o \equiv \{o, \iota\}$, which is not required for $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$. For a concrete example of a $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$, which lacks EXT_L^o, see 3.29 (for $\mathfrak{Acc}_{\beta\mathfrak{q}}$).

Next we consider 4.: Let $\psi := \varphi, [\mathsf{f}/F], [\mathsf{g}/G]$. From $\mathcal{V}_\psi(\forall A_\alpha.\ F\ A \doteq G\ A) \equiv \mathtt{T}$ we get that for all $\mathsf{a} \in \mathcal{D}_\alpha$ $\mathcal{V}_{\psi,[\mathsf{a}/A]}(F\ A \doteq G\ A) \equiv \mathtt{T}$. By lemma 2.36(3) we can conclude that $\mathcal{E}_{\psi,[\mathsf{a}/A]}(F\ A) \equiv \mathcal{E}_{\psi,[\mathsf{a}/A]}(G\ A)$ for all $\mathsf{a} \in \mathcal{D}_\alpha$ and hence $\mathcal{E}_{\psi,[\mathsf{a}/A]}(F)@\mathcal{E}_{\psi,[\mathsf{a}/A]}(A) \equiv \mathcal{E}_{\psi,[\mathsf{a}/A]}(G)@\mathcal{E}_{\psi,[\mathsf{a}/A]}(A)$ for all $\mathsf{a} \in \mathcal{D}_\alpha$. By definition of property $\mathfrak{q}$, which includes property $\mathfrak{f}$ and thus ensures functionality, we get $\mathcal{E}_\psi(F) \equiv \mathcal{E}_\psi(G)$. This finally gives us that $\mathcal{V}_\psi(F \doteq^{\alpha\to\beta} G) \equiv \mathtt{T}$ with lemma 2.36(1).

And finally in 5. we have that for all $\mathsf{a}, \mathsf{b} \in \mathcal{D}_o$ and all assignments φ: $v(\mathcal{E}_{\varphi,[\mathsf{a}/A][b/B]}(A \Leftrightarrow B)) \equiv \mathtt{T}$, iff $v(\mathcal{E}_{\varphi,[\mathsf{a}/A][b/B]}(A)) \equiv v(\mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(B))$. From $\mathfrak{b}$ we further know that v is the identity function and hence this statement is valid, iff $\mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(A) \equiv \mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(B)$. For the left to right direction of our statement we can now apply lemma 2.36(1) and in the right to left direction the assertion follows with 2.36(2). □

Next we discuss the role of Leibniz equality within the different semantic structures.

Theorem 2.38 (Properties of Leibniz Equality). *Let $\mathcal{M}$ be a Σ-model. For all assignments φ and all terms $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathit{wff}_\alpha(\Sigma)$ and $\mathbf{F}, \mathbf{G} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ we have:*

$\mathfrak{M}_\beta$ *If $\mathcal{M} \in \mathfrak{M}_\beta$, then $\mathcal{E}_\varphi(\doteq^\alpha)$ is an equivalence relation on $\mathcal{D}_\alpha$ with respect to v. In particular:*

re: $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{A})) \equiv \mathtt{T}$.

sy: *If $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, then $v(\mathcal{E}_\varphi(\mathbf{B} \doteq^\alpha \mathbf{A})) \equiv \mathtt{T}$.*

tr: *If $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$ and $v(\mathcal{E}_\varphi(\mathbf{B} \doteq^\alpha \mathbf{C})) \equiv \mathtt{T}$, then $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{C})) \equiv \mathtt{T}$.*

$\mathfrak{M}_{\beta\mathfrak{f}}$ *If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{f}}$, then $\mathcal{E}_\varphi(\doteq^\alpha)$ is a congruence relation on $\mathcal{D}_\alpha$ with respect to v. In particular:*

co: *If $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, then $v(\mathcal{E}_\varphi(\mathbf{F}\ \mathbf{A} \doteq^\beta \mathbf{F}\ \mathbf{B})) \equiv \mathtt{T}$.*

$\mathfrak{M}_{\beta\mathfrak{q}}$ *If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$, then $\mathcal{E}_\varphi(\doteq^\alpha)$ is a functional congruence relation on $\mathcal{D}_\alpha$ with respect to v. In particular:*

fu: $v(\mathcal{E}_\varphi(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})) \equiv \mathtt{T}$, *if* $v(\mathcal{E}_\varphi(\mathbf{F}\ \mathbf{A} \doteq^\beta \mathbf{F}\ \mathbf{A})) \equiv \mathtt{T}$ *for all* $\mathbf{A} \in \mathit{wff}_\alpha$.

$\mathfrak{M}_{\beta\mathfrak{qb}}$ *If $\mathcal{M} \in \mathfrak{H}$, then $\mathcal{E}_\varphi(\doteq^\alpha)$ is the equality relation on $\mathcal{D}_\alpha$.*

Proof:

$\mathfrak{M}_\beta$ **re** $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{A})) \equiv \mathtt{T}$, iff for all $\mathsf{p} \in \mathcal{D}_{\alpha\to o}$ we have that $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ \mathbf{A})) \equiv \mathtt{F}$ or $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ \mathbf{A})) \equiv \mathtt{T}$, which is obvious since v is total and surjective.

sy Suppose $v(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, but $v(\mathcal{E}_\varphi(\mathbf{B} \doteq^\alpha \mathbf{A})) \equiv \mathtt{F}$. From the latter we get that $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ \mathbf{B})) \equiv \mathtt{T}$ and $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ \mathbf{A})) \equiv \mathtt{F}$ for some $\mathsf{p} \in \mathcal{D}_{\alpha\to o}$. Without loss of generality, let $\mathsf{p} := \mathcal{E}_\varphi(V)$ for a fresh variable $V \in \Sigma_{\alpha\to o}$. From the former assumption we know that for all $\mathsf{q} \in \mathcal{D}_{\alpha\to o}$ holds $v(\mathcal{E}_{\varphi,[\mathsf{q}/P]}(P\ \mathbf{A})) \equiv \mathtt{F}$ or $v(\mathcal{E}_{\varphi,[\mathsf{q}/P]}(P\ \mathbf{B})) \equiv \mathtt{T}$ and hence $v(\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\lambda X.\ VX)/P]}(P\ \mathbf{A})) \equiv \mathtt{F}$ or $v(\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\lambda X.\ VX)/P]}(P\ \mathbf{B})) \equiv \mathtt{T}$ which is equivalent with $v(\mathcal{E}_{\varphi,[p/P]}(P\ \mathbf{A})) \equiv \mathtt{T}$ or $v(\mathcal{E}_{\varphi,[p/P]}(P\ \mathbf{B})) \equiv \mathtt{F}$ and contradicts the latter assumption.

tr Similar to **sy**.

$\mathfrak{M}_{\beta\mathfrak{f}}$ **co** Suppose $v(\mathcal{E}_\varphi(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})) \equiv \mathtt{T}$, but $v(\mathcal{E}_\varphi(\mathbf{F}\ \mathbf{A} \doteq^\alpha \mathbf{G}\ \mathbf{A})) \equiv \mathtt{F}$. From the latter we get that $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ (\mathbf{F}\ \mathbf{A}))) \equiv \mathtt{T}$ and $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ (\mathbf{G}\ \mathbf{A}))) \equiv \mathtt{F}$ for some $\mathsf{p} \in \mathcal{D}_{\alpha\to o}$. Without loss of generality let $\mathsf{p} := \mathcal{E}_\varphi(V)$ for a fresh variable $V \in \mathcal{V}_{\alpha\to o}$. From the former assumption we know that for all $\mathsf{q} \in \mathcal{D}_{\alpha\to o}$ holds $v(\mathcal{E}_{\varphi,[\mathsf{q}/Q]}(Q\ \mathbf{F})) \equiv \mathtt{F}$ or $v(\mathcal{E}_{\varphi,[\mathsf{q}/Q]}(Q\ \mathbf{G})) \equiv \mathtt{T}$ and hence $v(\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\lambda X.\ V\ (X\ A))/P]}(P\ \mathbf{F})) \equiv \mathtt{F}$ or $v(\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\lambda X.\ V\ (X\ A))/P]}(P\ \mathbf{F})) \equiv \mathtt{T}$, which is equivalent with $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ (\mathbf{F}\ \mathbf{A}))) \equiv \mathtt{F}$ or $v(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\ (\mathbf{G}\ \mathbf{A}))) \equiv \mathtt{T}$ and contradicts the former assumption.

$\mathfrak{M}_{\beta\mathfrak{q}}$ **fu** A direct consequence of lemma 2.37(4).

$\mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$ By property $\mathfrak{b}$ we know that v is the identity relation on $\mathcal{D}_o$ and thus we have that $\doteq$ denotes a relation for which the principles reflexivity, symmetry, transitivity, congruence and functionality hold. Hence $\doteq$ denotes the equality relation.

□

2.7 Primitive Equality

The situation of higher-order semantics becomes much simpler if we introduce equality as a primitive logical constant $=$ in Σ, which we will assume for the rest of this section. Since $=$ is logical, we have to specialise the notion of Σ-valuation (cf. 2.24) by requiring that $v(\mathcal{E}(=^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, iff $\mathsf{a} \equiv \mathsf{b}$. In this case, we call v a Σ-*valuation with equality*.

Furthermore, we say that a Σ-model $\mathcal{M}$ has *property* $\mathfrak{e}$, iff $\mathcal{M}$ has property $\mathfrak{f}$ and for all types α, $v(\mathcal{E}(=^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, iff $\mathsf{a} \equiv \mathsf{b}$.

Definition 2.39 (Henkin Model with Primitive Equality).
A (functional) Σ-model, which has property $\mathfrak{e}$ is called a (functional) Σ-*model with primitive equality* and a functional one with additional property $\mathfrak{b}$ is called a Σ-*Henkin model with primitive equality.* The class of all Σ-models with primitive equality are denoted by $\mathfrak{M}_{\beta\mathfrak{e}}$ and the class of all Σ-Henkin model with primitive equality by $\mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$.

*Remark 2.40 (**Property** $\mathfrak{e}$ **implies** $\mathfrak{q}$).* It is easy to see that property $\mathfrak{e}$ implies $\mathfrak{q}$. The only difference between both properties is, that property $\mathfrak{q}$ only ensures the existence of the intended semantical equality relation, while $\mathfrak{e}$ additionally requires that our new logical connectives $=^\alpha$ are indeed associated with this relations. The connection between property $\mathfrak{q}$ and $\mathfrak{e}$ is already discussed in [And72a]. Andrews concludes that it seems natural to require the existence of logical connectives $=^\alpha$ in the signature, if one is interested in extensionality. In this thesis we are especially interested to shed some light on both: in extensionality of Leibniz equality in case $=^\alpha\notin \Sigma$ and in extensionality of Leibniz equality and/or primitive equality in case $=^\alpha\in \Sigma$.

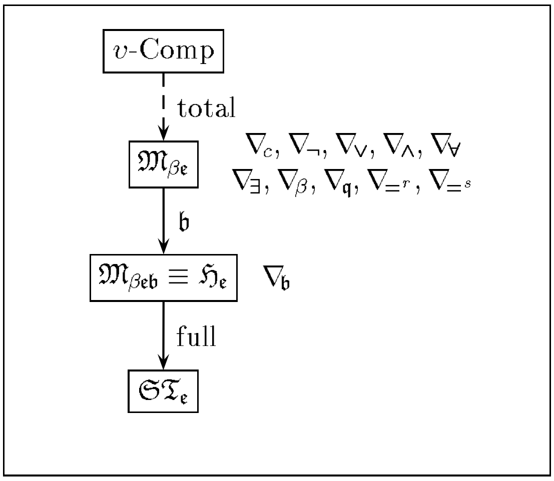

υ-Comp
total
$\mathfrak{M}_{\beta\mathfrak{e}}$
$\nabla_c, \nabla_\neg, \nabla_\vee, \nabla_\wedge, \nabla_\forall$
$\nabla_\exists, \nabla_\beta, \nabla_\mathfrak{q}, \nabla_{=^r}, \nabla_{=^s}$
$\mathfrak{b}$
$\mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}} \equiv \mathfrak{H}_\mathfrak{e}$
$\nabla_\mathfrak{b}$
full
$\mathfrak{ST}_\mathfrak{e}$

Figure 2.1: The landscape of Higher-Order Semantics with primitive equality

As property $\mathfrak{e}$ implies $\mathfrak{q}$ which in turn implies property $\mathfrak{f}$, it is easy to see that the landscape of higher-order semantics from Figure 1.1 at Page 7 collapses to the one in Figure 2.1.

Definition 2.41 (Extensionality for primitive equality). Analogous to the extensionality axioms for Leibniz equality, we can define such for primitive equality.

$$\begin{array}{lcl} \mathrm{EXT}^{\alpha\to\beta} & := & \forall F_{\alpha\to\beta}.\ \forall G_{\alpha\to\beta}(\forall X_\beta.\ F\ X = G\ X) \Rightarrow F =^\beta G \\ \mathrm{EXT}^o & := & \forall A_o.\ \forall B_o.\ (A \Leftrightarrow B) \Leftrightarrow A =^o B \end{array}$$

the *axioms of full extensionality for primitive equality.*

The following lemma shows that in a Σ-model with full equality the denotations of primitive equations and corresponding Leibniz equations are identical modulo υ.

Lemma 2.42 (Primitive and Leibniz equality). *If $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon) \in \mathfrak{M}_{\beta\mathfrak{e}}$, we have that $\upsilon(\mathcal{E}_\varphi(\mathbf{A} = \mathbf{B})) \equiv \upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq \mathbf{B}))$ for all $\mathbf{A}, \mathbf{B} \in wff(\Sigma)$.*

Proof: By lemma 2.36(3) we have $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq \mathbf{B})) \equiv \mathtt{T}$, iff $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$, since each $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ is also in $\mathfrak{M}_{\beta\mathfrak{q}}$. By property $\mathfrak{e}$ this is equivalent with $\upsilon(\mathcal{E}_\varphi(\mathbf{A} = \mathbf{B})) \equiv \mathtt{T}$. □

Lemma 2.43 (Extensionality in Σ-models with primitive equality).

1. *There exists a $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ for which EXT^o and EXT^o_L are not valid.*

2. *$\mathrm{EXT}^{\alpha\to\beta}$ and $\mathrm{EXT}^{\alpha\to\beta}_L$ are valid in $\mathcal{M}$, if $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$.*

3. *EXT^o and EXT^o_L is valid in $\mathcal{M}$, if $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$.*

Thus we can extend the table in Lemma 2.37 to the following one:

valid in	$\mathfrak{M}_{\beta\mathfrak{e}}$	$\mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$
$\mathrm{EXT}^{\alpha\to\beta}_L, \mathrm{EXT}^{\alpha\to\beta}$	$+$	$+$
$\mathrm{EXT}^o_L, \mathrm{EXT}^o$	$-$	$+$

Proof: The assertions follow from their respective counterparts in Lemma 2.37(3) with Lemma 2.42 and the fact that each $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ is also in $\mathfrak{M}_{\beta\mathfrak{q}}$. □

Theorem 2.44. *Let* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$*, then* $\mathcal{E}_\varphi(\dot{=}^\alpha)$ *and* $\mathcal{E}_\varphi(=^\alpha)$ *are equivalence relations on* $\mathcal{D}_\alpha$ *with respect to* v *for all assignments* φ*. If* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$*, then* $\mathcal{E}_\varphi(\dot{=}^\alpha) \equiv \mathcal{E}_\varphi(=^\alpha)$ *is the equality relation on* $\mathcal{D}_\alpha$*.*

Proof: If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$, then the proofs for $\dot{=}$ are provided by lemma 2.38. For $=$ they follow immediately with 2.42.

If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$, then the argumentation is analogous, and we first show that $\dot{=}$ and $=$ denote a functional congruence relation. As furthermore v is the identity relation on $\mathcal{D}_o$ we have that $\dot{=}$ and $=$ indeed denote the intended semantical equality relations. □

2.8 A Note on Defined and Primitive Equality

The extensional higher-order resolution approach discussed in Chapter 4 treats equality as a concept defined by Leibniz' principle. While this way of handling equality is theoretically convenient and suitable it turns out that it is quite inappropriate in practice[7], mainly because Leibniz equality introduces new flexible literals into the search space and thereby increases the amount of blind search with primitive substitution rule *Prim*. Furthermore proofs containing Leibniz equations are rather unintuitive and hard to understand for non-experts. Illustrating examples for such unintuitive problem formulations and proofs when using Leibniz equality are $\mathbf{E}_4^{ext}$ and $\mathbf{E}_5^{ext}$ presented in Section 8.1.

Our aim therefore is to avoid Leibniz equality and to use primitive equality instead. But in contrast to first-order logic we do not have the choice in higher-order logic to consider equality as a primitive notion only. The reason is, that infinitely many definitions of equality are always implicitly provided by our higher-order language (if we choose Standard- or Henkin semantics as the underlying semantical notion), and that there is no way to get rid of them. A definition of equality different from Leibniz equality, e.g., is discussed in Andrews' textbook [And86] (page 155), where he defines $\ddot{=}^\alpha := \lambda X_\alpha.\ \lambda Y_\alpha.\ \forall Q_{\alpha\to\alpha\to o}.\ (\forall Z_\alpha.\ (Q\ Z\ Z)) \Rightarrow (Q\ X\ Y)$. This definition is based on a reflexivity property, whereas Leibniz equality in some sense employs a substitutivity property. A proof for the eqivalence of both definitions of equality (and the one introduced below) in calculus $\mathcal{ER}$ will be presented in Section 8.3.

Apart from all the sensible definitions of equality in higher-order logic, one can always introduce additional artificial ones by adding arbitrary tautologous sub-formulae to a sensible definition of equality. For instance, if $\mathbf{A}^1 \ldots \mathbf{A}^n$ and $\mathbf{B}^1 \ldots \mathbf{B}^n$ are tautologous sentences, then we can define equality based on the Leibniz idea also by $\dddot{=}^\alpha := \lambda X_\alpha.\ \lambda Y_\alpha.\ \forall P_{\alpha\to o}.\ (\mathbf{A}^1 \wedge \ldots \wedge \mathbf{A}^n \wedge P\ X) \Rightarrow (P\ Y \wedge \mathbf{A}^1 \wedge \mathbf{A}_1 \wedge \ldots \wedge \mathbf{A}^n)$. By employing this idea, it is easy to see, that there are infinitely many valid but different ways of defining equality. As it is undecidable, whether a formula is a tautology, we also have the unfortunate fact that it is undecidable, whether a formula is equivalent to Leibniz equality. This argument immediately leads to the following corollary:

[7] Anyhow, the already mentioned successful case study with the Leo-system on the the examples from the MIZAR-articles Boolean and Basic Properties of sets showed that Leibniz equality can be handled at least to some extent in practice. Also the Tps-system [ABI$^+$96, AINP90], which has proven non-trivial theorems that require substantial amount of equational reasoning (e.g., theorem THM15b in [ABI$^+$96]), still employs Leibniz equality.

Corollary 2.45 (Defined equality in higher-order logic).
Given a higher-order sentence **A**. *It is undecidable whether* **A** *contains a sub-formula that expresses the equality between two terms.*

A consequence of this lemma is that we generally have to assume the presence of some defined equations in the input problems. We cannot generally detect all defined equations and remove them by primitive equations, as one might wish to. Consequently, even if we add a primitive equality treatment to the calculus, we still have to ensure that the calculus can handle Leibniz equality and all alternative definitions of equality as well (we want to point out, that calculus $\mathcal{ER}$ indeed can handle all forms of defined equality). This obviously contrasts with the situation in first-order logic, where one has the choice to either define equality (e.g., by axiomatising equality as a congruence relation) or to consider it as a primitive notion and to add special inference rules for equality (e.g., paramodulation rules) to the calculus.

Hence we have the requirement that we still have to take care of defined equality even if our calculus can handle primitive equality and we cannot remove the extensionality or other rules for defined equality from the calculus, as one might wish to. Of course, when adding some special primitive equality rules we can afterwards examine if some of the other rules became admissible and hence superfluous (c.f. remark 8.1).

A first approach to primitive equality treatment in higher-order logic is presented in Chapter 5, where we adapt the first-order paramodulation approach to our higher-order setting. The anticipated result of this attempt is, that just adapting the first-order paramodulation rules is not sufficient to ensure Henkin completeness. Incompleteness is caused by missing extensionality principles — this time with respect to positive primitive equations. At first glance the resulting higher-order paramodulation calculus seems to be inappropriate for some problem domains — e.g., for comparisons of sets, when sets are coded as characteristic functions, such that the extensionality properties play an important role. The reason is that the developed paramodulation approach combines term rewriting with difference reduction, whereas the difference reduction aspect comes from the the extensionality treatment already provided by calculus $\mathcal{ER}$ as well as from the new extensionality rules for positive primitive equations that are added to the calculus.

Hence, apart from the problem that well founded reduction orderings for improved paramodulation approaches are hard to develop in higher-order logic, the adapted paramodulation approach has to face a second problem: it has an intrinsic mixed term rewriting and difference reducing character which will be quite hard to control in practice.

As an alternative we therefore develop in Chapter 6 a second approach to primitive equality treatment in higher-order logic which adapts the ideas of first-order RUE-resolution [Dig79]. In contrast to the intrinsic mixed term rewriting and difference reducing character of the extensional higher-order paramodulation approach this extensional higher-order RUE-resolution approach has a pure difference reducing character which is probably much easier to guide in practice.

We assume for the Chapters 5 and 6 that the considered signature contains the primitive equality symbols $=^\alpha$ for all types α, and that the denotation of these symbol is fixed to the intended semantical equality relations of appropriate type. Furthermore, the primitive substitution rule is automatically extended, such that the new logical connectives $=^\alpha$ are now imitated as well.

Chapter 3

Higher-Order Model Existence

In this section we introduce abstract consistency properties and respective model existence theorems for the different semantical notions discussed in Chapter 2.1. These theorems have the following form, where $* \in \{\beta, \beta\mathfrak{b}, \beta\mathfrak{f}, \beta\mathfrak{q}, \beta\mathfrak{fb}, \beta\mathfrak{qb}\, \beta\mathfrak{e}, \beta\mathfrak{eb}\}$:

> *Theorem (Model Existence): For a given abstract consistency class $\mathfrak{Acc}_*$ and a set $H \in \mathfrak{Acc}_*$ there is a Σ-model $\mathcal{M}$ of H, such that $\mathcal{M} \in \mathfrak{M}_*$.*

The most important tools used in the proofs of the model existence theorems are the Σ-Hintikka sets. These sets are maximal elements in abstract consistency classes, and allow computations that resemble those in the considered semantical structures (e.g., Σ-Henkin models). These allow to construct $*$-valuations for the term structures that turn those into $*$-models.

The key step in the proof of the model existence theorems is an extension lemma, which guarantees a Σ-Hintikka set $\mathcal{H}$ for any set H of sentences in Γ_Σ. Apart from this, the proofs for the model existence theorems are standard.

3.1 Abstract Consistency

Let us now review a few technicalities that we will need for the proofs of the model existence theorems.

Definition 3.1 (Compactness). Let $\mathcal{C}$ be a class of sets.

1. $\mathcal{C}$ is called *closed under subsets*, iff for all sets S and T the following condition holds: if $S \subseteq T$ and $T \in \mathcal{C}$, then $S \in \mathcal{C}$.

2. $\mathcal{C}$ is called *compact*, iff for every set S the following condition holds: $S \in \mathcal{C}$, iff every finite subset of S is a member of $\mathcal{C}$.

Lemma 3.2. *If $\mathcal{C}$ is compact, then $\mathcal{C}$ is closed under subsets.*

Proof: Suppose $S \subseteq T$ and $T \in \mathcal{C}$. Every finite subset A of S is a finite subset of T, and since $\mathcal{C}$ is compact, we know that $A \in \mathcal{C}$. Thus $S \in \mathcal{C}$. □

Definition 3.3 (Sufficiently Pure). Let Σ be a signature and $\mathcal{T}$ be a set of Σ-sentences. $\mathcal{T}$ is called *sufficiently Σ-pure*, iff for each type α there is a set of constants $\mathcal{P}_\alpha \subseteq \mathcal{C}_\alpha$ with equal cardinality to $\mathit{wff}_\alpha(\Sigma)$, such that the elements of $\mathcal{P}$ do not occur in $\mathcal{T}$.

We will always presuppose that sets of sets of sentences are sufficiently Σ-pure in order to have enough witness constants. This can be obtained in practice by enriching the signature with spurious constants. Another way would be to use specially marked variables (which may never be instantiated) as in [Koh94b].

Definition 3.4 (Properties for Abstract Consistency Classes). Let Γ_Σ be a class of sets of Σ-sentences. We need the following conditions, where $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$ and $\mathbf{F}, \mathbf{G} \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$:[1]

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} \in \Phi$ and $\mathbf{B}$ is the β-normal form of $\mathbf{A}$, then $\mathbf{B} * \Phi \in \Gamma_\Sigma$.

$\nabla_\mathfrak{f}$ If $\mathbf{A} \in \Phi$ and $\mathbf{B}$ is the $\beta\eta$-normal form of $\mathbf{A}$, then $\mathbf{B} * \Phi \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\wedge$ If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi \cup \{\neg\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$.

$\nabla_\forall$ If $\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * (\mathbf{F}\ \mathbf{W}) \in \Gamma_\Sigma$ for each $\mathbf{W} \in \mathit{cwff}_\alpha(\Sigma)$.

$\nabla_\exists$ If $\neg\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \neg(\mathbf{F}\ w) \in \Gamma_\Sigma$ for any constant $w \in \Sigma_\alpha$, which does not occur in Φ.

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$.

$\nabla_\mathfrak{q}$ If $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \Phi$, then $\Phi * \neg(\mathbf{F}\ w \doteq^\beta \mathbf{G}\ w) \in \Gamma_\Sigma$ for any constant $w \in \Sigma_\alpha$, which does not occur in Φ.

(Additional abstract consistency conditions for primitive equality will be introduced later in Section 3.3.)

Remark 3.5. Note that for the connectives $\vee, \Pi^\alpha$ there are two conditions – a positive and a negative one – given in the definition above, namely $\nabla_\vee/\nabla_\wedge$ for $\vee$ and $\nabla_\forall/\nabla_\exists$ for Π^α. For $\doteq^o$ and $\doteq^{\alpha\to\beta}$ the situation is different, as we need only conditions for the negative cases. The positive cases can be inferred at the level of Hintikka sets by expanding the Leibniz definition of equality (see the proofs of $\overline{\nabla_\mathfrak{q}}^+$ in lemma 3.15 and $\overline{\nabla_\mathfrak{b}}^+$ in lemma 3.17).

Definition 3.6 (Abstract Consistency Classes). Let Σ be a signature and Γ_Σ be a class of sets of Σ-sentences. Using the properties from the previous definition we introduce the following abstract consistency classes:

$\mathfrak{Acc}_\beta$ If ∇_c, $\nabla_\neg$, ∇_β, $\nabla_\vee$, $\nabla_\wedge$, $\nabla_\forall$ and $\nabla_\exists$ are valid for Γ_Σ, then Γ_Σ is called an *abstract consistency class for Σ-models* ($\mathfrak{Acc}_\beta$).

Based upon this definition we introduce the following specialised abstract consistency classes: $\mathfrak{Acc}_{\beta\mathfrak{b}}$, $\mathfrak{Acc}_{\beta\mathfrak{f}}$, $\mathfrak{Acc}_{\beta\mathfrak{q}}$, $\mathfrak{Acc}_{\beta\mathfrak{fb}}$, $\mathfrak{Acc}_{\beta\mathfrak{qb}}$, where we indicate by indices which additional properties from $\{\nabla_\mathfrak{f}, \nabla_\mathfrak{q}, \nabla_\mathfrak{b}\}$ are required.

Sometimes we do not want to differentiate between the particular notions above. In this cases we simply speak of an *abstract consistency class*, with which we refer to an arbitrary but one in $\{\mathfrak{Acc}_\beta, \mathfrak{Acc}_{\beta\mathfrak{b}}, \mathfrak{Acc}_{\beta\mathfrak{f}}, \mathfrak{Acc}_{\beta\mathfrak{q}}, \mathfrak{Acc}_{\beta\mathfrak{fb}}, \mathfrak{Acc}_{\beta\mathfrak{qb}}\}$.

[1] In the following we will use $\varphi * \mathbf{A}$ as an abbreviation for $\varphi \cup \{A\}$.

Remark 3.7. Note that $\mathfrak{Acc}_{\beta\mathfrak{f}}$ corresponds to the abstract consistency property discussed by Andrews in [And71]. The only (technical) difference is that Andrews does not consider α-conversion as built-into the logic but needs a condition similar to ∇_β that requires α-standardised forms to be abstract consistent.

Lemma 3.8 (Non-atomic Consistency). *Let* Γ_Σ *be an abstract consistency class and* $\mathbf{A} \in cwff_o(\Sigma)$*, then for all* $\Phi \in \Gamma_\Sigma$ *we have* $\mathbf{A} \notin \Phi$ *or* $\neg\mathbf{A} \notin \Phi$*.*

Proof: Let $\mathbf{A} \in wff_o(\Sigma)$ and $\Phi \in \Gamma_\Sigma$, such that $\mathbf{A} \in \Phi$. By ∇_β we can assume that $\mathbf{A}$ is a β-normal form. So we prove the assertion by an induction over the structure of $\mathbf{A}$.

If $\mathbf{A}$ is atomic we get the assertion immediately by ∇_c. If $\mathbf{A}$ is not atomic, then its head must be a logical constant, therefore we can proceed by a case-analysis over the connectives and quantifiers.

Suppose $\mathbf{A}$ has the form $\neg\mathbf{B}$ and $\{\neg\mathbf{B}, \neg\neg\mathbf{B}\} \subseteq \Phi$. By $\nabla_\neg$ we know that $\{\neg\mathbf{B}, \mathbf{B}\} \cup \Phi \in \Gamma_\Sigma$, which contradicts the induction hypotheses. Now suppose $\mathbf{A}$ has the form $\mathbf{B} \vee \mathbf{C}$ and $\{\mathbf{B} \vee \mathbf{C}, \neg(\mathbf{B} \vee \mathbf{C})\} \subseteq \Phi$. By $\nabla_\vee$, $\nabla_\wedge$ we know that $\{\mathbf{B} \vee \mathbf{C}, \neg(\mathbf{B} \vee \mathbf{C}), \mathbf{B}, \neg\mathbf{B}, \neg\mathbf{C}\} \cup \Phi \in \Gamma_\Sigma$ or $\{\mathbf{B} \vee \mathbf{C}, \neg(\mathbf{B} \vee \mathbf{C}), \mathbf{C}, \neg\mathbf{B}, \neg\mathbf{C}\} \cup \Phi \in \Gamma_\Sigma$. In both cases the contradiction is given by the induction hypotheses. Suppose $\mathbf{A}$ has the form $\Pi(\lambda X. \mathbf{B})$ and $\{\Pi(\lambda X. \mathbf{B}), \neg\Pi(\lambda X. \mathbf{B})\} \subseteq \Phi$. By $\nabla_\exists$, $\nabla_\forall$ and ∇_β we know that $\{\Pi(\lambda Y. \mathbf{B}), \neg\Pi(\lambda Y. \mathbf{B}), [\mathbf{W}/Y]\mathbf{B}, \neg[\mathbf{W}/Y]\mathbf{B}\} \cup \Phi \in \Gamma_\Sigma$ which contradicts again the induction hypotheses. □

In contrast to [And71], we work with saturated abstract consistency classes in order to obtain total Σ-valuations, which makes the proofs of the model existence theorem much simpler and, e.g., yields much more natural models.

Definition 3.9 (Saturated). We call an abstract consistency class Γ_Σ *atomically saturated*, iff for all $\Phi \in \Gamma_\Sigma$ and for all atomic sentences $\mathbf{A} \in cwff_o(\Sigma)$, we have $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma_\Sigma$. If this property holds for all sentences $\mathbf{A} \in cwff_o(\Sigma)$, then we call Γ_Σ *saturated.*

Remark 3.10. Clearly, not all abstract consistency classes are saturated, since the empty set is one that is not, even if Σ is empty.

In the definition of abstract consistency class, we only had to require atomic consistency, i.e., that there are no atomic propositions that contradict each other in one abstract consistent set, to ensure consistency (see 3.8). The conjecture is that a similar theorem can be proven for saturatedness:

> *Conjecture:* Let Γ_Σ be an atomic saturated abstract consistency class. Then there exists an saturated abstract consistency class Γ'_Σ, with Γ_Σ is a subclass of Γ'_Σ.

Such a result would be of practical importance, as it allows to reduce the problem of proving saturatedness of a given calculus to proving atomic saturatedness.

Lemma 3.11. *Let* Γ_Σ *be a saturated abstract consistency class,* $\Phi \in \Gamma_\Sigma$ *and* $\mathbf{A}$ *an atomic sentence. Then* $\Phi * (\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$*.*

Proof: Since Γ_Σ is saturated and $\Phi \in \Gamma_\Sigma$, we must have $\Phi * (\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$ or $\Phi * \neg(\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$. We prove the assertion by refuting the second alternative. If $\Phi * \neg(\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$, then $\Phi \cup \{\neg(\mathbf{A} \vee \neg\mathbf{A}), \neg\mathbf{A}, \neg\neg\mathbf{A}, \mathbf{A}\} \in \Gamma_\Sigma$ by $\nabla_\wedge$ and $\nabla_\neg$. Since $\mathbf{A}$ is an atomic sentence we get a contradiction with lemma 3.8. □

Lemma 3.12 (Compactness of abstract consistency classes). *For each abstract consistency class Γ_Σ there exists an abstract consistency class Γ'_Σ of the same type, such that $\Gamma_\Sigma \subseteq \Gamma'_\Sigma$, and Γ'_Σ is compact. Furthermore Γ_Σ is saturated, iff Γ'_Σ is.*

Proof: (following and extending [And86], proposition no. 2506)
We choose $\Gamma'_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \text{every finite subset of } \Phi \text{ is in } \Gamma_\Sigma\}$. Now suppose that $\Phi \in \Gamma_\Sigma$. Γ_Σ is closed under subsets, so every finite subset of Φ is in Γ_Σ and thus $\Phi \in \Gamma'_\Sigma$. Hence $\Gamma_\Sigma \subseteq \Gamma'_\Sigma$.

Next let us show that each Γ'_Σ is compact. Suppose $\Phi \in \Gamma'_\Sigma$ and Ψ is an arbitrary finite subset of Φ. By definition of Γ'_Σ all finite subsets of Φ are in Γ_Σ and therefore $\Psi \in \Gamma'_\Sigma$. Thus all finite subsets of Φ are in Γ'_Σ whenever Φ is in Γ'_Σ. On the other hand, suppose all finite subsets of Φ are in Γ'_Σ. Then by the definition of Γ'_Σ the finite subsets of Φ are also in Γ_Σ, so $\Phi \in \Gamma'_\Sigma$. Thus Γ'_Σ is compact.

Next we show that if Γ_Σ satisfies ∇_*, then Γ'_Σ satisfies ∇_*, by considering the cases of definition 3.6. First note that by lemma 3.2 we have that Γ'_Σ is closed under subsets.

∇_c Let $\Phi \in \Gamma'_\Sigma$ and suppose there is an atom $\mathbf{A}$, such that $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$. Then $\{\mathbf{A}, \neg\mathbf{A}\} \in \Gamma_\Sigma$ contradicting ∇_c.

$\nabla_\neg$ Let $\Phi \in \Gamma'_\Sigma$, $\neg\neg\mathbf{A} \in \Phi$, Ψ be any finite subset of $\Phi * \mathbf{A}$ and $\Theta := (\Psi \setminus \{\mathbf{A}\}) * \neg\neg\mathbf{A}$. Θ is a finite subset of Φ, so $\Theta \in \Gamma_\Sigma$. Since Γ_Σ is an abstract consistency class and $\neg\neg\mathbf{A} \in \Theta$, we get $\Theta * \mathbf{A} \in \Gamma_\Sigma$ by $\nabla_\neg$. We know that $\Psi \subseteq \Theta * \mathbf{A}$ and Γ_Σ is closed under subsets, so $\Psi \in \Gamma_\Sigma$. Thus every finite subset Ψ of $\Phi * \mathbf{A}$ is in Γ_Σ and therefore by definition $\Phi * \mathbf{A} \in \Gamma'_\Sigma$.

$\nabla_\beta, \nabla_\mathfrak{f}, \nabla_\vee, \nabla_\wedge, \nabla_\forall, \nabla_\exists$ Analogous to $\nabla_\neg$.

$\nabla_\mathfrak{q}$ Let $\Phi \in \Gamma'_\Sigma$, $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \Phi$ and Ψ be any finite subset of $\Phi * \neg(\mathbf{F}\,\mathbf{W} \doteq \mathbf{G}\,\mathbf{W})$. We show that $\Psi \in \Gamma_\Sigma$. Clearly $\Theta := (\Psi \setminus \{\neg(\mathbf{F}\,\mathbf{W} \doteq \mathbf{G}\,\mathbf{W})\}) * \neg(\mathbf{F} \doteq \mathbf{G})$ is a finite subset of Φ and therefore $\Theta \in \Gamma_\Sigma$. Since Γ_Σ satisfies $\nabla_\mathfrak{q}$ and $\neg(\mathbf{F} \doteq \mathbf{G}) \in \Theta$, we have $\Theta * \neg(\mathbf{F}\,\mathbf{W} \doteq \mathbf{G}\,\mathbf{W}) \in \Gamma_\Sigma$ by $\nabla_\mathfrak{q}$. Furthermore, $\Psi \subseteq \Theta * \neg(\mathbf{F}\,\mathbf{W} \doteq \mathbf{G}\,\mathbf{W})$ and Γ_Σ is closed under subsets, so $\Psi \in \Gamma_\Sigma$. Thus every finite subset Ψ of $\Phi * \neg(\mathbf{F}\,\mathbf{W} \doteq \mathbf{G}\,\mathbf{W})$ is in Γ_Σ, therefore by definition we have $\Phi * \neg(\mathbf{F}\,\mathbf{W} \doteq \mathbf{G}\,\mathbf{W}) \in \Gamma'_\Sigma$.

$\nabla_\mathfrak{b}$ Let $\Phi \in \Gamma'_\Sigma$ with $\neg(\mathbf{A} \doteq \mathbf{B}) \in \Phi$ but $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\} \notin \Phi$ and $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\} \notin \Phi$. Then there exists finite subsets Φ_1 and Φ_2 of Φ, such that $\Phi_1 * \{\mathbf{A}, \neg\mathbf{B}\} \notin \Gamma_\Sigma$ and $\Phi_2 * \{\neg\mathbf{A}, \mathbf{B}\} \notin \Gamma_\Sigma$. Now we choose $\Phi_3 := \Phi_1 \cup \Phi_2 * \neg(\mathbf{A} \doteq \mathbf{B})$. Obviously Φ_3 is a finite subset of Φ and therefore $\Phi_3 \in \Gamma_\Sigma$. Since Γ_Σ satisfies $\nabla_\mathfrak{b}$, we have that $\Phi_3 \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi_3 \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$. From this and the fact that extensional abstract consistency classes are closed under subsets we get that $\Phi_1 \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi_2 \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$, which contradicts our assumption.

For the proof that Γ'_Σ is saturated, let $\Phi \in \Gamma'_\Sigma$, but neither $\Phi * \mathbf{A}$ nor $\Phi * \neg\mathbf{A}$ be in Γ'_Σ. Then there are finite subsets Φ^+ and Φ^- of Φ, such that $\Phi^+ * \mathbf{A} \notin \Gamma_\Sigma$ and $\Phi^- * \neg\mathbf{A} \notin \Gamma_\Sigma$ (since all finite subsets of Φ are in Γ_Σ). As $\Psi := \Phi^+ \cup \Phi^-$ is a finite subset of Φ, we have $\Psi \in \Gamma_\Sigma$. Furthermore, $\Psi * \mathbf{A} \in \Gamma_\Sigma$ or $\Psi * \neg\mathbf{A} \in \Gamma_\Sigma$, because Γ_Σ is saturated. Γ_Σ is closed under subsets, so $\Phi^+ * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi^- * \neg\mathbf{A} \in \Gamma_\Sigma$. This is a contradiction, so we can conclude that if $\Phi \in \Gamma_\Sigma$, then $\Phi * \mathbf{A} \in \Gamma'_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma'_\Sigma$. □

3.2 Hintikka Sets

Now we define Hintikka sets, which are maximal elements in an abstract consistency class. Hintikka sets connect syntax with semantics as they provide the basis for the model constructions in the model existence theorem 3.29.

Definition 3.13 (Σ-Hintikka Set). Let Γ_Σ be an abstract consistency class, then a set $\mathcal{H}$ is called a Σ-*Hintikka set for* Γ_Σ, iff it is maximal in Γ_Σ, i.e., iff for each sentence $\mathbf{D} \in cwff_o(\Sigma)$, such that $\mathcal{H} * \mathbf{D} \in \Gamma_\Sigma$, we already have $\mathbf{D} \in \mathcal{H}$.

In the following we discuss properties of Σ-Hintikka sets. Since we have different types of abstract consistency classes, depending on the additional requirements $\mathfrak{f}$, $\mathfrak{q}$ and $\mathfrak{b}$, we have to discuss different Hintikka lemmata.

Theorem 3.14 (Hintikka Lemma for $\mathfrak{Acc}_\beta$). *If Γ_Σ is a saturated $\mathfrak{Acc}_\beta$ and $\mathcal{H}$ is maximal in Γ_Σ, then the following statements hold for all $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$, $\mathbf{F} \in cwff_{\alpha\to o}(\Sigma)$ and $\mathbf{C}, \mathbf{D}, \mathbf{E} \in cwff_\alpha(\Sigma)$:*

$\overline{\nabla}_c^1$ $\mathbf{A} \notin \mathcal{H}$ *or* $\neg\mathbf{A} \notin \mathcal{H}$.

$\overline{\nabla}_c^2$ $\mathbf{A} \in \mathcal{H}$, *iff* $\neg\mathbf{A} \notin \mathcal{H}$.

$\overline{\nabla}_c^3$ $\neg\mathbf{A} \in \mathcal{H}$, *iff* $\mathbf{A} \notin \mathcal{H}$.

$\overline{\nabla}_\neg$ $(\neg\neg\mathbf{A}) \in \mathcal{H}$, *iff* $\mathbf{A} \in \mathcal{H}$.

$\overline{\nabla}_\beta$ *If* $\mathbf{A}\equiv_\beta\mathbf{B}$, *then* $\mathbf{A} \in \mathcal{H}$, *iff* $\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_\vee$ $(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$, *iff* $\mathbf{A} \in \mathcal{H}$ *or* $\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_\wedge$ $\neg(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$, *iff* $\neg\mathbf{A} \in \mathcal{H}$ *and* $\neg\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_\forall$ $\Pi^\alpha\mathbf{F} \in \mathcal{H}$, *iff for each* $\mathbf{D} \in cwff_\alpha(\Sigma)$ *we have* $(\mathbf{F}\ \mathbf{D}) \in \mathcal{H}$.

$\overline{\nabla}_\exists$ $\neg\Pi^\alpha\mathbf{F} \in \mathcal{H}$, *iff there is a* $\mathbf{D} \in cwff_\alpha(\Sigma)$, *such that* $\neg)\mathbf{F}\ \mathbf{D}) \in \mathcal{H}$.

$\overline{\nabla}_{\dot=}^r$ $\mathbf{A} \doteq^\alpha \mathbf{A} \in \mathcal{H}$

$\overline{\nabla}_{\dot=}^s$ *If* $\mathbf{F}[\mathbf{C}]_p \in \mathcal{H}$ *and* $\mathbf{C} \doteq^\alpha \mathbf{D} \in \mathcal{H}$, *then* $\mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$

$\overline{\nabla}_{\dot=}^{sy}$ $\mathbf{C} \doteq^\alpha \mathbf{D} \in \mathcal{H}$, *iff* $\mathbf{D} \doteq^\alpha \mathbf{C} \in \mathcal{H}$

$\overline{\nabla}_{\dot=}^{tr}$ $\mathbf{C} \doteq^\alpha \mathbf{D} \in \mathcal{H}$ *and* $\mathbf{D} \doteq^\alpha \mathbf{E} \in \mathcal{H}$, *then* $\mathbf{C} \doteq^\alpha \mathbf{E} \in \mathcal{H}$

$\overline{\nabla}_t$ $(\mathbf{A} \vee \neg\mathbf{A}) \in \mathcal{H}$ *for any sentence* $\mathbf{A}$.

Proof:

$\overline{\nabla}_c^1$ By 3.8.

$\overline{\nabla}_c^2$,$\overline{\nabla}_c^3$ Both are direct consequences of the saturation of Γ_Σ and $\overline{\nabla}_c^1$.

$\overline{\nabla}_\neg$ If $\neg\neg\mathbf{A} \in \mathcal{H}$, then $\mathcal{H} * \mathbf{A} \in \Gamma_\Sigma$ by $\nabla_\neg$. The maximality of $\mathcal{H}$ now gives us that $\mathbf{A} \in \mathcal{H}$. To obtain the converse, let us assume that $\mathbf{A} \in \mathcal{H}$. Then by $\overline{\nabla}_c^2$ we know that $\neg\mathbf{A} \notin \mathcal{H}$ and by $\overline{\nabla}_c^3$ $\neg\neg\mathbf{A} \in \mathcal{H}$.

$\overline{\nabla}_\beta$ Suppose $\mathbf{A}\equiv_\beta\mathbf{B}$. Since β-reduction is terminating and confluent there is unique $\mathbf{C}$, such that $\mathbf{C}$ is the β-normal-form of $\mathbf{A}$ and $\mathbf{B}$. Without loss of generality we show that if $\mathbf{A} \in \mathcal{H}$, then $\mathbf{B} \in \mathcal{H}$. For that we suppose that $\mathbf{A} \in \mathcal{H}$ but $\mathbf{B} \notin \mathcal{H}$. From the latter we get by $\overline{\nabla}_c^3$ that $\neg\mathbf{B} \in \mathcal{H}$. Note that the β-normal-form of $\mathbf{A}$ is $\mathbf{C}$ and of $\neg\mathbf{B}$ is $\neg\mathbf{C}$. By ∇_β and the maximality of $\mathcal{H}$ we know that $\{\mathbf{C}, \neg\mathbf{C}\} \in \mathcal{H}$, which contradicts $\overline{\nabla}_c^1$.

$\overline{\nabla}_\lor$ We get the first direction by $\nabla_\lor$ and the maximality of $\mathcal{H}$. For the converse direction let us assume that $\mathbf{A} \in \mathcal{H}$ or $\mathbf{B} \in \mathcal{H}$ but $(\mathbf{A}\lor\mathbf{B}) \notin \mathcal{H}$. Then by $\overline{\nabla}_c^3$ we get $\neg(\mathbf{A}\lor\mathbf{B}) \in \mathcal{H}$ and by the first direction of $\overline{\nabla}_\land$ we have $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ which contradicts the assumption with $\overline{\nabla}_c^1$.

$\overline{\nabla}_\land$ Analogous to the $\overline{\nabla}_\lor$ case; note that the argumentation is not circular. In both cases we use the forward direction of the counterpart to verify the backward direction, whereas forward directions are proven directly. The same holds for the proofs of $\overline{\nabla}_\forall$ and $\overline{\nabla}_\exists$ below.

$\overline{\nabla}_\forall$ Again, we get the first direction by $\nabla_\forall$ and the maximality of $\mathcal{H}$. For the converse direction let us assume that $(\mathbf{F}\ \mathbf{D}) \in \mathcal{H}$ for each $\mathbf{D} \in \mathit{cwff}_\alpha(\Sigma)$, but $\Pi^\alpha\mathbf{F} \notin \mathcal{H}$. Then $\neg\Pi^\alpha\mathbf{F} \in \mathcal{H}$ by $\overline{\nabla}_c^3$ and by the first direction of $\overline{\nabla}_\exists$ there is a $\mathbf{D} \in \mathit{cwff}_\alpha(\Sigma)$, such that $\neg(\mathbf{F}\ \mathbf{D}) \in \mathcal{H}$ which is a contradiction.

$\overline{\nabla}_\exists$ Analogous to $\overline{\nabla}_\forall$.

$\overline{\nabla}_{\doteq}^r$ Suppose $\mathbf{A} \doteq^\alpha \mathbf{A} \notin \mathcal{H}$. By $\overline{\nabla}_c^2$, the definition of $\doteq$, $\overline{\nabla}_\exists$ and $\overline{\nabla}_\mathfrak{b}^-$ we have $\neg(\neg\mathbf{Q}\ \mathbf{A} \lor \mathbf{Q}\ \mathbf{A})) \in \mathcal{H}$ for a $\mathbf{Q} \in \mathit{cwff}_{\beta\to o}(\Sigma)$. Applying $\overline{\nabla}_\land$ contradicts $\overline{\nabla}_c^2$.

$\overline{\nabla}_{\doteq}^s$ Suppose $\mathbf{F}[\mathbf{C}]_p \in \mathcal{H}$ and $\mathbf{C} \doteq \mathbf{D} \in \mathcal{H}$. From the latter we obtain $(\lambda P.\ \neg P\ \mathbf{C} \lor P\ \mathbf{D})(\lambda X.\ \mathbf{F}[X]_p) \in \mathcal{H}$ by the definition of $\doteq$ and $\overline{\nabla}_\forall$. Note that X is free for $\mathbf{F}[Y]_p$ so we have $\neg\mathbf{F}[\mathbf{C}]_p \lor \mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$ by $\overline{\nabla}_\beta$. From this we conclude with $\overline{\nabla}_\lor$ that $\neg\mathbf{F}[\mathbf{C}]_p \in \mathcal{H}$ or $\mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$. Since the first option contradicts our assumption with $\overline{\nabla}_c^1$, it must be the case that $\mathbf{F}[\mathbf{B}]_p \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{sy}$ By $\overline{\nabla}_{\doteq}^r$ and $\overline{\nabla}_{\doteq}^s$.

$\overline{\nabla}_{\doteq}^{tr}$ By $\overline{\nabla}_{\doteq}^r$, $\overline{\nabla}_{\doteq}^s$ and $\overline{\nabla}_{\doteq}^{sy}$.

$\overline{\nabla}_t$ Saturation of Γ_Σ and maximality of $\mathcal{H}$ entails that $\mathbf{A} \in \mathcal{H}$ or $\neg\mathbf{A} \in \mathcal{H}$. We now get the assertion by $\overline{\nabla}_\lor$. □

Depending on the kind of abstract consistency class we are considering, Hintikka sets have different properties. We discuss this different properties in the Hintikka lemmata below.

Theorem 3.15 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{f}}$). *If Γ_Σ is a saturated $\mathfrak{Acc}_{\beta\mathfrak{f}}$ and $\mathcal{H}$ is maximal in Γ_Σ, then for all $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathit{cwff}_o(\Sigma)$*

$\overline{\nabla}_\mathfrak{f}$ *If $\mathbf{A}\equiv_{\beta\eta}\mathbf{B}$, then $\mathbf{A} \in \mathcal{H}$ iff $\mathbf{B} \in \mathcal{H}$.*

Proof: Analogous to $\overline{\nabla}_\beta$ in Theorem 3.14 □

Theorem 3.16 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{q}}$). *If Γ_Σ is a saturated $\mathfrak{Acc}_{\beta\mathfrak{q}}$ and $\mathcal{H}$ is maximal in Γ_Σ, then for all $\mathbf{C} \in \mathit{cwff}_\alpha(\Sigma)$, and $\mathbf{F}, \mathbf{G} \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$:*

$\overline{\nabla}_{\mathfrak{q}}^{-}$ $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \mathcal{H}$, *iff there is a* $\mathbf{C} \in cwff_{\alpha}(\Sigma)$, *such that* $\neg(\mathbf{F}\ \mathbf{C} \doteq^{\beta} \mathbf{G}\ \mathbf{C}) \in \mathcal{H}$.

$\overline{\nabla}_{\mathfrak{q}}^{+}$ $\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G} \in \mathcal{H}$, *iff* $\mathbf{F}\ \mathbf{C} \doteq^{\beta} \mathbf{G}\ \mathbf{C} \in \mathcal{H}$ *for all* $\mathbf{C} \in cwff_{\alpha}(\Sigma)$

Proof:

$\overline{\nabla}_{\mathfrak{q}}^{-}$ We get the first direction by the definition of $\doteq$, $\nabla_{\mathfrak{q}}$ and the maximality of $\mathcal{H}$. For the converse let us suppose that $\neg(\mathbf{F}\ \mathbf{C} \doteq \mathbf{G}\ \mathbf{C}) \in \mathcal{H}$ but $\neg(\mathbf{F} \doteq \mathbf{G}) \notin \mathcal{H}$. From the latter we know by $\overline{\nabla}_{c}^{2}$, that $\mathbf{F} \doteq \mathbf{G} \in \mathcal{H}$ and by $\overline{\nabla}_{\doteq}^{s}$ we have that $\neg(\mathbf{G}\ \mathbf{C} \doteq \mathbf{G}\ \mathbf{C}) \in \mathcal{H}$ which contradicts $\overline{\nabla}_{\doteq}^{r}$ and $\overline{\nabla}_{c}^{1}$.

$\overline{\nabla}_{\mathfrak{q}}^{+}$ Suppose $\mathbf{F} \doteq \mathbf{G} \in \mathcal{H}$ but $\mathbf{F}\ \mathbf{C} \doteq \mathbf{G}\ \mathbf{C} \notin \mathcal{H}$, which means by $\overline{\nabla}_{c}^{3}$, that $\neg(\mathbf{F}\ \mathbf{C} \doteq \mathbf{G}\ \mathbf{C}) \in \mathcal{H}$. From this we get by the definition of $\doteq$, $\overline{\nabla}_{\exists}$ and $\overline{\nabla}_{\beta}$, that $\neg(\neg\mathbf{Q}\ (\mathbf{F}\ \mathbf{C}) \lor \mathbf{Q}\ (\mathbf{G}\ \mathbf{C})) \in \mathcal{H}$ for some $\mathbf{Q} \in wff_{\alpha\to o}(\Sigma)$. On the other hand we know from $\mathbf{F} \doteq \mathbf{G} \in \mathcal{H}$ by the definition of $\doteq$ and $\overline{\nabla}_{\forall}$ that $(\lambda P_{(\alpha\to\beta)\to o}.\ \neg P\ \mathbf{F} \lor P\ \mathbf{G})(\lambda X_{\alpha\to\beta}.\ \mathbf{Q}\ (X\ \mathbf{C})) \in \mathcal{H}$, and hence by $\overline{\nabla}_{\beta}$ that $\neg\mathbf{Q}\ (\mathbf{F}\ \mathbf{C}) \lor \mathbf{Q}\ (\mathbf{G}\ \mathbf{C}) \in \mathcal{H}$, which contradicts $\overline{\nabla}_{c}^{1}$. For the converse assume that $\mathbf{F}\ \mathbf{C} \doteq \mathbf{G}\ \mathbf{C} \in \mathcal{H}$ for all $\mathbf{C} \in \mathcal{H}$ but $\mathbf{F} \doteq \mathbf{G} \notin \mathcal{H}$. We get by $\overline{\nabla}_{c}^{3}$ that $\neg(\mathbf{F} \doteq \mathbf{G}) \in \mathcal{H}$ which contradicts the assumption with $\overline{\nabla}_{\mathfrak{q}}^{-}$ and $\overline{\nabla}_{c}^{1}$.

□

Theorem 3.17 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{b}}$). *If* Γ_{Σ} *is a saturated* $\mathfrak{Acc}_{\beta\mathfrak{b}}$ *and* $\mathcal{H}$ *is maximal in* Γ_{Σ}, *then for all* $\mathbf{A}, \mathbf{B} \in wff_{o}(\Sigma)$*:*

$\overline{\nabla}_{\mathfrak{b}}^{-}$ $\neg(\mathbf{A} \doteq^{o} \mathbf{B}) \in \mathcal{H}$, *iff* $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$

$\overline{\nabla}_{\mathfrak{b}}^{+}$ $(\mathbf{A} \doteq^{o} \mathbf{B}) \in \mathcal{H}$, *iff* $\{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{\Leftrightarrow}$ $(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$, *iff* $(\mathbf{A} \doteq^{o} \mathbf{B}) \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{c}$ *Either* $\mathbf{A} \doteq^{o} \mathbf{B} \in \mathcal{H}$ *or* $\mathbf{A} \doteq^{o} \neg\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{tf-}$ $\neg(\mathbf{T}_{o} \doteq \mathbf{F}_{o}) \in \mathcal{H}$, *if* $\mathbf{T}_{o}$ *and* $\mathbf{F}_{o}$ *are defined as in lemma 2.25.*

$\overline{\nabla}_{\doteq}^{tf+}$ *Either* $\mathbf{A} \doteq^{o} \mathbf{T}_{o} \in \mathcal{H}$ *or* $\mathbf{A} \doteq^{o} \mathbf{F}_{o} \in \mathcal{H}$.

Proof:

$\overline{\nabla}_{\mathfrak{b}}^{-}$ We get the first direction by the definition of $\doteq$, $\nabla_{\mathfrak{b}}$ and the maximality of $\mathcal{H}$. Now assume that $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ but $\neg(\mathbf{A} \doteq \mathbf{B}) \notin \mathcal{H}$. From the latter we know by the definition of $\doteq$ and $\overline{\nabla}_{\forall}$ that $\{(\lambda P_{o\to o}.\ \neg P\mathbf{A} \lor P\mathbf{B})(\lambda X_{o}.\ X), (\lambda P_{o\to o}.\ \neg P\ \mathbf{A} \lor P\ \mathbf{B})(\lambda X_{o}.\ \neg X)\} \subseteq \mathcal{H}$ and by $\overline{\nabla}_{\beta}$ and $\overline{\nabla}_{\lor}$ that one of $\{\neg\mathbf{A}, \neg\neg\mathbf{A}\}$, $\{\mathbf{B}, \neg\mathbf{B}\}$, $\{\neg\mathbf{A}, \neg\mathbf{B}\}$ or $\{\mathbf{B}, \neg\neg\mathbf{A}\}$ must be a subset of $\mathcal{H}$. All four cases contradict $\overline{\nabla}_{c}^{1}$ together with the assumption.

$\overline{\nabla}_{\mathfrak{b}}^{+}$ Since Γ_{Σ} is saturated we have $\mathbf{A} \in \mathcal{H}$ or $\neg\mathbf{A} \in \mathcal{H}$. From this we easily get the first direction by $\overline{\nabla}_{\doteq}^{s}$. For the converse suppose that $\{\mathbf{A}, \mathbf{B}\} \in \mathcal{H}$ or $\{\neg\mathbf{A}, \neg\mathbf{B}\} \in \mathcal{H}$ but $(\mathbf{A} \doteq \mathbf{B}) \notin \mathcal{H}$ which means by $\overline{\nabla}_{c}^{3}$ that $\neg(\mathbf{A} \doteq \mathbf{B}) \in \mathcal{H}$. By $\overline{\nabla}_{\mathfrak{b}}^{-}$ we have $\{\neg\mathbf{A}, \mathbf{B}\} \in \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \in \mathcal{H}$. In each of the four cases the contradiction follows by $\overline{\nabla}_{c}^{1}$.

$\overline{\nabla}_{\doteq}^{\Leftrightarrow}$ If we assume $(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$, then by the definition of $\Leftrightarrow$ and $\overline{\nabla}_{\wedge}$ we have $\{\neg\mathbf{A} \vee \mathbf{B}, \mathbf{A} \vee \neg\mathbf{B}\} \subseteq \mathcal{H}$, and by $\overline{\nabla}_{\vee}$ that $\{\neg\mathbf{A}, \mathbf{A}\} \subseteq \mathcal{H}$ or $\{\neg\mathbf{B}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$. Note that the first two alternatives are impossible because of $\overline{\nabla}_{c}^{1}$. Now we assume that $\mathbf{A} \doteq \mathbf{B} \notin \mathcal{H}$ from which we obtain by $\overline{\nabla}_{c}^{3}$ and $\overline{\nabla}_{\mathfrak{b}}^{-}$ that $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$. We have to consider four cases and in each case we get a contradiction with $\overline{\nabla}_{c}^{1}$.

$\overline{\nabla}_{\doteq}^{c}$ Assume that $\mathbf{A} \doteq \mathbf{B} \notin \mathcal{H}$ and $\mathbf{A} \doteq \neg\mathbf{B} \notin \mathcal{H}$. By $\overline{\nabla}_{c}^{3}$ we have $\neg(\mathbf{A} \doteq \mathbf{B}) \in \mathcal{H}$ and $\neg(\mathbf{A} \doteq \neg\mathbf{B}) \in \mathcal{H}$, and by $\overline{\nabla}_{\mathfrak{b}}^{-}$ we get from the former that $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ and from the latter that $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ or $\{\neg\mathbf{A}, \neg\neg\mathbf{B}\} \subseteq \mathcal{H}$. We have to consider four cases and in each we get a contradiction with $\overline{\nabla}_{c}^{1}$. Analogous we can show with $\overline{\nabla}_{\mathfrak{b}}^{+}$ that $\mathbf{A} \doteq \mathbf{B} \in \mathcal{H}$ and $\mathbf{A} \doteq \neg\mathbf{B} \in \mathcal{H}$ leads to a contradiction.

$\overline{\nabla}_{\doteq}^{tf-}$ From $\overline{\nabla}_{t}$ we know that $\mathbf{T}_o \in \mathcal{H}$. Hence by $\overline{\nabla}_{c}^{2}$ and $\overline{\nabla}_{c}^{3}$ that $\neg\mathbf{F}_o \in \mathcal{H}$ and finally by $\overline{\nabla}_{\mathfrak{b}}^{-}$ we get $\neg(\mathbf{T}_o \doteq^{o} \mathbf{F}_o) \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{tf+}$ Follows immediately from $\overline{\nabla}_{\doteq}^{c}$.

□

3.3 Primitive Equality

We now define abstract consistency properties for primitive equality. For this we have different options, e.g. we could introduce primitive equality by postulating $=$ to be a functional congruence relation or alternatively we could state properties connecting $=$ with $\doteq$.

Our concrete choice, namely a property postulating reflexivity and substitutivity of $=$, is motivated from a practical point of view, as we believe that reflexivity and substitutivity are more easy to verify in practical applications.

Definition 3.18 (Abstract Consistency with Primitive Equality). Let Σ be signature and let Γ_Σ be a $\mathfrak{Acc}_\beta$, then we define the following condition, where $\Phi \in \Gamma_\Sigma$:

$\nabla_{\mathfrak{e}}^{r}$ $\neg(\mathbf{A} =^{\alpha} \mathbf{A}) \notin \Phi$.

$\nabla_{\mathfrak{e}}^{s}$ If $\mathbf{F}[\mathbf{A}]_p \in \Phi$ and $\mathbf{A} = \mathbf{B} \in \Phi$, then $\Phi * \mathbf{F}[\mathbf{B}]_p \in \Gamma_\Sigma$.

Using this properties we introduce the abstract consistency classes $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and $\mathfrak{Acc}_{\beta\mathfrak{e}\mathfrak{b}}$ based upon the definition of $\mathfrak{Acc}_\beta$.

Instead of using reflexivity and substitution principles we can also introduce the following alternative definition for abstract consistency with primitive equality:

Definition 3.19 (Alternative Abstract Consistency with Primitive Equality). Let Σ be signature and let Γ_Σ be a $\mathfrak{Acc}_\beta$, then we define the following condition, where $\Phi \in \Gamma_\Sigma$:

$\nabla_{\underline{=}}^{\doteq}$ If $\neg(\mathbf{A} = \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} \doteq \mathbf{B}) \in \Gamma_\Sigma$.

$\nabla_{\underline{\doteq}}^{=}$ If $\neg(\mathbf{A} \doteq \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} = \mathbf{B}) \in \Gamma_\Sigma$.

Using this properties instead of $\nabla_{\mathfrak{e}}^r$ and $\nabla_{\mathfrak{e}}^s$ we can introduce analoga to the abstract consistency classes $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ above.

Remark 3.20. Just as in the case with Leibniz equality, we can extend a abstract consistency class with primitive equality so that it is compact.

Proof: We proceed just as in the proof of Lemma 3.12 but check the cases for $\nabla_{\mathfrak{e}}^r$ and $\nabla_{\mathfrak{e}}^s$.

For $\nabla_{\mathfrak{e}}^r$ let $\Phi \in \Gamma'_\Sigma$ and suppose there is an $\mathbf{A} \in \mathit{cwff}_o$ with $\neg(\mathbf{A} = \mathbf{A}) \in \Phi$. Then $\{\neg(\mathbf{A} = \mathbf{A})\} \in \Gamma_\Sigma$ contradicting $\nabla_{\mathfrak{e}}^r$.

For $\nabla_{\mathfrak{e}}^s$ let $\Phi \in \Gamma'_\Sigma$, $\{\mathbf{F}[\mathbf{A}]_p, \mathbf{A} = \mathbf{B}\} \subseteq \Phi$, Ψ be any finite subset of $\Phi * \mathbf{F}[\mathbf{B}]_p$ and $\Theta := (\Psi \setminus \{\mathbf{F}[\mathbf{B}]_p\}) \cup \{\mathbf{F}[\mathbf{A}]_p, \mathbf{A} = \mathbf{B}\}$. Θ is a finite subset of Φ, so $\Theta \in \Gamma_\Sigma$. Since Γ_Σ is an $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ and $\{\mathbf{F}[\mathbf{A}]_p, \mathbf{A} = \mathbf{B}\} \subseteq \Theta$, we get $\Theta * \mathbf{F}[\mathbf{B}]_p \in \Gamma_\Sigma$ by $\nabla_{\mathfrak{e}}^s$. We know that $\Psi \subseteq \Theta * \mathbf{F}[\mathbf{B}]_p$ and Γ_Σ is closed under subsets, so $\Psi \in \Gamma_\Sigma$. Thus every finite subset Ψ of $\Phi * \mathbf{F}[\mathbf{B}]_p$ is in Γ_Σ and therefore by definition $\Phi * \mathbf{F}[\mathbf{B}]_p \in \Gamma'_\Sigma$. □

The next lemma shows the connection between Leibniz equality and primitive equality for $\mathfrak{Acc}_{\beta\mathfrak{e}}$.

Lemma 3.21 (Leibniz versus Primitive Equality). *Let Γ_Σ be a saturated $\mathfrak{Acc}_{\beta\mathfrak{e}}$. For all $\Phi \in \Gamma_\Sigma$, all $\mathbf{A}, \mathbf{B} \in \mathit{wff}_o(\Sigma)$ and $\mathbf{F}, \mathbf{G} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ holds:*

1. *If $\neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Gamma_\Sigma$*
2. *If $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Gamma_\Sigma$*
3. *If $\mathbf{A} \doteq^\alpha \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} =^\alpha \mathbf{B} \in \Gamma_\Sigma$*
4. *If $\mathbf{A} =^\alpha \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \doteq^\alpha \mathbf{B} \in \Gamma_\Sigma$*
5. *If $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G}) \in \Phi$, then $\Phi * \neg(\mathbf{F}w =^\beta \mathbf{G}w) \in \Gamma_\Sigma$ for any constant $w \in \mathcal{C}_\alpha$, which does not occur in Φ.*

Proof:

1. Suppose $\neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Phi$, but $\Phi * \neg(\mathbf{A} =^\alpha \mathbf{B}) \notin \Gamma_\Sigma$. Since Γ_Σ is saturated we have $\Phi * \mathbf{A} =^\alpha \mathbf{B} \in \Gamma_\Sigma$ and by $\nabla_{\mathfrak{e}}^s$, that $\Phi * \mathbf{A} =^\alpha \mathbf{B} * \neg(\mathbf{B} \doteq^\alpha \mathbf{B}) \in \Gamma_\Sigma$. From the definition of $\doteq$ we further conclude with $\nabla_\exists$ that $\Phi * \mathbf{A} =^\alpha \mathbf{B} * \neg(\mathbf{B} \doteq^\alpha \mathbf{B}) * \neg(\neg p\, \mathbf{B} \lor p\, \mathbf{B}) \in \Gamma_\Sigma$ for any constant $p \in \mathcal{C}_{\alpha\to o}$. From this we get the contradiction with $\nabla_\land$ and lemma 3.8.
2. Suppose $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Phi$, but $\Phi * \neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \notin \Gamma_\Sigma$. Since Γ_Σ is saturated we have $\Phi * \mathbf{A} \doteq^\alpha \mathbf{B} \in \Gamma_\Sigma$ and by definition of $\doteq$, $\nabla_\forall$ and the subset closure of Γ_Σ that $\Phi * (\lambda P_{\alpha\to o}.\, \neg P\, \mathbf{A} \lor P\, \mathbf{B})(\lambda X_\alpha.\, \mathbf{A} = X) \in \Gamma_\Sigma$. By ∇_β, $\nabla_\lor$ and the subset closure of Γ_Σ we finally get that $\Phi * \neg(\mathbf{A} = \mathbf{A}) \in \Gamma_\Sigma$ or $\Phi * \mathbf{A} = \mathbf{B} \in \Gamma_\Sigma$. The former is contradictory with $\nabla_{\mathfrak{e}}^r$ and lemma 3.8, and the latter with the assumption $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Phi$ and lemma 3.8.
3. Suppose $\mathbf{A} \doteq^\alpha \mathbf{B} \in \Phi$, but $\Phi * \mathbf{A} =^\alpha \mathbf{B} \notin \Gamma_\Sigma$. Since Γ_Σ is saturated we have $\Phi * \neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Gamma_\Sigma$ and by (2) and the subset closure of Γ_Σ that $\Phi * \neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Gamma_\Sigma$ which contradicts the assumption with lemma 3.8.
4. Analogous to (3) with (1).
5. From $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G}) \in \Phi$ we can derive with (2), $\nabla_{\mathfrak{q}}$, (1) and the subset closure of Γ_Σ that $\Phi * \neg(\mathbf{FC} =^{\alpha\to\beta} \mathbf{GC}) \in \Gamma_\Sigma$.

□

Remark 3.22. Lemma 3.21 again shows that in an $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ the symbol $=$ defines the same relation as $\doteq$, namely a functional congruence relation modulo υ. And if we are considering an $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$ then both describe the equality relation. This shows that the conditions $\nabla_{\mathfrak{e}}^{r}$ and $\nabla_{\mathfrak{e}}^{s}$ are sufficient for this purpose. We could alternatively introduce primitive equality by requiring the statements 1. and 2. of lemma 3.21, i.e., as suggested in definition 3.19

We now discuss two new Hintikka lemmata, which take the logical nature of $=$ into account

Theorem 3.23 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{e}}$). *If Γ_{Σ} is a saturated $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and $\mathcal{H}$ is maximal in Γ_{Σ}, then the following statements hold for all $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathit{wff}_{\alpha}(\Sigma)$, $\mathbf{F}, \mathbf{G} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ and $\mathbf{D}, \mathbf{E} \in \mathit{wff}_{o}(\Sigma)$:*

$\overline{\nabla}_{=}^{r}$ $(\mathbf{A} =^{\alpha} \mathbf{A}) \in \mathcal{H}$.

$\overline{\nabla}_{=}^{s}$ *If* $\mathbf{D}[\mathbf{A}]_{p} \in \mathcal{H}$ *and* $\mathbf{A} =^{\alpha} \mathbf{B} \in \mathcal{H}$*, then* $\mathbf{D}[\mathbf{B}]_{p} \in \mathcal{H}$.

$\overline{\nabla}_{=}^{sy}$ $\mathbf{A} =^{\alpha} \mathbf{B} \in \mathcal{H}$*, iff* $(\mathbf{B} =^{\alpha} \mathbf{A}) \in \mathcal{H}$.

$\overline{\nabla}_{=}^{tr}$ $\mathbf{A} =^{\alpha} \mathbf{B} \in \mathcal{H}$ *and* $\mathbf{B} =^{\alpha} \mathbf{C} \in \mathcal{H}$*, then* $\mathbf{A} =^{\alpha} \mathbf{C} \in \mathcal{H}$

$\overline{\nabla}_{=}^{q-}$ $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G}) \in \mathcal{H}$*, iff there is a* $\mathbf{C} \in \mathit{wff}_{\alpha}(\Sigma)$*, such that* $\neg(\mathbf{F}\ \mathbf{C} =^{\beta} \mathbf{G}\ \mathbf{C}) \in \mathcal{H}$.

$\overline{\nabla}_{=}^{q+}$ $\mathbf{F} =^{\alpha\to\beta} \mathbf{G} \in \mathcal{H}$*, iff* $\mathbf{F}\ \mathbf{C} =^{\beta} \mathbf{G}\ \mathbf{C} \in \mathcal{H}$ *for all* $\mathbf{C} \in \mathit{wff}_{\alpha}(\Sigma)$.

$\overline{\nabla}_{=}^{\doteq+}$ $\mathbf{A} \doteq^{\alpha} \mathbf{B} \in \mathcal{H}$*, iff* $\mathbf{A} =^{\alpha} \mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_{=}^{\doteq-}$ $\neg(\mathbf{A} \doteq^{\alpha} \mathbf{B}) \in \mathcal{H}$*, iff* $\neg(\mathbf{A} =^{\alpha} \mathbf{B}) \in \mathcal{H}$.

Proof:

$\overline{\nabla}_{=}^{r}$ Follows by $\nabla_{\mathfrak{e}}^{r}$ and $\overline{\nabla}_{c}^{2}$.

$\overline{\nabla}_{=}^{s}$ By maximality of $\mathcal{H}$ and $\nabla_{\mathfrak{e}}^{s}$.

$\overline{\nabla}_{=}^{sy}$,$\overline{\nabla}_{=}^{tr}$ By $\overline{\nabla}_{=}^{r}$ and $\overline{\nabla}_{=}^{s}$

$\overline{\nabla}_{=}^{\doteq+}$ By maximality of $\mathcal{H}$, 3.21(3.) and 3.21(4.)

$\overline{\nabla}_{=}^{\doteq-}$ By maximality of $\mathcal{H}$, 3.21(1.) and 3.21(2.)

$\overline{\nabla}_{=}^{q-}$ Follows from $\overline{\nabla}_{\mathfrak{q}}^{-}$ with $\overline{\nabla}_{=}^{\doteq+}$.

$\overline{\nabla}_{=}^{q+}$ Follows from $\overline{\nabla}_{\mathfrak{q}}^{-}$ with $\overline{\nabla}_{=}^{\doteq-}$.

□

Theorem 3.24 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{e}\mathfrak{b}}$). *If Γ_{Σ} is a saturated $\mathfrak{Acc}_{\beta\mathfrak{e}\mathfrak{b}}$ and $\mathcal{H}$ is maximal in Γ_{Σ}, then for all $\mathbf{A}, \mathbf{B} \in \mathit{wff}_{o}(\Sigma)$:*

$\overline{\nabla}_{=}^{b-}$ $\neg(\mathbf{A} =^{o} \mathbf{B}) \in \mathcal{H}$*, iff* $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$.

$\overline{\nabla}_{=}^{b+}$ $\mathbf{A} =^{o} \mathbf{B} \in \mathcal{H}$*, iff* $\{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$.

$\overline{\nabla}_{=}^{\Leftrightarrow}$ $\mathbf{A} \Leftrightarrow \mathbf{B} \in \mathcal{H}$, *iff* $\mathbf{A} = \mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_{=}^{c}$ *Either* $\mathbf{A} = \mathbf{B} \in \mathcal{H}$ *or* $\mathbf{A} = \neg\mathbf{B} \in \mathcal{H}$.

Proof: The statements follow direct from their counterparts $\overline{\nabla}_{\mathfrak{b}}^{-}$ – $\overline{\nabla}_{\doteq}^{c}$ in lemma 3.17 with the help of $\overline{\nabla}_{=}^{\doteq+}$ and $\overline{\nabla}_{=}^{\doteq-}$. □

3.4 Model Existence

We shall now present the proof of the abstract extension lemma, which will nearly immediately yield the model existence theorems. For the proof we adapt the construction of Henkin's completeness proof from [Hen50, Hen96].

Theorem 3.25 (Abstract Extension Lemma). *Let* Σ *be a signature,* Γ_Σ *be a compact abstract consistency class and let* $H \in \Gamma_\Sigma$ *be sufficiently* Σ*-pure. Then there exists a* Σ*-Hintikka set* $\mathcal{H}$ *for* Γ_Σ*, such that* $H \subseteq \mathcal{H}$*.*

Proof: We construct $\mathcal{H}$ by inductively constructing a sequence of sets $\mathcal{H}^i$ such that $\mathcal{H}^i \in \Gamma_\Sigma$. Then the Σ-Hintikka set is $\mathcal{H} := \bigcup_{i \in \mathbb{N}} \mathcal{H}^i \in \Gamma_\Sigma$.
Let $\mathbf{A}_1, \mathbf{A}_2, \ldots$ be an enumeration of $\mathit{cwff}_o(\Sigma)$. We define $\mathcal{H}^0 := H$ and the set $\mathcal{H}^{n+1}$ according to the following table 3.1. Since the construction is uniform for all kinds of abstract consistency classes $\mathcal{H}^{n+1}$ depends on the respective kind of abstract consistency class Γ_Σ we are interested in and in the properties of $\mathbf{A}_n$ with respect to this Γ_Σ. (*How to read the table*: Assume Γ_Σ is an an $\mathfrak{Acc}_{\beta\mathfrak{q}}$ and $\mathbf{A}_n$ is of form $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})$. The table defines $\mathcal{H}^{n+1}$ to be $\mathcal{H}^n * \mathbf{A}_n * \neg(\mathbf{F}\ w \doteq^{\beta} \mathbf{G}\ w)$ for a fresh $w \in \mathcal{C}_\alpha$ in case $\mathbf{A}_n \in \Gamma_\Sigma$ and $\mathcal{H}^n$ otherwise.)

$\mathcal{H}^{n+1}$		$\mathfrak{Acc}_\beta/\mathfrak{Acc}_{\beta\mathfrak{b}}/$ $\mathfrak{Acc}_{\beta\mathfrak{f}}/\mathfrak{Acc}_{\beta\mathfrak{fb}}$	$\mathfrak{Acc}_{\beta\mathfrak{q}}/\mathfrak{Acc}_{\beta\mathfrak{qb}}/$	$\mathfrak{Acc}_{\beta\mathfrak{e}}/\mathfrak{Acc}_{\beta\mathfrak{eb}}/$
$\mathcal{H}^n * \mathbf{A}_n \notin \Gamma_\Sigma$		$\mathcal{H}^n$	$\mathcal{H}^n$	$\mathcal{H}^n$
$\mathcal{H}^n * \mathbf{A}_n$ $\in \Gamma_\Sigma$ and	$\mathbf{A}_n$ of form $\neg\Pi^\alpha\mathbf{B}$	$\mathcal{H}^n * \mathbf{A}_n *$ $\neg(\mathbf{B}\ w)$	$\mathcal{H}^n * \mathbf{A}_n *$ $\neg(\mathbf{B}\ w)$	$\mathcal{H}^n * \mathbf{A}_n *$ $\neg(\mathbf{B}\ w)$
	$\mathbf{A}_n$ of form $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n *$ $\neg(\mathbf{F}\ w \doteq^{\beta} \mathbf{G}\ w)$	$\mathcal{H}^n * \mathbf{A}_n *$ $\neg(\mathbf{F}\ w \doteq^{\beta} \mathbf{G}\ w)$
	$\mathbf{A}_n$ of form $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G})$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n *$ $\neg(\mathbf{F}\ w =^{\beta} \mathbf{G}\ w)$
	$\mathbf{A}_n$ of other form	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n$
				$w \in \mathcal{C}_\alpha$ is a constant which is fresh for $\mathcal{H}^n$

Figure 3.1: Hintikka set construction: How to construct sets $\mathcal{H}^{n+1}$ from $\mathcal{H}^n$

Next we show by induction, that $\mathcal{H}^n \in \Gamma_\Sigma$ for all $n \in \mathbb{N}$. The base case holds by construction (for all kinds of abstract consistency classes). So let $\mathcal{H}^n \in \Gamma_\Sigma$. We have to show that $\mathcal{H}^{n+1} \in \Gamma_\Sigma$. This is trivial in case $\mathcal{H}^n * \mathbf{A}_n \notin \Gamma_\Sigma$ (again for all abstract consistency classes). In case $\mathcal{H}^n * \mathbf{A}_n \in \Gamma_\Sigma$ we have to consider four sub-cases:

1. If $\mathbf{A}_n$ is of form $\neg\Pi^\alpha\mathbf{B}$, then we get the conclusion trivially by $\nabla_\exists$ (for all cases).

2. If $\mathbf{A}_n$ is of form $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})$ the conclusion is either trivial (by $\nabla_{\exists}$ in case of $\mathfrak{Acc}_{\beta}$, $\mathfrak{Acc}_{\beta\mathfrak{b}}$, $\mathfrak{Acc}_{\beta\mathfrak{f}}$ or $\mathfrak{Acc}_{\beta\mathfrak{fb}}$), or follows by $\nabla_{\mathfrak{q}}$ in case of $\mathfrak{Acc}_{\beta\mathfrak{q}}$ or $\mathfrak{Acc}_{\beta\mathfrak{qb}}$, or follows by $\nabla_{\mathfrak{q}}$ and Lemma 3.21(1), 3.21(5), and 3.21(2) in case of $\mathfrak{Acc}_{\beta\mathfrak{e}}$ or $\mathfrak{Acc}_{\beta\mathfrak{eb}}$.

3. If $\mathbf{A}_n$ is of form $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G})$ the conclusion is either trivial (by $\nabla_{\exists}$ in case of an $\mathfrak{Acc}_{\beta}$, $\mathfrak{Acc}_{\beta\mathfrak{b}}$, $\mathfrak{Acc}_{\beta\mathfrak{f}}$, $\mathfrak{Acc}_{\beta\mathfrak{fb}}$, $\mathfrak{Acc}_{\beta\mathfrak{q}}$, $\mathfrak{Acc}_{\beta\mathfrak{qb}}$) or follows by 3.21(5).

4. If $\mathbf{A}_n$ is of any other form, then the conclusion is trivial (for all cases).

Since Γ_{Σ} is compact, we also have $\mathcal{H} \in \Gamma_{\Sigma}$.

Now we know that our inductively defined set $\mathcal{H}$ is indeed in Γ_{Σ} and that $H \subseteq \mathcal{H}$. It only remains to show that $\mathcal{H}$ is maximal in Γ_{Σ}. So let $\mathbf{A}_n \in \mathit{wff}_o(\Sigma)$ be the n-th sentence from the above sequence, such that $\mathcal{H} * \mathbf{A}_n \in \Gamma_{\Sigma}$. Since $\mathcal{H}$ is closed under subsets we know that $\mathcal{H}^n * \mathbf{A}_n \in \Gamma_{\Sigma}$. By definition of $\mathcal{H}^{n+1}$ we conclude that $\mathbf{A}_n \in \mathcal{H}^{n+1}$ and hence $\mathbf{A}_n \in \mathcal{H}$. □

Next we define two congruence relations which we need in the model existence theorems below in order to build quotient models.

Definition 3.26 (Congruence Relations $\sim_{\mathcal{H}}$ and $\dot\sim_{\mathcal{H}}$). Let Γ_{Σ} be an abstract consistency class and $\mathcal{H}$ be a Hintikka set for Γ_{Σ}. For all $\mathbf{A}, \mathbf{B} \in \mathit{wff}(\Sigma)$ we define:

$$\mathbf{A}_{\gamma} \dot\sim_{\mathcal{H}} \mathbf{B}_{\gamma}, \text{ iff } \mathbf{A} \doteq^{\gamma} \mathbf{B} \in \mathcal{H}.$$

$$\mathbf{A}_{\gamma} \sim_{\mathcal{H}} \mathbf{B}_{\gamma}, \text{ iff } \begin{cases} \mathbf{A} \equiv \mathbf{B} & \text{if } \gamma \equiv \iota \\ \{\mathbf{A}, \mathbf{B}\} \in \mathcal{H} \text{ or } \{\mathbf{A}, \mathbf{B}\} \cap \mathcal{H} \equiv \emptyset & \text{if } \gamma \equiv o \\ \mathbf{AC} \sim_{\mathcal{H}} \mathbf{BC} \text{ for all } \mathbf{C} \in \mathit{wff}_{\alpha}(\Sigma) & \text{if } \gamma \equiv \alpha \to \beta \end{cases}$$

Lemma 3.27 (Functional Congruence Relations). *Let Γ_{Σ} be an abstract consistency class and $\mathcal{H}$ be a Hintikka set for Γ_{Σ}. Then $\sim_{\mathcal{H}}$ is a functional congruence relation, if Γ_{Σ} is an $\mathfrak{Acc}_{\beta}$ and $\dot\sim_{\mathcal{H}}$ is a functional congruence relation, if Γ_{Σ} is an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$.*

Proof: $\dot\sim_{\mathcal{H}}$ is a functional congruence relation by $\overline{\nabla}^{r}_{\doteq}$, $\overline{\nabla}^{sy}_{\doteq}$, $\overline{\nabla}^{tr}_{\doteq}$, $\overline{\nabla}^{-}_{\mathfrak{q}}$ and $\overline{\nabla}^{+}_{\mathfrak{q}}$, which are valid in case Γ_{Σ} is an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$.

Note that $\sim_{\mathcal{H}}$ is a functional congruence by construction. □

Remark 3.28. Note that in Lemma 2.37 $\mathrm{EXT}^{\alpha\to\beta}_{L}$ does not hold for $\doteq$ and hence $\dot\sim_{\mathcal{H}}$ is not a functional congruence in case Γ_{Σ} is not at least an $\mathfrak{Acc}_{\beta\mathfrak{q}}$. Hence $\dot\sim_{\mathcal{H}}$ is unsuitable for the model construction of an $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{fb}}$ (or $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{b}}$) from a given $\mathcal{M}' \in \mathfrak{M}_{\beta\mathfrak{f}}$ (or $\mathcal{M}' \in \mathfrak{M}_{\beta}$) as demonstrated below but fits well for the construction of an $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{qb}}$. Fortunately the relation $\sim_{\mathcal{H}}$ is already a functional congruence in case Γ_{Σ} is an $\mathfrak{Acc}_{\beta\mathfrak{f}}$.

We now use the Σ-Hintikka sets, guaranteed by lemma 3.25, to construct a Σ-valuation for the Σ-term structure that turns it into the desired model $\mathcal{M}$.

Theorem 3.29 (Model Existence Theorem). *Let Γ_{Σ} be an saturated $\mathfrak{Acc}$ and let $H \in \Gamma_{\Sigma}$ be a sufficiently Σ-pure set of sentences. For all $i \in \{\beta, \beta\mathfrak{f}, \beta\mathfrak{fb}, \beta\mathfrak{q}, \beta\mathfrak{qb}, \beta\mathfrak{e}, \beta\mathfrak{eb}\}$ (cf. Def. 2.28, 2.31 and 2.39) we have: If Γ_{Σ} is an $\mathfrak{Acc}_i$ (cf. Def. 3.6 and 3.18), then there exists a countable model in $\mathcal{M} \in \mathfrak{M}_i$ that satisfies H.*

Proof: Let Γ_{Σ} be an abstract consistency class. We can assume without loss of generality (see lemma 3.12) that Γ_{Σ} is compact, so the preconditions of 3.25 are met, and therefore there exists a Σ-Hintikka set $\mathcal{H} \subseteq \mathit{wff}_o(\Sigma)$ for Γ_{Σ}, such that $H \subseteq \mathcal{H}$.

Now, for each different kind of abstract consistency class, we will construct a countable model $\mathcal{M}^{\mathcal{H}}$ of the corresponding type. These model constructions closely reflect the relations of the different model types as discussed in Chapter 2.1 and shown in figures 1.1 and 2.1. We start with the construction of an $\mathcal{M}_1^{\mathcal{H}} \in \mathfrak{M}_\beta$ and an $\mathcal{M}_2^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{f}}$ based upon the non-functional termstructure $\mathcal{TS}(\Sigma)^\beta$ and the functional $\mathcal{TS}(\Sigma)^{\beta\eta}$. The remaining model constructions are then based upon these two basic constructions.

$\mathfrak{M}_\beta$ Let Γ_Σ be an $\mathfrak{Acc}_\beta$. Given the Σ-Hintikka set $\mathcal{H}$ with $H \subseteq \mathcal{H}$ from above, we choose $v(\mathbf{C}) := \mathtt{T}$, iff $\mathbf{C} \in \mathcal{H}$. Note that we have $v(\mathbf{C}) := \mathtt{F}$, iff $\neg\mathbf{C} \in \mathcal{H}$ by $\overline{\nabla}_c^2$. By $\overline{\nabla}_\beta$ we know that v is well-defined on $\mathit{cwff}(\Sigma)\downarrow_\beta$ and by $\overline{\nabla}_c^2$, $\overline{\nabla}_c^3$ we have that v is a total function on $\mathcal{TS}_o(\Sigma)^\beta$.

Furthermore by $\overline{\nabla}_c^2$, $\overline{\nabla}_c^3$, $\overline{\nabla}_\vee$ and $\overline{\nabla}_\forall$ we have that v is a Σ-valuation of the Σ-term structure $\mathcal{TS}(\Sigma)^\beta$ and thus $\mathcal{M}_1^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^\beta, v)$ is a Σ-model by construction. We have $\mathcal{M}_1^{\mathcal{H}} \models H$, since $H \subseteq \mathcal{H}$. Note that $\mathcal{M}_1^{\mathcal{H}}$ is indeed countable, since the sets of well-typed formulae are countable.

$\mathfrak{M}_{\beta\mathfrak{f}}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{f}}$ and hence also an $\mathfrak{Acc}_\beta$. Analogous to the previous case we construct the countable Σ-model $\mathcal{M}_2^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^{\beta\eta}, v)$ with $\mathcal{M}_2^{\mathcal{H}} \models H$. Note that in this case v is well-defined on $\mathcal{TS}_o(\Sigma)^{\beta\eta}$ because of $\overline{\nabla}_\mathfrak{f}$. By lemma 2.15 we know that $\mathcal{M}_2^{\mathcal{H}}$ is functional and hence $\mathcal{M}_2^{\mathcal{H}}$ is in $\mathfrak{M}_{\beta\mathfrak{f}}$.

We proceed with the construction of an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ and $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{b}}$ based upon the previous construction of an $\mathcal{M}_1^{\mathcal{H}} \in \mathfrak{M}_\beta$ and accordingly of an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ and an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{fb}}$ based upon an $\mathcal{M}_2^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{f}}$. Thus we start out with a countable Σ-model $\mathcal{M}_1^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^\beta, v)$ or $\mathcal{M}_2^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^{\beta\eta}, v)$, such that $\mathcal{M}_i^{\mathcal{H}} \models H$, for $i = 1, 2$. Property $\mathfrak{q}$ is easy to verify, as it follows from the properties discussed in the Hintikka-lemmata 3.14 and 3.16.

$\mathfrak{M}_{\beta\mathfrak{q}}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{q}}$. From $\overline{\nabla}_{\doteq}^r$, $\overline{\nabla}_{\doteq}^{sy}$, $\overline{\nabla}_{\doteq}^{tr}$, $\overline{\nabla}_\mathfrak{q}^-$ and $\overline{\nabla}_\mathfrak{q}^+$ we can derive that $\doteq^\alpha$ is indeed the $\mathfrak{q}^\alpha$ required by property $\mathfrak{q}$ and hence $\mathcal{M}^{\mathcal{H}}$ is a countable Σ-model in $\mathfrak{M}_{\beta\mathfrak{q}}$.

To verify property $\mathfrak{b}$ instead we have to construct an $\mathcal{M}^{\mathcal{H}}$ from $\mathcal{M}_i^{\mathcal{H}}$ $(i = 1, 2)$ by reducing the set of truth values to $\{\mathtt{T}, \mathtt{F}\}$, which can be done with the help of a functional congruence relation.

$\mathfrak{M}_{\beta\mathfrak{b}}$,$\mathfrak{M}_{\beta\mathfrak{fb}}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{q}}$ or $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. By lemma 3.27, $\overline{\nabla}_\mathfrak{b}^+$ and $\overline{\nabla}_c^3$ we can show that the relation $\sim_{\mathcal{H}}$ defined in 3.26 is a functional Σ-congruence for $\mathcal{M}_i^{\mathcal{H}}$ and thus, by lemma 2.34, the quotient structure $\mathcal{M}_i^{\mathcal{H}}/_{\sim_{\mathcal{H}}}$ is a functional Σ-model that satisfies H. From $\overline{\nabla}_c^2$, $\overline{\nabla}_c^3$ and the choice of v we conclude that $\sim_{\mathcal{H}}$ has exactly two equivalence classes on $\mathcal{TS}_o(\Sigma)^{\beta\eta}$. Thus we have $\mathcal{D}_o \equiv \{\mathtt{T} := [\![\mathbf{T}_o]\!]_\sim, \mathtt{F} := [\![\mathbf{F}_o]\!]_\sim\}$, if we define $\mathbf{T}_o$ and $\mathbf{F}_o$ as in lemma 2.25. Using $\overline{\nabla}_t$ and $\overline{\nabla}_c^2$ we further get that v is the identity relation. Finally note that $\mathcal{M}_i^{\mathcal{H}}/_{\sim_{\mathcal{H}}}$ is countable since $\mathcal{M}_i^{\mathcal{H}}$ is.

We finish the constructions for the cases without a primitive notion of equality with the construction of a Σ-Henkin model $\mathcal{N}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{qb}}$ in case we are considering an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$.

We start with $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ guaranteed by the discussion above. Analogous to the construction of an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{b}}$, we make use of a functional congruence relation in order to construct a quotient model which fulfills property $\mathfrak{b}$. But instead of the relation $\sim_{\mathcal{H}}$ we had to use before, we apply the simpler relation $\dot{\sim}_{\mathcal{H}}$ which is a functional congruence relation for the elements of $\mathfrak{M}_{\beta\mathfrak{qb}}$.

$\mathfrak{H} \equiv \mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$ By lemma 3.27 and $\overline{\nabla}_{\mathfrak{b}}^{+}, \overline{\nabla}_{c}^{3}$ we know that the relation $\dot{\sim}_{\mathcal{H}}$ is a functional Σ-congruence for $\mathcal{M}^{\mathcal{H}}$, so the quotient structure $\mathcal{N}^{\mathcal{H}} := \mathcal{M}^{\mathcal{H}}/_{\dot{\sim}_{\mathcal{H}}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ with $\mathcal{N}^{\mathcal{H}} \models H$ by lemma 2.34. Now we can conclude that $\mathcal{D}_{o} \equiv \{\mathtt{T}, \mathtt{F}\}$ with the same argumentation as in the $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ case.

It remains to discuss the cases with primitive equality and we start with the $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$, resp. $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$ from above.

$\mathfrak{M}_{\beta\mathfrak{e}}$ Let Γ_{Σ} be an $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and hence an $\mathfrak{Acc}_{\beta\mathfrak{q}}$. We construct the countable $\mathcal{E}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ with $\mathcal{E}^{\mathcal{H}} \models H$ as discussed above. It remains to show that property $\mathfrak{e}$ is valid for $\mathcal{E}^{\mathcal{H}}$ which follows from property $\mathfrak{q}$ by $\overline{\nabla}_{=}^{\dot{=}+}$ and $\overline{\nabla}_{=}^{\dot{=}-}$.

$\mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$ This case is analogous: Let Γ_{Σ} be an $\mathfrak{Acc}_{\beta\mathfrak{e}\mathfrak{b}}$ and hence an $\mathfrak{Acc}_{\beta\mathfrak{q}\mathfrak{b}}$. We construct a countable $\mathcal{E}^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^{\beta\eta}, v) \in \mathfrak{H}$ with $\mathcal{E}^{\mathcal{H}} \models H$. Again property $\mathfrak{e}$ is valid for $\mathcal{E}^{\mathcal{H}}$ by property $\mathfrak{q}$, $\overline{\nabla}_{=}^{\dot{=}+}$, and $\overline{\nabla}_{=}^{\dot{=}-}$.

□

Chapter 4

Extensional Higher-Order Resolution: $\mathcal{ER}$

In this chapter we introduce the calculus $\mathcal{ER}$ for extensional higher-order resolution. The key idea of $\mathcal{ER}$ is to integrate the search for unifiers and for refutations on the same level, i.e., we allow for recursive calls to the refutation process from within higher-order unification and vice versa. $\mathcal{ER}$ is Henkin complete without additional axioms. In Section 4.1 we shall first discuss calculus $\mathcal{ER}$ and illustrate the connections, modifications and extensions with respect to the underlying calculus $\mathcal{HORES}$ as introduced in [Koh94b]. We then formally introduce calculus $\mathcal{ER}$ in Section 4.2 and define $\mathcal{ER}_f$ and $\mathcal{ER}_{fc}$, which generalise $\mathcal{ER}$ by unfolding clause normalisation derivations and additionally providing the *FlexFlex*-unification rule. In Section 4.3 we present a simplified proof (compared to the technique employed in [Koh94b]) of a lifting lemma for the generalised calculus $\mathcal{ER}_{fc}$ before we examine Henkin completeness of $\mathcal{ER}_{fc}$ in Section 4.4. As we are actually more interested in calculus $\mathcal{ER}$ than in $\mathcal{ER}_{fc}$ or $\mathcal{ER}_f$, we discuss the equivalence of these three calculi in Section 4.5.

4.1 A Review of $\mathcal{HORES}$ and $\mathcal{ER}$

Traditional first-order resolution [Rob65] can be seen as a two layered approach, where the overall search for a refutation (based on the resolution rule *resolve* and the factorisation rule *factorise*) is performed at a layer above: this layer passes subproblems to the lower layer, such as the initial clause normalisation process or the intermediate unification problems. An important fact is that all the side computations performed at the lower layer are decidable. First-order unification is the main engine of first-order resolution, it is in a sense a filter in the refutation search in order to separate inappropriate clauses from the search space and to compute most general representations for all suitable variable instantiations for the appropriate ones.

In our higher-order setting clause normalisation remains uncritical and the set of clauses $\mathcal{CNF}(\Phi)$ for a given set of higher-order formulae Φ can easily be computed with the clause normalisation rules as follows: Initially all formulae $\mathbf{A} \in \Phi$ are replaced by pre-clause $[\mathbf{A}_{\downarrow_h}]^T$. Then the clause normalisation rules are exhaustively applied to Φ. Thus, in calculus $\mathcal{ER}$ clause normalisation does not cause any decidability problems and can still be employed as a side computation (evoked by rule *Cnf* of Figure 4.2) whenever it appears to be appropriate.

In contrast however to clause normalisation, higher-order unification is undecidable [SG89,

$$\frac{\mathbf{C} \lor [\mathbf{A} \lor \mathbf{B}]^T}{\mathbf{C} \lor [\mathbf{A}]^T \lor [\mathbf{B}]^T}\ \lor^T \qquad \frac{\mathbf{C} \lor [\mathbf{A} \land \mathbf{B}]^F}{\mathbf{C} \lor [\mathbf{A}]^F}\ \lor^F_l \qquad \frac{\mathbf{C} \lor [\mathbf{A} \land \mathbf{B}]^F}{\mathbf{C} \lor [\mathbf{B}]^F}\ \lor^F_r$$

$$\frac{\mathbf{C} \lor [\neg \mathbf{A}]^T}{\mathbf{C} \lor [\mathbf{A}]^F}\ \neg^T \qquad \frac{\mathbf{C} \lor [\neg \mathbf{A}]^F}{\mathbf{C} \lor [\mathbf{A}]^T}\ \neg^F$$

$$\frac{\mathbf{C} \lor [\Pi^\alpha \mathbf{A}]^T \quad X_\alpha \text{ new variable}}{\mathbf{C} \lor [\mathbf{A}\ X]^T}\ \Pi^T$$

$$\frac{\mathbf{C} \lor [\Pi^\alpha \mathbf{A}]^F \quad \mathrm{sk}_\alpha \text{ is a Skolem term for this clause}}{\mathbf{C} \lor [\mathbf{A}\ \mathrm{sk}_\alpha]^F}\ \Pi^F$$

Figure 4.1: The Clause Normalisation Calculus $\mathcal{CNF}$

Sny91] and can thus no longer be employed as a side computation like in the traditional first-order setting. Huet solved the undecidability problem in [Hue72, Hue73a] by delaying unification in his original constraint resolution approach instead of employing it as a filter. This is somewhat unrealistic in practice, as the filter effect is now delayed until the end too, and there are just too many candidates which fail only at the end of the computation. Therefore Kohlhase allows within his sorted variant of Huet's resolution calculus $\mathcal{HORES}$ (see [Koh94b]) for eager unification, i.e., the unification filter is (inspite of its theoretical undecidability) employed as early as possible during the refutation process. This is realistic as undecidable unification problems do rarely occur in practice.

The unsorted variant of $\mathcal{HORES}$ provides the basis for our calculus $\mathcal{ER}$ (as well as for $\mathcal{EP}$ and $\mathcal{ERUE}$). The first modification of calculus $\mathcal{ER}$ with respect to $\mathcal{HORES}$ concerns Skolemsisation: the Skolemisation technique employed by $\mathcal{HORES}$ is not sound. $\mathcal{HORES}$ adds special variable conditions to each clause in which the binding restrictions obtained from the Skolemisation steps are encoded. Consequently, after modifications of a clause (like the renaming of free variables) these variable conditions have to be updated as well. As the employed variable conditions and the updating mechanisms are not strong enough to prevent $\mathcal{HORES}$ from proving obviously invalid statements like *Each function has a fix-point* $(\forall F_{\alpha\to\beta}.\ \exists X_\alpha.\ (F\ X) \doteq X)$, we employ in calculus $\mathcal{ER}$ traditional Skolemisation again. More precisely, we use Miller's sound approach for higher-order Skolemisation [Mil83]. Another, more notational modification belongs to the encoding of unification constraints which we uniformly present as negated equations in all calculi introduced in this thesis. The most important modification of $\mathcal{ER}$ over $\mathcal{HORES}$ is that we add new extensionality rules to the calculus in order to reach Henkin completeness without the need for extensionality axioms. The rules of $\mathcal{HORES}$ that are directly reflected in $\mathcal{ER}$ are the clause normalisation rules presented in Figure 4.1 and basically the resolution and the pre-unification rules as stated in Figure 4.2 — except for the extensionality rules *Equiv* and *Leib*, and partly the extensionality rule *Func*.

The rules of calculus $\mathcal{ER}$ can be divided into the following three groups: clause normal-

$\boxed{\text{Clause Normalisation:}}$ (defined for arbitrary clauses)

$$\frac{\mathcal{D} \quad \mathcal{C} \in \mathcal{CNF}(\mathcal{D})}{\mathcal{C}}\ Cnf$$

$\boxed{\text{Resolution:}}$ (defined on proper clauses only)

$$\frac{[\mathbf{A}]^{\alpha} \lor \mathbf{C} \quad [\mathbf{B}]^{\beta} \lor \mathbf{D} \quad \alpha \neq \beta}{\mathbf{C} \lor \mathbf{D} \lor [\mathbf{A}{=}\mathbf{B}]^{F}}\ Res$$

$$\frac{[\mathbf{A}]^{\alpha} \lor [\mathbf{B}]^{\alpha} \lor \mathbf{C} \quad \alpha \in \{T, F\}}{[\mathbf{A}]^{\alpha} \lor \mathbf{C} \lor [\mathbf{A} \ {=}\mathbf{B}]^{F}}\ Fac$$

$$\frac{[Q_{\gamma}\ \overline{\mathbf{U}^{k}}]^{\alpha} \lor \mathbf{C} \quad \mathbf{P} \in \mathcal{AB}_{\gamma}^{\{\neg,\lor\}\cup\{\Pi^{\beta}|\beta\in\mathcal{T}\}} \quad \alpha \in \{T, F\}}{[Q_{\gamma}\ \overline{\mathbf{U}^{k}}]^{\alpha} \lor \mathbf{C} \lor [Q{=}\mathbf{P}]^{F}}\ Prim$$

$\boxed{\text{Extensional (Pre-)Unification:}}$ (defined for arbitrary clauses)

$$\frac{\mathbf{C} \lor [\mathbf{M}_{\alpha\to\beta}{=}\mathbf{N}_{\alpha\to\beta}]^{F} \quad s_{\alpha} \text{ Skolem term for this clause}}{\mathbf{C} \lor [\mathbf{M}\ s{=}\mathbf{N}\ s]^{F}}\ Func$$

$$\frac{\mathbf{C} \lor [\mathbf{A}_{\alpha\to\beta}\ \mathbf{C}_{\alpha} = \mathbf{B}_{\alpha\to\beta}\ \mathbf{D}_{\alpha}]^{F}}{\mathbf{C} \lor [\mathbf{A} = \mathbf{B}]^{F} \lor [\mathbf{C} = \mathbf{D}]^{F}}\ Dec$$

$$\frac{\mathbf{C} \lor [\mathbf{A}{=}\mathbf{A}]^{F}}{\mathbf{C}}\ Triv \qquad\qquad \frac{\mathbf{C} \lor [X{=}\mathbf{A}]^{F} \quad X \notin \text{free}(\mathbf{A})}{\mathbf{C}_{\{\mathbf{A}/X\}}}\ Subst$$

$$\frac{\mathbf{C} \lor [F_{\gamma}\ \overline{\mathbf{U}^{n}}{=}h\ \overline{\mathbf{V}^{m}}]^{F} \quad \mathbf{G} \in \mathcal{AB}_{\gamma}^{h}}{\mathbf{C} \lor [F{=}\mathbf{G}]^{F} \lor [F\ \overline{\mathbf{U}^{n}}{=}h\ \overline{\mathbf{V}^{m}}]^{F}}\ FlexRigid$$

$$\frac{\mathbf{C} \lor [\mathbf{M}_{o}{=}\mathbf{N}_{o}]^{F}}{\mathbf{C} \lor [\mathbf{M}_{o} \Leftrightarrow \mathbf{N}_{o}]^{F}}\ Equiv \qquad\qquad \frac{\mathbf{C} \lor [\mathbf{M}_{\alpha}{=}\mathbf{N}_{\alpha}]^{F}}{\mathbf{C} \lor [\forall P_{\alpha\to o}.\ P\ \mathbf{M} \Rightarrow P\ \mathbf{N}]^{F}}\ Leib$$

$\mathcal{AB}_{\gamma}^{h}$ is the set of partial bindings of type γ for head h as defined in [SG89]

Figure 4.2: The Extensional Higher-Order Resolution calculus $\mathcal{ER}$

isation rules, resolution rules, and extensional higher-order pre-unification rules. The set of clause normalisation rules are displayed in Figure 4.1 and the resolution and unification rules of $\mathcal{ER}$ in Figure 4.2. For all rules we assume commutativity of $\vee$, symmetry of $=$, and associativity of $\vee$. Furthermore, we assume that the literals of our clauses are always kept in head-normal form. Consequently we suppose that the newly generated or modified clauses are always immediately transformed into head-normal form.[1]

We now discuss the rules of $\mathcal{ER}$ in detail and start with those that are directly imported from $\mathcal{HORES}$. We then describe extensionality rules that are new in $\mathcal{ER}$ and which make $\mathcal{ER}$ Henkin complete.

The higher-order resolution rule *Res* and factorisation rule *Fac* employed in $\mathcal{ER}$ and $\mathcal{HORES}$ obviously differ from their first-order counterparts. Instead of using unification as a filter, which checks the rules applicability and even computes a most general representation of all suitable variable instantiations justifiying the particular resolution or factorisation steps, they add — as suggested by Huet — respective unification constraints to the generated clauses and generally delay the application of unification. Consequently, the search space thereby explodes as any two literals with contrary polarities can be resolved upon and any two literals in a clause with identical polarities can be factorised.

In order to avoid this search space explosion $\mathcal{ER}$ and $\mathcal{HORES}$ allow for eager pre-unification. This is realised by rule *Subst* which propagates (partly) solved unification constraints back to the other literals of the same clause. Thus, the idea is not to delay unification in $\mathcal{ER}$ till an empty clause is derived, but to employ it early and parallel to the overall refutation search, thereby partly regaining the filter properties of unification. Clearly, as higher-order unification and pre-unification is generally undecidable we still cannot prematurely decide all unification problems — but many.

After applying rule *Subst* (or analogously the extensionality rules *Equiv* or *Leib* as introduced below), clause normalisation may become necessary in order to obtain proper clauses again. This is due to the fact that instantiating predicate variables at head positions of some literals, i.e., flexible literal heads, may lead to pre-clauses instead of proper ones. The clause normalisation process is evoked by the application of rule *Cnf*. It performs exhaustive $\mathcal{CNF}$-derivations from a pre-clause $\mathcal{D}$ to a proper clause $\mathcal{C} \in \mathcal{CNF}(\mathcal{D})$ according to the rules presented in Figure 4.1. Thus, the whole derivation according to the $\mathcal{CNF}$-rules is hidden inside the single rule application of *Cnf* in calculus $\mathcal{ER}$.

It is well known for higher-order resolution that a primitive substitution rule (as suggested in [And89]) or a splitting rule (see [Hue72]) is needed, as unification is too weak to compute all necessary instantiations for flexible literal heads.

The primitive substitution rule *Prim* provided by calculus $\mathcal{ER}$ is a variant of the rule suggested in [And89] and is conceptually simpler as Huet's splitting rule. Rule *Prim* allows to instantiate flexible heads of the literals by a partial binding that imitates a logical constant. The important role of this rule can be illustrated by the example $\exists X_o \exists Y_o.\ X \vee Y$ which is obviously a theorem with respect to Henkin semantics. By negation and clause normalisation we obtain the two unit clauses $[X]^F$ and $[Y]^F$. Both clauses consist of exactly one negated literal with a flexible head. Neither resolution nor factorisation is applicable and thus without the primitive substitution rule we cannot find a refutation. The application of *Prim* with the

[1] For the formal proofs in this chapter we do not assume that $\mathcal{ER}$ automatically takes idempotency of $\vee$ into account and does not automatically factorise identical literals in the clause normalisation process, since this eases our argumentations. In practice, however, one is certainly interested to optimise clause normalisation as far as possible.

partial binding $\{X \leftarrow \neg X'\}$ on clause $[X]^F$ results after clause normalisation with rule *Cnf* in $[X']^T$. Thus, by applying rule *Prim* we add important but missing logical structure to our clauses, such that a refutation with the other calculus rules becomes possible.

$\mathcal{HORES}$ is not completene with respect to Henkin semantics. The problem is that despite the primitive substitution rule *Prim*, which in some sense supports higher-order unification algorithm in computing instantiations of variables[2], the unification rules are still too weak to handle the extensionality principles sufficiently. More precisely, higher-order unification as employed in $\mathcal{HORES}$ or in Huet's original approach is a pure syntactically oriented algorithm for unifying terms. But for reaching Henkin completeness we need unification with respect to the theory defined by the extensionality principles, as we are interested to unify terms like $\mathbf{A}_o \wedge \mathbf{B}_o$ and $\mathbf{B}_o \wedge \mathbf{A}_o$ or even $\lambda X_\alpha.\ \mathbf{A}_{\alpha\to o}\ X \wedge \mathbf{B}_{\alpha\to o}\ X$ and $\lambda X_\alpha.\ \mathbf{B}_{\alpha\to o}\ X \wedge \mathbf{A}_{\alpha\to o}\ X$.

Clearly, $\mathcal{HORES}$ as well as the traditional approaches [And71, Hue72] can be made Henkin complete by adding the extensionality axioms to the search space, which is unfeasible in practice.

As a solution to this problem, the calculus $\mathcal{ER}$ adds the three new extensionality rules *Leib*, *Equiv*, and *Func* to the unification rules and thereby avoids the extensionality axioms in the search space. Rule *Leib* (see Figure 4.2) simply instantiates the equality symbol within unification constraints by its Leibniz definition and *Equiv* reflects the extensionality property for truth values in a negative way: if two formulae are not equal, then they are also not equivalent. Rule *Func* analogously reflects functional extensionality: if two functions are not equal then there exists a witness s_α on which these functions differ. To ensure soundness, s_α has to be a new Skolem term that contains all the free variables occurring in the given clause. Why is rule *Func* presented as a new extensionality rule but also as a usual unification rule? The reason is that the pre-unification rules α and η as presented in $\mathcal{HORES}$ already partially realise the negative aspect of the functional extensionality principle:

$$\frac{C \vee [(\lambda X_\alpha.\ \mathbf{A}){=}(\lambda Y_\alpha.\ \mathbf{B})]^F \qquad s_\alpha \text{ Skolem term for this clause}}{C \vee [\mathbf{A}_{\{s/X\}} = \mathbf{B}_{\{s/Y\}}]^F}\ \alpha$$

$$\frac{C \vee [(\lambda X_\alpha.\ \mathbf{A}){=}\mathbf{B}]^F \qquad s_\alpha \text{ Skolem term for this clause}}{C \vee [\mathbf{A}_{\{s/X\}} = (\mathbf{B}\ s)]^F}\ \eta$$

Note that the purely type information based rule *Func* extends and generalises these two rules and thus rule *Func* has the following two meanings in our calculus: If one or both unification terms is a λ-abstraction, it works like the traditional α- and η-rules as, e.g., used in $\mathcal{HORES}$ and [BK98a]. If on the other hand neither of the unification terms is a λ-abstraction rule, *Func* realises the functional extensionality principle.

In cooperation the new extensionality rules connect the unification part of our calculus with the resolution part by allowing for recursive calls of the overall refutation process from within higher-order unification.

We want to point out that none of the three new extensionality rules introduces any flexible literal and even better, they introduce no new free variable at all; even if they heavily increase the search space for refutations, they behave much better — as the experiments (see

[2] Clause normalisation removes logical structure from the input formulas and translates them into the clause structure. The primitive substitution rule on the other hand can always introduce new logical structure for flexible literal heads, which cannot be computed by unification. In this sense primitive substitution supports higher-order unification in a refutation approach.

Section 7.4) with the LEO theorem prover [BK98b, Ben97] showed — than the addition of extensionality axioms in the traditional approaches, which introduces many flexible literals in the refutation process.

One important aspect that is illustrated by the examples in this thesis as well as the examples discussed in [BK98a] is that eager pre-unification becomes essential in $\mathcal{ER}$ and many proofs cannot be found when delaying the unification process until the end.

Another important modification of $\mathcal{ER}$ with respect to $\mathcal{HORES}$ concerns the encoding of unification constraints. In $\mathcal{ER}$ they are encoded as negated equational literals. In this sense a clause $\mathcal{C} \vee [\mathbf{L}^1 = \mathbf{R}^1]^F \vee \ldots \vee [\mathbf{L}^n = \mathbf{R}^n]^F$ can also be read as the implication $([\mathbf{L}^1 = \mathbf{R}^1]^T \wedge \ldots \wedge [\mathbf{L}^n = \mathbf{R}^n]^T) \Rightarrow \mathcal{C}$, i.e., the unification constraints describe the conditions under which clause rest $\mathcal{C}$ holds. Consequently the question arises whether or not resolution and factorisation rules are allowed to be applied on these unification constraints, which look like ordinary literals. In order to obtain a Henkin complete calculus this is not necessary — as the completeness proofs in [BK98a, BK97b] and the alternative one in this thesis show. Consequently the unification constraints do not necessarily have to be encoded as negative equational literals, any other form will work as well. But the encoding of unification constraints as negated equational literals becomes essential for the extensional higher-order RUE-resolution calculus presented in Chapter 6 and the extensional higher-order paramodulation calculus presented in Chapter 5.

We briefly sum up the particular modifications with respect to [BK98a]:

- The unification rules α and η employed in [BK98a] are avoided as they are subsumed by rule *Func*.
- Instead of employing clause normalisation in the definitions of rule *Subst* and the extensionality rules *Equiv* and *Leib* we add the extra clause normalisation rule *Cnf* to the calculus.
- We slightly modify the decomposition rule *Dec*. This modification is illustrated in detail by Example $\mathbf{E}^{Dec}$ in Subsection 8.2.

We will present in this chapter an alternative proof for the Henkin completeness of calculus $\mathcal{ER}$ to the one given in [BK98a, BK97b]. The motivation for this new proof is threefold:

- We have slightly modified the calculus $\mathcal{ER}$ in this thesis.
- The completeness proofs of the new, further extended calculi $\mathcal{EP}$ (extensional higher-order paramodulation) and $\mathcal{ERUE}$ (extensional higher-order RUE-Resolution) are carried out analogously, such that many lemmata can be either directly reused or with minor modifications.
- The lifting lemma in the completeness proofs in [BK98a, BK97b] builds upon an argument also employed in [Koh94b] which uses a quite complicated notion of clause isomorphisms susceptible to errors. In this thesis we present a lifting argument that omits the notion of clause isomorphisms. This is possible as we analyse a generalised resolution calculus $\mathcal{ER}_{fc}$ instead of $\mathcal{ER}$. This enriched calculus additionally employs the instantiation guessing *FlexFlex*-rule (see Figure 4.3) and applies the single clause normalisation rules instead of grouping them into exhaustive clause normalisation chains with rule *Cnf*. Consequently, in Subsection 4.5 we will discuss the theorem equivalence between $\mathcal{ER}_{fc}$ and $\mathcal{ER}$.

An important convention for this and the following chapters concerns α-equality of clauses and the arity of Skolem terms:

*Remark 4.1 (**Equality of clauses**).* In resolution based theorem proving one usually assumes all clauses to be variable disjoint. In practice this is achieved by automatically renaming the variables within each newly generated clause. In this thesis we implicitly use this convention, too.

Another implicit convention concerns the Skolem terms. We briefly illustrate this aspect by an example. Assume that the following pre-clause is given:

$$\mathcal{C}_1 : [\forall X_\iota.\ p_{\iota\to o}\ X_\iota\ Y_\iota]^T \vee [\forall Z_\iota.\ q_{\iota\to o}\ Z]^F$$

Clause normalisation either leads to

$$\mathcal{C}_2 : [p_{\iota\to o}\ X_\iota\ Y_\iota]^T \vee [q_{\iota\to o}\ (s^1_{\iota\to\iota}\ Y_\iota)]^F \quad \text{or to} \quad \mathcal{C}_3 : [p_{\iota\to o}\ X_\iota\ Y_\iota]^T \vee [q_{\iota\to o}\ (s^2_{\iota\to\iota\to\iota}\ X_\iota\ Y_\iota)]^F$$

where $(s^1_{\iota\to\iota}\ Y_\iota)$ and $(s^2_{\iota\to\iota\to\iota}\ X_\iota\ Y_\iota)$ are new Skolem terms. The first clause $\mathcal{C}_2$ is the result of applying rule Π^T first and Π^F to the result, whereas the second clause $\mathcal{C}_3$ is the result of applying first Π^F and then Π^T. Both results differ with respect to the arity of the new Skolem terms. It is well known for refutation approaches that each refutation using only one of these clauses can be analogously carried out with the other one. For a discussion of this Skolemisation aspect in the context of of sequent calculi we refer to [AMS98]. In the following we will therefore ignore the different arities of Skolem terms caused by switching the order of single applications of clause normalisation rules (switching the order of clause normalisation rules will be employed in some of the proofs in this thesis).

4.2 Basic Definitions

Instead of a direct proof of Henkin completeness for $\mathcal{ER}$, we first analyse the slightly enriched calculus $\mathcal{ER}_{fc}$. Aside from the unfolding of exhaustive clause normalisation derivations this calculus provides the well known *FlexFlex* unification rule displayed in Figure 4.3. In case a clause contains a *flex-flex*-unification constraint this rule allows to guess an instantiation for one of the flexible heads such that the unification process can proceed with its eager unification attempts. It was already pointed out by Huet [Hue72] that in practice we are interested in avoiding this possibly infinitely branching rule (there may be infinitely many constants in the signature) and it turned out that within a refutation approach one can in fact avoid the *FlexFlex*-rule and delay the operations on *flex-flex*-constraints until one of the head variables gets bound. However, employing this additional rule within the lifting lemma eases the proofs as it turns our eager pre-unification approach into an eager unification approach. This allows to omit the clause isomorphisms that are needed in the respective proofs in [Koh94b]. The motivation for the unfolding of exhaustive clause normalisation derivation by rejecting rule *Cnf* and lifting the clause normalisation rules to calculus level is analogous: We want to ease the proofs in this section and especially the analogous but slightly more complicated ones for the extensional higher-order paramodulation calculus $\mathcal{EP}$ and extensional higher-order RUE-resolution calculus $\mathcal{ERUE}$ in the following sections.

A formal proof for the admissibility of rule *FlexFlex* has not been carried out yet, but evidence is given by the direct proof of Henkin completeness in [BK98a] (which admittedly lacks a bit of transparency and clarity within the lifting argument) and the case studies with

$$\frac{\mathbf{C} \vee [F_{\overline{\gamma^n} \to \alpha}\ \overline{\mathbf{U}^n} = H_{\overline{\delta^m} \to \alpha}\ \overline{\mathbf{V}^m}]^F \quad \mathbf{G} \in \mathcal{AB}^h_{\overline{\gamma^n} \to \alpha} \text{ for a } h_\tau \in \mathcal{C}_\tau}{\mathbf{C} \vee [F\ \overline{\mathbf{U}^n} = H\ \overline{\mathbf{V}^m}]^F \vee [F = \mathbf{G}]^F}\ \mathit{FlexFlex}$$

Figure 4.3: The *FlexFlex* unification rule

the LEO prover [BK98b]. Actually, there is no example known to the author that requires the application of rule *FlexFlex*.

Definition 4.2 (Clause Normalisation). The calculus $\mathcal{CNF}$ consists of the clause normalisation rules displayed in Figure 4.1. We assume that the result of each rule application is transformed into head-normal form[3]. By applying these rules exhaustively to a pre-clause $[\mathbf{A}]^\alpha$ ($\alpha \in \{T, F\}$) one can derive the set $\mathcal{CNF}([\mathbf{A}]^\alpha)$ of proper clauses derivable from $[\mathbf{A}]^\alpha$.

Lemma 4.3 (Soundness of $\mathcal{CNF}$). *The rules $\mathcal{CNF}\backslash\{\Pi^F\}$ preserve validity and the rule Π^F preserves satisfiability with respect to Henkin semantics.*

Proof: The proofs for the rules in $\mathcal{CNF}\backslash\{\Pi^F\}$ are analogous to the first-order case and Skolemisation has been corrected for higher-order logic by Miller in [Mil83, Mil91, Mil92]. □

Definition 4.4 (Unification). We define the following two calculi for higher-order unification and higher-order pre-unification:

$\mathcal{UNI}$ The calculus $\mathcal{UNI}$ consists of the pre-unification rules *Triv, Func, Dec, FlexRigid* and *Subst* as presented in Fig 4.2. We assume that the result of each rule application is transformed into head-normal form.[4] These rules form a quite close variant of the higher-order pre-unification calculus discussed in [Koh94b] and [SG89, Sny91]. We already mentioned that rule *Func*, which simply applies the functional extensionality principle, subsumes the rules α and η used in [BK98a] and [Koh94b].

$\mathcal{UNI}_f$ The calculus $\mathcal{UNI}_f$ is defined as $\mathcal{UNI} \cup \{\mathit{FlexFlex}\}$.

Theorem 4.5 (Higher-Order Unification and Pre-Unification).

Soundness $\mathcal{UNI}_f$ *(resp. $\mathcal{UNI}$) is a sound calculus for higher-order unification (resp. higher-order pre-unification). More precisely, for each derivation $\Delta : \mathbf{E} := [\mathbf{L}_1 = \mathbf{R}_1]^F \vee \ldots \vee [\mathbf{L}_n = \mathbf{R}_n]^F \vdash_{\mathcal{UNI}} E'$ (resp. $\Delta : \mathbf{E} \vdash_{\mathcal{UNI}} E'$), where E' is in solved form and corresponds with unifier (pre-unifier) σ, holds that σ is a unifier (resp. pre-unifier) of $\{\mathbf{L}_1 =^? \mathbf{R}_1, \ldots, \mathbf{L}_n =^? \mathbf{R}_n\}$.*

Completeness $\mathcal{UNI}_f$ *(resp. $\mathcal{UNI}$) is a complete calculus for higher-order unification (resp. higher-order pre-unification). More precisely, for each higher-order unification problem $\{\mathbf{L}_1 =^? \mathbf{R}_1, \ldots, \mathbf{L}_n =^? \mathbf{R}_n\}$ with unifier (resp. pre-unifier) σ, there exists a derivation $\Delta : \mathbf{E} := [\mathbf{L}_1 = \mathbf{R}_1]^F \vee \ldots \vee [\mathbf{L}_n = \mathbf{R}_n]^F \vdash_{\mathcal{UNI}} E'$ (resp. $\Delta : \mathbf{E} \vdash_{\mathcal{UNI}} E'$), such that E' is in solved form and corresponds with unifier (resp. pre-unifier) σ.*

[3] Remember the special definition of head-normal form for unification constraints as given in Chapter 2.1.

[4] Note that this is the main difference to the pre-unification rules presented in [SG89] which presupposes that all results are reduced to $\beta\eta$-normal form.

Proof: We will not present a formal proof here and instead refer to [Koh94b]. The only difference of our rules to the respective ones used in [Koh94b] is, that the latter consider sorts as well and employ a extra-logical form of Skolemisation. Note that our set of rules furthermore only slightly modifies the set of unification rules discussed in [SG89, Sny91]. □

The first complete set of transformations for higher-order unification was defined in [Pie73, Hue73a] and undecidability of higher-order unification was first discussed in [Hue73b]. Huet then introduced higher-order pre-unification in [Hue75]. For a modern presentation of higher-order unification and pre-unification we refer to [SG89, Sny91]. Sorted higher-order unification and pre-unification is discussed in [Koh94b].

As the calculus $\mathcal{UNI}_f$ realises higher-order unification and as rule *Subst* allows to propagate solutions back to the non-unification constraints of a clause we get the following corollary.

Corollary 4.6 (Higher-Order Unification). *Let $\mathcal{C} \vee E$ be a clause with unification constraints E. Then for each unifier σ of E we have that $\mathcal{C} \vee E \vdash_{\mathcal{UNI}_f} \mathcal{C}_\sigma$.*

Definition 4.7 (Extensional Higher-Order Resolution).

$\mathcal{ER}$ The calculus $\mathcal{ER}$ consists of the following inference rules displayed in Figure 4.2, i.e., $\mathcal{ER} := \{Cnf, Res, Fac, Prim\} \cup \mathcal{UNI} \cup \{Leib, Equiv\}$

$\mathcal{ER}_f$ The extension $\mathcal{ER}_f$ of calculus $\mathcal{ER}$ that employs full higher-order unification instead of higher-order pre-unification is defined as $\mathcal{ER}_f := \mathcal{ER} \cup \{FlexFlex\}$.

$\mathcal{ER}_{fc}$ The calculus $\mathcal{ER}_{fc}$ that employs stepwise instead of exhaustive clause normalisation is defined by $\mathcal{ER}_{fc} := (\mathcal{ER}_f \backslash \{Cnf\}) \cup \mathcal{CNF}$.

A set of formulae Φ is *refutable* in calculus $R \in \{\mathcal{ER}, \mathcal{ER}_f, \mathcal{ER}_{fc}\}$, iff there is a derivation $\Delta : \Phi_{cl} \vdash_R \square$, where $\Phi_{cl} := \{[\mathbf{F}_{\downarrow_h}]^T | \mathbf{F} \in \Phi\}$ is the set obtained from Φ by simple pre-clausification. We remark again, that unification constraints are treated as special literals, which are only accessible to the unification rules.

Remark 4.8 ***(General Higher-Order*** **E*****-Unification)***. The extensional higher-order resolution calculus $\mathcal{ER}$ ($\mathcal{ER}_f$ or $\mathcal{ER}_{fc}$) can also be viewed as a test calculus for general higher-order E-pre-unifiability (E-unifiability): Assume an arbitrary set of equations $\mathbf{E}_1 \ldots \mathbf{E}_n$ describing a theory E and an E-unification problem $\mathbf{T}_1 = \mathbf{T}_2$ is given. If we pass clauses $[\mathbf{E}_1]^T \ldots [\mathbf{E}_n]^T$ and $[\mathbf{T}_1 = \mathbf{T}_2]^F$ as an input problem to our calculus, then the calculus $\mathcal{ER}$ tests if the unification constraint $[\mathbf{T}_1 = \mathbf{T}_2]^F$ is solvable with respect to theory E enriched by the extensionality properties. The overall answer substitution computed by the refutation is obviously also an answer substitution to our E-unification problem.

Theorem 4.9 (Soundness of Extensional Higher-Order Resolution). *The calculi $\mathcal{ER}$, $\mathcal{ER}_f$ and $\mathcal{ER}_{fc}$ are sound for Henkin semantics, i.e., let Φ be a set of formulae, such that $\Phi \vdash_R \square$, then Φ is unsatisfiable with respect to Henkin semantics (and consequently standard semantics).*

Proof: We already know by lemma 4.3 that the rules in $\mathcal{CNF}$ either preserve validity or satisfiability with respect to Henkin models and thus the latter also holds for compound rule *Cnf*. The extensionality principles are valid in Henkin models (see Lemmata 2.37 and 2.43) and thus the extensionality rules *Leib* and *Equiv* preserve validity wrt. Henkin semantics, whereas rule *Func* only preserves satisfiabilty as it immediately applies Skolemisation after employing the functional extensionality principle. All remaining unification rules as well as the resolution rules *Res*, *Fac*, and *Prim* can easily be shown to preserve validity with respect to Henkin models. Now the assertion follows from the well known result that preservation of satisfiability ensures soundness within a refutation approach. □

The following lemma states that each non-proper clause $\mathcal{C}$ in a clause set Φ can be replaced by its corresponding set of proper clauses $\mathcal{CNF}(\mathcal{C})$, without affecting the set of derivable proper clauses. We want to remark that this lemma implicitly employs the convention of Remark 4.1. This lemma is applied in the completeness proof of calculus $\mathcal{ER}_{fc}$, more precisely, in Lemma 4.15, which is then applied in the completeness proof to verify the consistency properties $\nabla_\vee$ and $\nabla_\mathfrak{b}$, and to show saturatedness. In the completeness proofs for $\mathcal{EP}_{fc}$ and $\mathcal{ERUE}_{fc}$, analoga of this lemma will be additionally employed to verify the abstract consistency property $\nabla_\mathfrak{e}^s$ (again indirect within the proofs of respective lemmata). The Lemma can be proven for all three calculi, but we will need it only for calculus $\mathcal{ER}_{fc}$.

Lemma 4.10 (Proper Derivations). *For each clause set Φ, clause $\mathcal{C}$, and proper clause $\mathcal{C}'$, such that $\Phi * \mathcal{C} \vdash_{\mathcal{ER}_{fc}} \mathcal{C}'$, we have $\Phi \cup \mathcal{CNF}(\mathcal{C}) \vdash_{\mathcal{ER}_{fc}} \mathcal{C}'$.*

Proof: The proof is by induction on the length of the derivation $\Phi * \mathcal{C} \vdash_{\mathcal{ER}_{fc}} \mathcal{C}'$. In the base case ($n \equiv 0$) we know that $\mathcal{C}' \in \Phi$ as $\mathcal{C}'$ must be different from $\mathcal{C}$. Thus, the assertion holds trivially. In the induction step ($n > 0$) we consider the first step in derivation $\Phi * \mathcal{C} \vdash^r \Phi * \mathcal{C} * \mathcal{D} \vdash_{\mathcal{ER}_{fc}} \mathcal{C}'$, where $\mathcal{D}$ is a clause. If $\mathcal{C}$ is not a premise clause for the application of rule r then the assertion follows immediately by induction hypotheses. Thus, let us assume that $\mathcal{C}$ is a premise clause for the application of rule r. By induction hypotheses applied to clause $\mathcal{D}$ we get that $(\Phi * \mathcal{C}) \cup \mathcal{CNF}(\mathcal{D}) \vdash_{\mathcal{ER}_{fc}} \mathcal{C}'$. As the resolution rules $Res, Prim, Fac$ are only defined on proper clauses, we know that the only rules that are possible for r are the unification rules in $\mathcal{UNI}_f$ and the rule *Cnf*. It is easy to check that in all those cases we have that $\mathcal{CNF}(\mathcal{C}) \vdash_{\mathcal{ER}_{fc}} \mathcal{CNF}(\mathcal{D})$. And thus, we finally get that $\Phi \cup \mathcal{CNF}(\mathcal{C}) \vdash_{\mathcal{ER}_{fc}} \Phi \cup \mathcal{CNF}(\mathcal{C}) \cup \mathcal{CNF}(\mathcal{D}) \vdash_{\mathcal{ER}_{fc}} \mathcal{C}'$. □

4.3 Lifting Properties

This subsection examines some lifting properties of the calculus $\mathcal{ER}_{fc}$. We shall prove a lifting argument for all clause normalisation rules in $\mathcal{CNF}$ before we then present the lifting lemma for $\mathcal{ER}_{fc}$. The lifting lemma for calculus $\mathcal{ER}_{fc}$, which states that each derivation in $\mathcal{ER}_{fc}$ performed on an instantiated level has a direct counterpart on the uninstantiated level as well, turns out to be quite trivial. The reason is that $\mathcal{ER}_{fc}$ provides rule *FlexFlex*, which can be employed in addition to rule *Prim* to introduce the necessary instantiations on the abstract level whenever the replay of the derivation given on the instantiated level is blocked caused by missing clause or unification term structure. Furthermore, clause normalisation rules are not grouped into one single rule *Cnf* but treated as single inference rules of their own, which also eases the particular argumentations.

Lemma 4.11 (Lifting Lemma for Clause Normalisation). *Let $\mathcal{D}_1, \mathcal{D}_2$ be clauses and σ be a substitution. For each derivation $\Delta_1 : (\mathcal{D}_1)_\sigma \vdash^1_{\mathcal{CNF}} \mathcal{D}_2$ exists a clause $\mathcal{D}_3$, a substitution δ, and a derivation $\Delta_2 : \mathcal{D}_1 \vdash_{\mathcal{ER}_{fc}} \mathcal{D}_3$, such that $(\mathcal{D}_3)_{\sigma \circ \delta} \equiv \mathcal{D}_2$.*

Proof: The proof is by case distinction on the rules in $\mathcal{CNF}$. As all cases are analogous we only consider the $\vee_l^F$ rule here. In this case we have that $(\mathcal{D}_1)_\sigma := (L_1)_\sigma \vee \ldots \vee (L_n)_\sigma \vee$

$[\mathbf{A}_\sigma \vee \mathbf{B}_\sigma]^F$ and $\mathcal{D}_2 := (L_1)_\sigma \vee \ldots \vee (L_n)_\sigma \vee [\mathbf{A}_\sigma]^F$ for some literals $(L_i)_\sigma$, $1 \leq i \leq n$, and terms $\mathbf{A}_\sigma, \mathbf{B}_\sigma$. We now consider the possible structure of the focused literal in the uninstantiated clause $\mathcal{D}_1 : L_1 \vee \ldots \vee L_n \vee [\mathbf{C}]^F$.

In case $[\mathbf{C}]^F \equiv [\mathbf{A} \vee \mathbf{B}]^F$ we can obviously apply $\vee_l^F$ leading to $\mathcal{D}_3 : L_1 \vee \ldots \vee L_n \vee [\mathbf{A}]^F$, such that the assertion follows trivially.

Otherwise we have $[\mathbf{C}]^F \equiv [H\ \overline{\mathbf{U}^m}]^F$, such that $(H\ \overline{\mathbf{U}^m})_\sigma \equiv \mathbf{A}_\sigma \vee \mathbf{B}_\sigma$, as either (i) $H_\sigma \equiv \lambda\overline{X^m}.\ \mathbf{L} \vee \mathbf{R}$, with $(\mathbf{L}[\overline{\mathbf{U}^m}/\overline{X^m}])_\sigma \equiv \mathbf{A}_\sigma$ and $(\mathbf{R}[\overline{\mathbf{U}^m}/\overline{X^m}])_\sigma \equiv \mathbf{B}_\sigma$, or (ii) $H_\sigma \equiv \lambda\overline{X^m}.\ X^n\ \mathbf{T}$ (for $1 \geq n \geq m$), with $((X^n\ \mathbf{T})[\overline{\mathbf{U}^m}/\overline{X^m}]))_\sigma \equiv \mathbf{A}_\sigma \vee \mathbf{B}_\sigma$.

Case (i): we now apply rule *Prim* to clause $\mathcal{D}_1$ with general binding $\lambda\overline{X^m}.\ (H^1\ \overline{X^m}) \vee (H^2\ \overline{X^m})$ for predicate variable H (H^1, H^2 are new predicate variables of appropriate type) and obtain the clause

$$\mathcal{D}_4 : L_1 \vee \ldots \vee L_n \vee [H = \lambda\overline{X^m}.\ (H^1\ \overline{X^m}) \vee (H^2\ \overline{X^m})]^F$$

With rule *Subst* we get clause

$$\mathcal{D}_5 : (L_1)_\tau \vee \ldots \vee (L_n)_\tau \vee [(H^1\ \overline{\mathbf{U}^m}) \vee (H^2\ \overline{\mathbf{U}^m})]^F$$

where $\tau := [\lambda\overline{X^m}.\ (H^1\ \overline{X^m}) \vee (H^2\ \overline{X^m})/H]$. Now rule $\vee^F$ is applicable which leads to clause $\mathcal{D}_3 : (L_1)_\tau \vee \ldots \vee (L_n)_\tau \vee [(H^1\ \overline{\mathbf{U}^m})]^F$. It is easy to verify that τ is more general than σ (beacause of the flexible variables H^1 and H^2) and that hence there must be a substitution γ that appropriately instantiates the new predicate variables H^1 and H^2 and that coincides with σ on all other variables, such that we have $(L_i)_{\gamma\circ\tau} \equiv (L_i)_\sigma$, $(H^1\ \overline{\mathbf{U}^m})_{\gamma\circ\tau} \equiv \mathbf{A}_\sigma$. This finally proves the assertion as $(\mathcal{D}_3)_{\gamma\circ\tau} \equiv \mathcal{D}_2$.

Case (ii): In this case the assertion follows analogously if we apply rule *Prim* with the projection binding $\lambda\overline{X^m}.\ X^n\ (H\ \overline{X^m})$. □

We are now ready to prove the lifting lemma for calculus $\mathcal{ER}_{fc}$. The main result (a special case of statement 4.12(2)) is that for any substitution σ and clause set Φ holds that Φ is refutable in calculus $\mathcal{ER}_{fc}$ provided that Φ_σ is. Here we even prove a more general result, stating that *lifting* is even possible for general derivations and not only for refutations.

Lemma 4.12 (Lifting Lemma for $\mathcal{ER}_{fc}$). *Let Φ be a set of clauses, $\mathcal{D}_1$ be a clause and σ a substitution. We have that:*

1. *For each derivation $\Delta_1 : \Phi_\sigma \vdash^1_{\mathcal{ER}_{fc}} \mathcal{D}_1$ there exists a substitution δ, a clause $\mathcal{D}_2$ and a derivation $\Delta_2 : \Phi \vdash_{\mathcal{ER}_{fc}} \mathcal{D}_2$, such that $(\mathcal{D}_2)_\delta \equiv \mathcal{D}_1$.*

2. *For each derivation $\Delta_1 : \Phi_\sigma \vdash_{\mathcal{ER}_{fc}} \mathcal{D}_1$ there exists a substitution δ, a clause $\mathcal{D}_2$ and a derivation $\Delta_2 : \Phi \vdash_{\mathcal{ER}_{fc}} \mathcal{D}_2$, such that $(\mathcal{D}_2)_\delta \equiv \mathcal{D}_1$.*

Proof:

(1) The proof is by case distinction on all rules in $\mathcal{ER}_{fc}$ and in all cases we construct a derivation Δ_2 as required.

Res Assume the first step in Δ_1 employs resolution rule *Res* to clauses $[\mathbf{A}_\sigma]^\alpha \vee C_\sigma$ and $[\mathbf{B}_\sigma]^\beta \vee D_\sigma$ with resolvent $\mathcal{D}_1 : C_\sigma \vee D_\sigma \vee [\mathbf{A}_\sigma{=}\mathbf{B}_\sigma]^F$. Then an analogous resolution step is possible between $[\mathbf{A}]^\alpha \vee C$ and $[\mathbf{B}]^\beta \vee D$ leading to resolvent $\mathcal{D}_2 : C \vee D \vee [\mathbf{A}{=}\mathbf{B}]^F$. We trivially have that $(\mathcal{D}_2)_\sigma \equiv \mathcal{D}_1$.

Prim Assume the first step in Δ_1 employs rule *Prim* to a flexible literal in a clause $C_\sigma \lor [H\ \overline{(\mathbf{A}_\sigma)^n}]^\alpha$ leading to clause $\mathcal{D}_1 : C_\sigma \lor [H\overline{(\mathbf{A}_\sigma)^n}]^\alpha \lor [H = \mathbf{G}]^F$, where G is a general binding for variable H imitating a logical connective. Then an analogous proof step with an identical partial binding is possible on the uninstantiated clause $C \lor [H\ \overline{\mathbf{A}^n}]^\alpha$ leading to $\mathcal{D}_2 : C \lor [H\ \overline{\mathbf{A}^n}]^\alpha \lor [H = \mathbf{G}]$ such that $(\mathcal{D}_2)_\sigma \equiv \mathcal{D}_1$.

Fac This case is analogous to *Res*.

Triv, *Dec*, *Func*, *FlexFlex*, *Leib*, *Equiv* These cases are all analogous to *Prim*. Note that $=$ is a special symbol not available in the signature and thus the abstract literal must also have head $=$. Therefore, all these rules must be applicable on the abstract level as well.

Subst In this case the ground literal is of form $[X = \mathbf{A}_\sigma]^F$. If X_α is of base type, the corresponding abstract literal obviously must have form $[Y_\alpha = \mathbf{A}]^F$ for a variable Y. Then the assertion follows immediately. If $X_{\alpha\to\beta}$ is of functional type the corresponding abstract literal can principically also be of form (i) $[Y_{\gamma\to(\alpha\to\beta)}\ \mathbf{B}_\gamma = \mathbf{A}]^F$ or (ii) $[\mathbf{C}_{\gamma\to(\alpha\to\beta)}\ \mathbf{B}_\gamma = \mathbf{A}]^F$, where $\mathbf{C}$ is a λ-abstraction. Case (ii) can actually be excluded due to our special head-normal form convention of unification constraints. In case (i) rule *Subst* can not immediately be replayed at abstract level. But in a straightforward derivation which employs the rules *FlexRigid* or *FlexFlex* one can finally subsequently unify apply a substitution that is identical or more general than $[\mathbf{A}_\sigma/X]$. E.g., let the ground derivation be $\mathcal{C}_\sigma : [X_{\iota\to o}\ b]^T \lor [X_{\iota\to o} = \lambda Z_\iota.\ p_{\iota\to o}\ Z]^F \vdash^{Subst} \mathcal{D}_1 : [p\ b]^T$, and let $\mathcal{C} : [(Y_{\iota\to\iota\to o}\ a_\iota)\ b]^T \lor [(Y\ a) = \lambda Z_\iota.\ P_{\iota\to o}\ Z]^F$ such that $\sigma \equiv [\lambda W_\iota.\ X_{\iota\to o}/Y, p/P]$. We consider the following derivation on the abstract level (s_ι is a Skolem term, and H^1 and H^2 are new free variables of apropriate type): $\mathcal{C} \vdash^{Func} [(Y\ a)\ b]^T \lor [(Y\ a)\ s = P\ s]^F \vdash^{FlexFlex} [(Y\ a)\ b]^T \lor [(Y\ a)\ s = P\ s]^F \lor [P = \lambda V_\iota.\ p\ (H^1_{\iota\to\iota}\ V)]^F \vdash^{Subst} [(Y\ a)\ b]^T \lor [(Y\ a)\ s = p\ (H^1\ s)]^F \vdash^{FlexRigid} [(Y\ a)\ b]^T \lor [(Y\ a)\ s = p\ (H^1\ s)]^F \lor [Y = \lambda U_\iota.\ \lambda V_\iota.\ p\ (H^2_{\iota\to\iota\to o}\ U\ V)]^F \vdash^{Subst} [p\ (H^2\ a\ b)]^T \lor [p(H^2\ a\ s) = p\ (H^1\ s)]^F \vdash^{Dec,Triv} [p\ (H^2\ a\ b)]^T \lor [(H^2\ a\ s) = (H^1\ s)]^F \vdash^{FlexFlex} [p\ (H^2\ a\ b)]^T \lor [(H^2\ a\ s) = (H^1\ s)]^F \lor [H^1 = \lambda X_\iota.\ X]^F \vdash^{Subst} [p\ (H^2\ a\ b)]^T \lor [(H^2\ a\ s) = s]^F \vdash^{FlexRigid} [p\ (H^2\ a\ b)]^T \lor [(H^2\ a\ s) = s]^F \lor [H^2 = \lambda U_\iota.\ \lambda V_\iota.\ V]^F \vdash^{Subst,Triv} \mathcal{D}_2 : [p\ b]^T$. Thus, the subsequent instantiations computed on the abstract level in our example lead to clause $\mathcal{D}_2$, i.e., the result of the substitution step on the ground level.

FlexRigid Whereas the abstract literal must have head $=$ it is probably a *FlexFlex*-constraint such that we cannot immediately replay the *FlexRigid* step. In this case we employ the rules *FlexFlex* and *Subst* in order to introduce the corresponding rigid head of the instantiated literal. Thereby the *FlexRigid* step performed on the instantiated level becomes possible on the abstract level as well. It is easy to verify that the resulting clause on the abstract level is more general than the instantiated counterpart.

$r \in \mathcal{CNF}$ By clause normalisation lifting lemma 4.11.

(2) The proof is by induction on the length of derivation Δ_1. The base case is trivial and in the induction step we first employ statement (1) and then the induction hypothesis. □

4.4 Completeness

We now focus on the Henkin completeness proof for the calculus $\mathcal{ER}_{fc}$. Towards this end we first prove a lemma stating that reduction to head-normal form is sufficient in calculus $\mathcal{ER}_{fc}$ (note our special definition of head-normal form for unification constraints as described in Chapter 2.1). A second lemma then provides some important refutational properties of $\mathcal{ER}$. Finally in theorem 4.16 we show that the set of propositions that cannot be refuted in calculus $\mathcal{ER}_{fc}$ defines an abstract consistency property for Henkin models as defined in 3.6. The latter entails Henkin completeness for $\mathcal{ER}_{fc}$ by theorem 3.29.

Lemma 4.13 (Head-Normal Form). *Let Φ be a set of clauses. If $\Delta : \Phi{\downarrow_{\beta\eta}} \vdash_{\mathcal{ER}_{fc}} \Box$ then $\Delta' : \Phi{\downarrow_h} \vdash_{\mathcal{ER}_{fc}} \Box$*

Proof: The proof is by induction on the length n of Δ and the base case ($n \equiv 0$) is trivial. In the step case ($n > 0$) we consider the first derivation step in $\Phi{\downarrow_{\beta\eta}} \vdash^r \Phi' \vdash \Box$. In case $r \in \{Res, Fac\}$ the assertion follows immediately by induction hypotheses as both rules are applicable independently from the structure of the literals atoms we focus on. In case $r \in \{Prim\} \cup \mathcal{CNF}$ the heads of the atoms obviously play an important role. But note that the head symbols in the $\beta\eta$-normal form and the head-normal form coincide, such that rule r is applicable in the head-normal form case as well. Therefore, we again get the assertion by induction hypotheses. In case $r \in \mathcal{UNI}$ our special definition of head-normal form for unification constraints (which requires both hand sides of the constraint to be reduced to head-normal form) ensures the assertion with an analogous argument as in the previous case. (Note that the other direction of this lemma, which we do not need in this thesis, can be proven analogously.) □

Remark 4.14 ***(Head-Normal Form).*** It is not essential for our calculi whether we keep our clauses and literals in head-normal form or $\beta\eta$-normal form. The reason why we have chosen reduction to head-normal form is that we expect that reduction to head-normal form is less expensive in practice (which has not been investigated yet). One can also choose $\beta\eta$-normal form. This variation will hardly influence any of the further discussions and theorems in this thesis.

Lemma 4.15. *Let Φ be a set of clauses and $\mathbf{A}, \mathbf{B}$ be formulae. It holds:*

1. *If $\Phi * [\mathbf{A}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ and $\Phi * [\mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$, then $\Phi * [\mathbf{A} \lor \mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$.*
2. *If $\Phi * [\mathbf{A}]^T * [\mathbf{B}]^F \vdash_{\mathcal{ER}_{fc}} \Box$ and $\Phi * [\mathbf{A}]^F * [\mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$, then $\Phi * [\mathbf{A} \Leftrightarrow \mathbf{B}]^F \vdash_{\mathcal{ER}_{fc}} \Box$.*

Proof: (1) Without loss of generality we assume that the pre-clauses $[\mathbf{A}]^T$ and $[\mathbf{B}]^T$ are variable-disjoint. We can replay derivation $\Phi * [\mathbf{A}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ in context $\Phi * [\mathbf{A} \lor \mathbf{B}]^T$, such that we get for each clause $\mathcal{C} \in \mathcal{CNF}([\mathbf{B}]^T)$ $\Phi * [\mathbf{A} \lor \mathbf{B}]^T \vdash_{\mathcal{CNF}} [\mathbf{A}]^T \lor \mathcal{C} \vdash_{\mathcal{ER}_{fc}} \mathcal{C} \lor E$ for a set of *FlexFlex* constraints E containing no variables occurring in $[\mathbf{B}]^F$ or $\mathcal{C}$. By corollary 4.6 we know that each unifier σ of E can be derived in $\mathcal{UNI}_f$ and thus we get for each unifier σ of clause $\mathcal{C} \lor E$ that $\mathcal{C} \lor E \vdash_{\mathcal{UNI}_f} \mathcal{C}_\sigma$. Now the assertion follows by lemma 4.10 as $\Phi * [\mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ and as the domain of σ contains none of the free variables in $\mathcal{C}$, such that $\mathcal{C}_\sigma \equiv \mathcal{C}$ for each clause $\mathcal{C} \in \mathcal{CNF}([\mathbf{B}]^T)$.

(2) This statement (which is also well known from first-order resolution) can be proven by a tedious but straightforward computation which we only roughly sketch here. From the two assumptions we get by lemma 4.10 that (i) $\Phi * \mathcal{CNF}([\mathbf{A}]^T) * \mathcal{CNF}([\mathbf{B}]^F) \vdash_{\mathcal{ER}_{fc}} \Box$ and (ii) $\Phi * \mathcal{CNF}([\mathbf{A}]^F) * \mathcal{CNF}([\mathbf{B}]^T) \vdash_{\mathcal{ER}_{fc}} \Box$. The idea now is to apply exhaustive clause normalisation to the clause $[\mathbf{A} \Leftrightarrow \mathbf{B}]^F$ and then to show that (iii) $\Phi * \mathcal{CNF}([\mathbf{A} \Leftrightarrow \mathbf{B}]^F) \vdash_{\mathcal{ER}_{fc}} \Box$. Note that $\mathcal{CNF}([\mathbf{A} \Leftrightarrow \mathbf{B}]^F) := \{\mathcal{C} \lor \mathcal{D} | \mathcal{C} \in \mathcal{CNF}([\mathbf{A}]^\alpha)\ \mathit{and}\ \mathcal{D} \in \mathcal{CNF}([\mathbf{B}]^\alpha)\ \mathit{for}\ \alpha \in \{T, F\}\}$. Thus, the task is to show that (iii) is a consequence of (i) and (ii), which is possible, e.g., by simultaneous induction on the structure of $\mathbf{A}$ and $\mathbf{B}$. □

Theorem 4.16 (Completeness of $\mathcal{ER}_{fc}$). *The calculus $\mathcal{ER}_{fc}$ is complete with respect to Henkin models.*

Proof: We adapt the proofs given in [BK98a] and [Koh94b] which in turn are based on the ideas of [And71].

Let Γ_Σ be the set of Σ-sentences which cannot be refuted by the calculus $\mathcal{ER}_{fc}$ ($\Gamma_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) | \Phi_{cl} \not\vdash_{\mathcal{ER}_{fc}} \Box\}$), then we show that Γ_Σ is a saturated abstract consistency class for Henkin models 3.6 which entails Henkin completeness for $\mathcal{ER}_{fc}$ by theorem 3.29.

In particular we have to verify that Γ_Σ ensures the abstract consistency properties ∇_c, $\nabla_\neg$, ∇_β, $\nabla_\lor$, $\nabla_\land$, $\nabla_\forall$, $\nabla_\exists$, $\nabla_\mathfrak{b}$, $\nabla_\mathfrak{q}$. Furthermore we have to show that Γ_Σ is saturated.

∇_c Suppose that $\mathbf{A}, \neg\mathbf{A} \in \Phi$, where $\mathbf{A} \in \mathit{cwff}_o(\Sigma)$. Since $\mathbf{A}$ is atomic we have $\Phi_{cl} * [\mathbf{A}]^T * [\neg\mathbf{A}]^T \vdash_{Cnf} \Phi_{cl} * [\mathbf{A}]^T * [\mathbf{A}]^F$ and hence we can derive □ with *Res* and *Triv*. This contradicts our assumption.

In all of the remaining cases, we show the contrapositives, e.g., in the next case we prove, that for all $\Phi \in \Gamma_\Sigma$, if $\Phi * \neg\neg\mathbf{A} * \mathbf{A} \notin \Gamma_\Sigma$, then $\Phi * \neg\neg\mathbf{A} \notin \Gamma_\Sigma$, which entails the assertion.

$\nabla_\neg$ Let us assume that $\Phi_{cl} * [\neg\neg\mathbf{A}]^T * [\mathbf{A}]^T \vdash_{\mathcal{ER}_{fc}} \Box$. We immediately get the assertion since $[\neg\neg\mathbf{A}]^T \vdash_{\mathcal{CNF}} [\mathbf{A}]^T$.

$\nabla_\mathfrak{f}$ If $\Phi_{cl} * [\mathbf{A}]^T * [\mathbf{A}_{\downarrow_{\beta\eta}}]^T \vdash_{\mathcal{ER}_{fc}} \Box$, then we get that $\Phi_{cl} * [\mathbf{A}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ by lemma 4.13 (note that $\mathbf{A}$ is assumed to be in head-normal form).

$\nabla_\lor$ If $\Phi_{cl} * [\mathbf{A} \lor \mathbf{B}]^T * [\mathbf{A}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ and $\Phi_{cl} * [\mathbf{A} \lor \mathbf{B}] * [\mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$, then $\Phi_{cl} * [\mathbf{A} \lor \mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ by lemma 4.15(1).

$\nabla_\land$ Analogous to $\nabla_\neg$ as $[\neg(\mathbf{A} \lor \mathbf{B})]^T \vdash_{\mathcal{CNF}} [\neg\mathbf{A}]^T$ and $[\neg(\mathbf{A} \lor \mathbf{B})]^T \vdash_{\mathcal{CNF}} [\neg\mathbf{B}]^T$.

$\nabla_\forall$ Let $\Phi_{cl} * [\Pi^\alpha\ \mathbf{F}]^T * [\mathbf{F}\ \mathbf{A}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ for each closed formula $\mathbf{A}$. By lifting lemma 4.12 we get that $\Phi_{cl} * [\Pi^\alpha\ \mathbf{F}]^T * [\mathbf{F}\ X]^T \vdash_{\mathcal{ER}_{fc}} \Box$ for a new variable X_α and thus obviously $\Phi_{cl} * [\Pi^\alpha\ \mathbf{F}]^T \vdash_{\mathcal{ER}_{fc}} \Box$.

$\nabla_\exists$ Let us assume that $\Phi_{cl} * [\neg(\Pi\ \mathbf{F})]^T * [\neg(\mathbf{F}\ w)]^T \vdash_{\mathcal{ER}_{fc}} \Box$. Note that $\neg(\Pi\ \mathbf{F})$ is a closed formula and furthermore that w does not occur in $\Phi_{cl} * [\neg(\Pi\ \mathbf{F})]^T$. We get the assertion as $[\neg(\Pi\ \mathbf{F})]^T \vdash_{\mathcal{CNF}} [\neg(\mathbf{F}\ w')]^T$ for a Skolem constant w', which is just a renaming of w above.

$\nabla_\mathfrak{b}$ We show that if $\Phi_{cl} * [\neg(\mathbf{A} \doteq^o \mathbf{B})]^T * [\neg\mathbf{A}]^T * [\mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$ and $\Phi_{cl} * [\neg(\mathbf{A} \doteq^o \mathbf{B})]^T * [\mathbf{A}]^T * [\neg\mathbf{B}]^T \vdash_{\mathcal{ER}_{fc}} \Box$, then $\Phi_{cl} * [\neg(\mathbf{A} \doteq \mathbf{B})]^T \vdash_{\mathcal{ER}_{fc}} \Box$. Note that $\Phi_{cl} * [\neg(\mathbf{A} \doteq \mathbf{B})]^T \equiv \Phi_{cl} * [\neg\Pi(\lambda P_{o\to o}.\ \neg P\ \mathbf{A} \lor P\ \mathbf{B})]^T \vdash_{\mathcal{CNF}} \Phi_{cl} * [r\ \mathbf{A}]^T * [r\ \mathbf{B}]^F$, where $r_{o\to o}$ is a new Skolem constant. Now consider the following derivation

$$\cfrac{\cfrac{\cfrac{[r\ \mathbf{A}]^T \quad [r\ \mathbf{B}]^F}{[r\ \mathbf{A} = r\ \mathbf{B}]^F}\ \mathit{Res}}{[\mathbf{A} = \mathbf{B}]^F}\ \mathit{Dec}, \mathit{Triv}}{[\mathbf{A} \Leftrightarrow \mathbf{B}]^F}\ \mathit{Equiv}$$

Hence $\Phi_{cl} * [\neg(\mathbf{A} = \mathbf{B})]^T \vdash_{\mathcal{ER}_{fc}} \Phi_{cl} * [\neg(\mathbf{A} = \mathbf{B})]^T * [\mathbf{A} \Leftrightarrow \mathbf{B})]^F$ and we get the conclusion as a consequence of lemma 4.15(2).

$\nabla_{\mathfrak{q}}$ We show that if $\Phi_{cl} * [\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})]^T * [\neg(\mathbf{F}\ w \doteq^{\beta} \mathbf{G}\ w)]^T \vdash_{\mathcal{ER}_{fc}} \Box$, then $\Phi_{cl} * [\neg(\mathbf{F} \doteq \mathbf{G}))]^T \vdash_{\mathcal{ER}_{fc}} \Box$. Note that $\Phi_{cl} * [\neg(\mathbf{F} \doteq \mathbf{G})]^T * [\neg(\mathbf{F}\ w \doteq \mathbf{G}\ w)]^T \equiv \Phi_{cl} * [\neg\Pi(\lambda_{(\alpha\to\beta)\to o}.\ \neg(Q\ \mathbf{F}) \lor (Q\ \mathbf{G}))]^T * [\neg\Pi(\lambda P_{\beta\to o}.\ \neg(P\ (\mathbf{F}\ w)) \lor (P\ (\mathbf{G}\ w)))]^T \vdash_{\mathcal{CNF}} \Phi_{cl} * [q\ \mathbf{F}]^T * [q\ \mathbf{G}]^F * [p\ (\mathbf{F}\ w)]^T * [p\ (\mathbf{G}\ w)]^F$ and that $\Phi_{cl} * [\neg(\mathbf{F} \doteq \mathbf{G})] \vdash_{\mathcal{CNF}} \Phi_{cl} * [r\ \mathbf{F}]^T * [r\ \mathbf{G}]^F$, where $p_{\beta\to o}, q_{(\alpha\to\beta)\to o}$ and $r_{(\alpha\to\beta)\to o}$ are new Skolem constants. Now consider the following derivation:

$$\frac{\dfrac{\dfrac{\dfrac{[r\ \mathbf{F}]^T \quad [r\ \mathbf{G}]^F}{[r\ \mathbf{F} = r\ \mathbf{G}]^F}\ Res}{[\mathbf{F} = \mathbf{G}]^F}\ Dec, Triv}{[\mathbf{F}\ s = \mathbf{G}\ s]^F}\ Func}{\begin{array}{c}[t\ (\mathbf{F}\ s)]^T \\ [t\ (\mathbf{G}\ s)]^F\end{array}}\ Leib$$

Here again s_α and $t_{\beta\to o}$ are new Skolem constants. Hence $\Phi_{cl} * [r\ \mathbf{F}]^T * [r\ \mathbf{G}]^F \vdash_{\mathcal{ER}_{fc}} \Phi_{cl} * [r\ \mathbf{F}]^T * [r\ \mathbf{G}]^F * [t\ (\mathbf{F}\ s)]^T * [t\ (\mathbf{G}\ s)]^F$. Now the conclusion follows from the assumption as s, t and r are only renamings of the Skolem symbols w, p and q and as all do not occur in Φ_{cl}.

To see that Γ_Σ is saturated let $\mathbf{A} \in cwff_o(\Sigma)$ and $\Phi \subseteq cwff_o(\Sigma)$ with $\Phi_{cl} \not\vdash_{\mathcal{ER}_{fc}} \Box$. We have to show that $\Phi_{cl} * \mathbf{A} \not\vdash_{\mathcal{ER}_{fc}} \Box$ or $\Phi_{cl} * \neg\mathbf{A} \not\vdash_{\mathcal{ER}_{fc}} \Box$. For that suppose $\Phi_{cl} \not\vdash_{\mathcal{ER}_{fc}} \Box$, but $\Phi_{cl} * \mathbf{A} \vdash_{\mathcal{ER}_{fc}} \Box$ and $\Phi_{cl} * \neg\mathbf{A} \vdash_{\mathcal{ER}_{fc}} \Box$. By lemma 4.15(1) we get that $\Phi_{cl} * \mathbf{A} \lor \neg\mathbf{A} \vdash_{\mathcal{ER}_{fc}} \Box$, and hence, since $\mathbf{A} \lor \neg\mathbf{A}$ is a tautology, it must be the case that $\Phi_{cl} \vdash_{\mathcal{ER}_{fc}} \Box$, which contradicts our assumption.

□

Remark 4.17 ***(Eager Unification)****.* In contrast to Huet [Hue72, Hue73a] eager unification is essential within our approach. This is illustrated by the argumentations for $\nabla_{\mathfrak{b}}$ and $\nabla_{\mathfrak{q}}$ in the completeness proof 4.16 as well as many of the examples presented in Chapter 8.

Conjecture 4.18 (Restricted rule *Leib***).** *Even though rule Leib is employed to terms of arbitrary types in the completeness proof (see 4.16($\nabla_{\mathfrak{q}}$)) we conjecture, that this rule can be restricted to unification constraints between terms of primitive type:*

$$\frac{\mathbf{C} \lor [\mathbf{M}_\alpha {=} \mathbf{N}_\alpha]^F \quad \alpha \in \{o, \iota\}}{\mathbf{C} \lor [\forall P_{\alpha\to o}.\ P\ M \Rightarrow P\ N]^F}\ Leib'$$

Unfortunately a formal proof for this conjecture, whose validity would be of practical importance, is still missing. In order to prove this conjecture, it would suffice to prove that if $\Phi_{cl} * [\mathbf{A} \doteq^{\alpha\to\beta} \mathbf{B})]^F \vdash_{\mathcal{ER}_{fc}} \Box$ then $\Phi_{cl} * [\mathbf{A}\ s \doteq^{\beta} \mathbf{B}\ s]^F \vdash_{\mathcal{ER}_{fc}} \Box$ for any new Skolem term s_α. Note that $[\forall X_\alpha.\ \mathbf{A}\ X \doteq^{\beta} \mathbf{B}\ X] \vdash_{\mathcal{CNF}} [\mathbf{A}\ s \doteq^{\beta} \mathbf{B}\ s]^F$. Thus, this lemma expresses one direction of the functional extensionality property formulated with Leibniz equality and with this lemma we could reduce the applications of rule Leib in all three calculi ($\mathcal{ER}$, $\mathcal{EP}$ and $\mathcal{ERUE}$) discussed in this thesis and employ the restricted rule Leib' instead. This result would be of practical importance as it allows us to restrict the search space. In the extensional RUE-Resolution approach rule Leib may even become redundant (c.f. remark 8.1).

4.5 Theorem Equivalence

With respect to our theoretical goal of proving Henkin completeness it does not make a difference whether exhaustive clause normalisation derivations are grouped together into one rule *Cnf* like in calculi $\mathcal{ER}_f$ and $\mathcal{ER}$, or if the single clause normalisation rules are lifted as single inference rules to calculus level like in $\mathcal{ER}_{fc}$. But note that there is a practical motivation for the ungrouping of $\mathcal{CNF}$-derivation as well: Calculus $\mathcal{ER}_{fc}$ allows to avoid redundant applications of identical unification derivations to all proper clauses belonging to the same abstract clause. For instance, it may be more appropriate first to apply unification rules to a non-proper clause and then to apply clause normalisation rule to the result than the other way around.

Remark 4.19 ***(Convention).***

1. Whereas the $\mathcal{CNF}$ and the $\mathcal{UNI}$ rules aside from *Subst* do not directly influence each others applicability, they may indirectly influence each others applicability via Skolemisation. More precisely if r_2 is rule Π^T and r_1 is rule *Func* the arity of the Skolem term that is introduced in *Func* increases its arity when switching r_1 and r_2.

2. Another important and simplifying convention in all proof transformations in this thesis is to consider proper proof trees instead of proof graphs, such that each derived clause in a derivation is used exactly once as a premise clause in the subsequent derivation steps.

Lemma 4.20 (Derivability of Proper Clauses). *For each proper clause $\mathcal{C}$ and clause set Φ, such that $\Delta_1 : \Phi \vdash_{\mathcal{ER}_{fc}} \mathcal{C}$, there is a $\mathcal{ER}_f$-derivation $\Delta_2 : \Phi \vdash_{\mathcal{ER}_f} \mathcal{C}$.*

Proof: The proof idea is to show that the single, distributed clause normalisation steps can be grouped in exhaustive $\mathcal{CNF}$-chains, which can then be replaced by rule *Cnf*. The proof is by induction[5] on the length n of derivation Δ_1 and the base case ($n \equiv 0$) is trivial. In the induction step ($n > 0$) we consider the first step in derivation $\Delta_1 : \Phi \vdash^{r_1} \Phi * \mathcal{D}_1 \vdash_{\mathcal{ER}_{fc}} \mathcal{C}$, where $r_1 \in \mathcal{ER}_{fc}$. We proceed by examining all possibilities for r_1:

$r_1 \in \{Res, Fac, Prim\}$ As these rules operate on proper clauses only, we have on the one hand that r_1 is also a proper $\mathcal{ER}_f$ derivation step. On the other hand we know by induction hypothesis that there is a proper $\mathcal{ER}_f$ derivation $\Phi * \mathcal{D}_1 \vdash_{\mathcal{ER}_f} \mathcal{C}$. Thus, $\Phi \vdash_{\mathcal{ER}_f} \mathcal{C}$.

$r_1 \in \mathcal{UNI}_f \cup Leib, Equiv$ Analogous to above the first step is also a proper $\mathcal{ER}_f$ derivation step, such that the assertion follows immediately by induction hypotheses.

$r_1 \in \mathcal{CNF}$ In this case we look for the first step in derivation Δ_1 that employs a non-$\mathcal{CNF}$-rule. Without loss of generality let us assume that this happens in step n for $n > 1$. Then Δ_1 has form $\Delta_1 : \Phi \vdash^{r_1} \Phi * \mathcal{D}_1 \vdash^{r_2} \ldots \vdash^{r_{n-1}} \Phi * \mathcal{D}_1 * \ldots * \mathcal{D}_{n-1} \vdash^{r_n} \Phi * \mathcal{D}_1 * \ldots * \mathcal{D}_n \vdash_{\mathcal{ER}_{fc}} \mathcal{C}$, such that $r_j \in \mathcal{CNF}$ for $1 < j < n$. Without loss of generality let us assume that each of the steps r_j employs the newly generated clause from the previous step as premise clause (we can reorder the derivation steps in Δ_1 without

[5] Note that with convention 4.19, which says that we consider proof trees rather than proof graphs, this induction proof (and many others in this thesis) should actually be carried out by induction on the structure of the proof tree or by induction on the depth of the proof tree. But note that the derivations described by proof trees can be linearised in a unique way, such that the length of the linearised proof trees gives us an well-founded ordering as well.

any influence to the particularly derived clauses such that this assumption is met). In case $r_n \in \{Res, Fac, Prim\}$, $\mathcal{D}_n$ must be a proper clause, such that we can obviously replace the initial $n-1$ derivation steps in Δ_1 by a single application of rule *Cnf* in calculus $\mathcal{ER}_f$. Now we again employ the convention that we consider proper proof trees instead of proof graphs in our formal proofs such that each derived clause in a derivation is used exactly once as a premise clause in one of the following derivation steps. Consequently $\Delta_1 : \Phi \vdash_{\mathcal{CNF}} \Phi * \mathcal{D}_n \vdash_{\mathcal{ER}_{fc}} \mathcal{C}$ and the assertion follows by induction hypotheses.
In case $r_n \in \mathcal{UNI}_f$, we verify that the clause normalisation steps r_j for $1 \leq j < n$ do not affect the unification constraints of the involved clauses. Thus we can obviously apply rule r_n in the first place in Δ_1 as well. Furthermore, the clause normalisation steps r_j for $1 \leq j < n$ are applicable to the result of this new first step in Δ_1 and the result of this derivation chain in the n-th step is clause $\mathcal{D}_n$. Now the assertion follows again by induction hypotheses.

□

As □ is also a proper clause we immediately get the following corollary.

Corollary 4.21 (Theorem Equivalence of $\mathcal{ER}_{fc}$ and $\mathcal{ER}_f$). *The calculi $\mathcal{ER}_{fc}$ and $\mathcal{ER}_f$ are theorem equivalent, i.e., $\Phi \vdash_{\mathcal{ER}_{fc}} \Box$, iff $\Phi \vdash_{\mathcal{ER}_f} \Box$.*

It is no longer possible to delay all (pre-)unification rules, but our claim is that the additional rule *FlexFlex* can still be delayed until the end of a refutation. This result would give us that rule *FlexFlex* is not needed at all, as □ is defined modulo *flex-flex* constraints.

Conjecture 4.22 (Theorem Equivalence of $\mathcal{ER}$ and $\mathcal{ER}_{fc}$ (or $\mathcal{ER}_f$)).
The calculi $\mathcal{ER}$ and $\mathcal{ER}_{fc}$ (or $\mathcal{ER}_f$) are theorem equivalent.

There is evidence given for its validity. None of the challenging examples discussed in this thesis or in any of the papers [BK98a, BK98b, Ben97, BK97b] needs rule *FlexFlex*. Furthermore, our case study with the extensional higher-order resolution prover LEO [BK98b] showed that none of the examples from the MIZAR-articles *Boolean and Basic Properties of Sets* [TS89, Byl89], which are very interesting with respect to extensionality principles, require the application of the *FlexFlex* rule. Additional evidence for this claim is given by the direct proof carried out in [BK98a].

Chapter 5

Extensional Higher-Order Paramodulation: $\mathcal{EP}$

In this chapter we shall try to adapt traditional first-order paramodulation to higher-order logic. The paramodulation rule is not sufficient to ensure Henkin completeness as we run into problems with the Boolean and functional extensionality principles of primitive equality, hence additional extensionality rules are needed.

5.1 A Naive and Incomplete Adaptation of Paramodulation

Figure 5.1 shows the traditional paramodulation rule and introduces a more elegant higher-order reformulation.[1] Note that we assume the symmetric case for both rules.

The paramodulation rule *Para* replaces any subterm $\mathbf{T}_\beta$ (which contains no variable that is bound outside $\mathbf{T}$) by $\mathbf{R}_\beta$ provided that $\mathbf{T}$ and $\mathbf{L}$ are unifiable. Analogously to higher-order resolution and factorisation, unification has to be delayed and thus a unification constraint $[\mathbf{T} = \mathbf{L}]^F$ is added to the resulting clause. Unfortunately, we thereby introduce with each application of rule *Para* twice as many (because of symmetric application of equations) new clauses into the search space as there are subterms $\mathbf{T}_\alpha$ of type α in the given term $\mathbf{A}$. To illustrate applications of rule *Para* we consider the following two unit clauses $\mathcal{C}_1 : [p\ (f\ (f\ a))]^T$ and $\mathcal{C}_2 : [f = h]^T$, where $p_{\iota\to o}$, $f_{\iota\to\iota}$, $h_{\iota\to\iota}$ and a_ι are constants. By application of paramodulation in left to right direction we obtain the follwing two clauses: $\mathcal{C}_3 : [p\ (h\ (f\ a))]^T \vee [f = f]^F$ and $\mathcal{C}_4 : [p\ (f\ (h\ a))]^T \vee [f = f]^F$. In our trivial example the generated unification constraints can be immedietaly eliminated in eager unification attempt with rule *Triv*.

Rule *Para'* combines the expressivness of our higher-order language with the power of higher-order (pre-)unification and avoids the introduction of too many clauses into the search space. Instead only one new clause with a new flexible literal head is created. The idea of this rule is very simple: it describes the respective resolution step between the left premise clause and the clause that corresponds to the right premise clause when replacing the positive primitive equation by a (normalised) positive Leibniz equation (the modified right clause would have form $[P\ \mathbf{L}]^F \vee [P\ \mathbf{R}]^T \vee D$ — or with an additional primitive substitution step $[P\ \mathbf{L}]^T \vee [P\ \mathbf{R}]^F \vee D$).

When applying rule *Para'* to the clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ from left to right, we introduce only

[1] This rule was suggested by Michael Kohlhase.

$\boxed{\text{Paramodulation:}}$ (defined for proper clauses only)

$$\frac{[\mathbf{A}[\mathbf{T}_\beta]]^\alpha \vee C \quad [\mathbf{L} =^\beta \mathbf{R}]^T \vee D}{[\mathbf{A}[\mathbf{R}]]^\alpha \vee C \vee D \vee [\mathbf{T} =^\beta \mathbf{L}]^F}\ Para$$

$$\frac{[\mathbf{A}]^\alpha \vee C \quad [\mathbf{L} =^\beta \mathbf{R}]^T \vee D}{[P_{\alpha\to o}\ \mathbf{R}]^\alpha \vee C \vee D \vee [\mathbf{A} =^o P_{\beta\to o}\ \mathbf{L}]^F}\ Para'$$

In *Para* subterm **T** of **A** must not contain free variables that are bound outside

Figure 5.1: The Paramodulation rules *Para* and *Para'*

one new clause: $\mathcal{C}_5 : [P\ h]^T \vee [p\ (f\ (f\ a)) = P\ f]^F$. By eager pre-unification we can generate the following four instantiations for P and propagate this partial solutions with rule *Subst* to the literal $[P\ h]^T$:

1) $P \leftarrow \lambda Z_{\iota\to\iota}.\ p\ (f\ (f\ a))$ 2) $P \leftarrow \lambda Z_{\iota\to\iota}.\ p\ (Z\ (f\ a))$
3) $P \leftarrow \lambda Z_{\iota\to\iota}.\ p\ (f\ (Z\ a))$ 4) $P \leftarrow \lambda Z_{\iota\to\iota}.\ p\ (Z\ (Z\ a))$

The pure imitation solution (1) introduces clause $\mathcal{C}_1$ again and is thus redundant.[2] By instantiating solutions (2) and (3) we obtain exactly the clauses $\mathcal{C}_3$ and $\mathcal{C}_4$ (clearly without the trivial pairs), which are identical to the result of the applications of the traditional paramodulation rule. Solution (4) is the most interesting one as it encodes the simultaneous application of $[f = h]^T$ to both subterms f in $[p\ (f\ (f\ a))]^T$. As we have illustrated, the only clause resulting from the application of rule *Para'* encodes all the possible traditional paramodulation steps together with all simultaneous applications and even the original clause itself. The only problem for practical applications is, that we introduce a new flexible literal head with each application of *Para'*, such that the primitive substitution rule becomes applicable. As this seems to be rather useless, a good heuristic in practice might be to avoid primitive substitution steps on flexible heads generated by rule *Para'*.

Remark 5.1 ***(Reflexivity Resolution)***. Note that a reflexivity resolution rule is already naturally integrated into calculus $\mathcal{ER}$, as it is derivable with the help of the unification rules. The unification rules already operate on negative primitive equational literals, i.e., unification constraints, like $[f\ X = Q\ a]^F$, such that each negative equation between two unifiable terms can be refuted by them.

We want to point out, that the encoding of unification constraints as negated literals is essential here, whereas it is not of importance for calculus $\mathcal{ER}$. If we would not encode unfication constraints as negative primitive equational literals and would differentiate between both concepts, then we obviously have to face the reflexivity resolution problem.

Remark 5.2 ***(Paramodulation into Unification Constraints)***. It turns out that paramodulation into unification constraints is not neccessary, and in fact we can show its deriv-

[2] On the other hand this possibly leads to interesting heuristics in practice: When applying rule *Para'* with a right premise clause which is a unit equation, we can remove the left premise clause from the search space as this clause gets encoded into the result of the paramodulation step itself.

ability:

$$\frac{\mathcal{C}_1 : C \vee [\mathbf{A}[\mathbf{T}_\alpha] \; = \mathbf{B}]^F \quad \mathcal{C}_2 : [\mathbf{L} =^\alpha \mathbf{R}]^T \vee D}{\mathcal{C}_3 : C \vee D \vee [\mathbf{A}[\mathbf{R}_\alpha] \; = \mathbf{B}]^F \vee [\mathbf{T} =^\alpha \mathbf{L}]^F} \; Para$$

Each such step can be replaced by the following derivation (p is a Skolem constant):

$$\begin{array}{ll}
Leib(\mathcal{C}_1), \mathcal{CNF}: & \mathcal{D}_1 : C \vee [p \; \mathbf{A}[\mathbf{T}_\alpha]]^T \\
 & \mathcal{D}_2 : C \vee [p \; \mathbf{B}]^F \\
Para(\mathcal{D}_1, \mathcal{C}_2): & \mathcal{D}_3 : C \vee D \vee [p \; \mathbf{A}[\mathbf{R}_\alpha]]^T \vee [\mathbf{T} =^\alpha \mathbf{L}]^F \\
Res(\mathcal{D}_3, \mathcal{D}_2), Fac, Triv: & \mathcal{D}_4 : C \vee D \vee [(p \; \mathbf{A}[\mathbf{R}_\alpha]) = (p \; \mathbf{B})]^F \vee [\mathbf{T} =^\alpha \mathbf{L}]^F \\
Dec(\mathcal{D}_4), Triv: & \mathcal{D}_5 : C \vee D \vee [\mathbf{A}[\mathbf{R}_\alpha] = \mathbf{B}]^F \vee [\mathbf{T} =^\alpha \mathbf{L}]^F
\end{array}$$

On the one hand paramodulation into unification constraints may shorten proofs and in some examples this seems to be very appropriate, but on the other hand such an approach may be hard to guide in practice.

Remark 5.3 ***(Functional Extensionality and rule* Para'*)***. It has been claimed by an unknown referee of [Ben98], that rule *Para'* already captures full extensionality, such that rule *Para'* should be preferred over *Para*. Example $\mathbf{E}_1^{func}$ $((\forall X_\iota . \; f_{\iota\to\iota} X = g_{\iota\to\iota} \; X) \Rightarrow (p_{(\iota\to\iota)\to o} \; f \Rightarrow p_{(\iota\to\iota)\to o} \; g))$ discussed in Section 8.6 demonstrates that this is not true: even rule *Para'* requires additional extensionality rules to ensure full functional extensionality.

Definition 5.4 **($\mathcal{EP}_{naive}$).** The calculus $\mathcal{EP}_{naive}$ consists of the rules of calculus $\mathcal{ER}$ (see Figure 4.2) enriched by the paramodulation rule *Para* (see Figure 5.1). We assume that the result of each rules application is transformed into head-normal form. A set of formulae Φ is refutable in calculus $\mathcal{EP}_{naive}$, iff there is a derivation $\Delta : \Phi_{cl} \vdash_{\mathcal{EP}_{naive}} \Box$, where $\Phi_{cl} := \{[\mathbf{F}{\downarrow_h}]^T | \mathbf{F} \in \Phi\}$ is the set of clauses obtained from Φ by simple pre-clausification. We want to point out that primitive equations are not expanded by Leibniz definition.

Next, we discuss soundness of the extended calculus $\mathcal{EP}_{naive}$ and show by a counterexample that $\mathcal{EP}_{naive}$ is not complete with respect to Henkin semantics.

Theorem 5.5 (Soundness of naive higher-order paramodulation).
The calculus $\mathcal{EP}_{naive}$ is sound with respect to Henkin semantics.

Proof: Soundness of the traditional paramodulation rule *Para* is obvious (note that we avoid the replacement of subterms T with free variables that are bound outside): given a standard model M for the two premise clauses, then one can easily see that the paramodulant is also valid in M, as either the unification constraint evaluates to F and we are done or it evaluates to T giving rise to the validity of literal $[\mathbf{A}[\mathbf{R}]]]^\alpha$, in case $[\mathbf{A}[\mathbf{T}]]]^\alpha$ guaranteed the validity of the first premise clause and $[\mathbf{L} = \mathbf{R}]^T$ guaranteed the validity of the second premise clause (all other cases are trivial).

The proof of soundness for the new rule *Para'* is analogous, even though this rules looks more complicated: we consider all possible variable assignments φ which map variable $P_{\alpha\to o}$ to a function in the domain $D_{\alpha\to o}$ and employ an analogous argumentation like above. □

Theorem 5.6 (Incompleteness of $\mathcal{EP}_{naive}$). *The calculus $\mathcal{EP}_{naive}$ is incomplete with respect to Henkin semantics.*

Proof: The assertion is proven by the following counterexamples to the assumption of Henkin completeness of calculus $\mathcal{EP}_{naive}$:

Example 5.7 ***(Incompleteness of*** $\mathcal{EP}_{naive}$***).***

$\mathbf{E}_1^{Para}$ $\neg\exists X_o.\ (X = \neg X)$
This formula expresses, that the negation operator is fix-point free, which is obviously the case in Henkin semantics. Our calculus is not able to find a proof as clause normalisation of the negated assertion leads to the single clause

$$\mathcal{C}_1 :\ [a = \neg a]^T$$

where a_o is a new Skolem constant. The only rule that is applicable is self-paramodulation on positions $\langle 1\rangle$, $\langle 2\rangle$ and $\langle\rangle$ leading to the following clauses:

$$\begin{array}{lll}
Para(\mathcal{C}_1,\mathcal{C}_1)\ at\ \langle 1\rangle : & \mathcal{C}_2 :\ [a = \neg a]^T \vee [\neg a = a]^F & \mathcal{C}_3 :\ [\neg a = \neg a]^T \vee [a = a]^F \\
Para(\mathcal{C}_1,\mathcal{C}_1)\ at\ \langle 2\rangle : & \mathcal{C}_4 :\ [a = \neg a]^T \vee [a = \neg a]^F & \mathcal{C}_5 :\ [a = a]^T \vee [\neg a = a]^F \\
Para(\mathcal{C}_1,\mathcal{C}_1)\ at\ \langle\rangle : & \mathcal{C}_6 :\ [a]^T \vee [\neg a = (a = \neg a)]^F & \mathcal{C}_7 :\ [a]^F \vee [a = (a = \neg a)]^F
\end{array}$$

Case distinction on the possible denotations $\{\mathtt{T},\mathtt{F}\}$ for a shows that all these clauses are tautologies. Thus no refutation is possible in $\mathcal{EP}_{naive}$.

$\mathbf{E}_2^{Para}$ $\neg\exists G_{\iota\to\iota\to o}.\ \forall P_{\iota\to o}.\ \exists X_\iota.\ G\ X =^{\iota\to o} P$
This is a simple formulation of cantor's theorem stating that there exists no surjective function from the set of individuals into the set of sets of individuals. Clause normalisation results in

$$\mathcal{C}_1 : [G\ X =^{\iota\to o} p]^T$$

where $p_{\iota\to o}$ is a Skolem constant. Similar to the case above, a refutation in $\mathcal{EP}_{naive}$ is not possible.

$\mathbf{E}_3^{Para}$ $\exists M_{o\to o}.\ M \neq \emptyset$, where $\emptyset_{o\to o}$ is defined as $\lambda X_o.\ \mathbf{F}_o$. This formula expresses, that there exists a set of truth values that is not empty. As the set of truth values is fixed in Henkin semantics and has exactly two elements, this assertion is obviously valid. Clause normalisation results in

$$\mathcal{C}_1 :\ [M = \lambda X_o.\ \bot]^T$$

where $M_{o\to o}$ is a free variable. Analogous to $\mathbf{E}_1^{Para}$ no refutation is possible.

$\mathbf{E}_4^{Para}$ $\neg\exists M_{o\to o}.\ M = \overline{M}$, where the set complement operator $\overline{}_{(o\to o)\to(o\to o)}$ is defined by $\lambda S_{o\to o}.\ \lambda X_o.\ \neg(X \in S)$ and $\in_{o\to(o\to o)\to o}$ as function (predicate) application $\lambda X_o.\ \lambda S_{o\to o}.\ S\ X$. This formula expresses, that there is no set M of truth values in Henkin semantics, such that M and its complementary set $\overline{M}$ are identical. Clause normalisation leads to

$$\mathcal{C}_1 :\ [m = \lambda X_o.\ \neg(m\ X)]^T$$

where $m_{o\to o}$ is a fresh Skolem constant for M. In contrast to $\mathbf{E}_3^{Para}$ there is no free variable. Again no refutation is possible.

$\mathbf{E}_5^{Para}$ If $\forall P_{(\iota\to\iota\to o)\to(\iota\to\iota\to\iota\to o)}.\ (P\ q_{\iota\to\iota\to o} = P\ r_{\iota\to\iota\to o})$, then $(\forall X_\iota.\ q\ X \neq \lambda Z_\iota.\ \neg(r\ X\ Z))$. The first equation expresses, that all relations of type $\iota\to\iota\to\iota\to o$ that can be defined based upon relation q or r, are equal (which in fact means that q and r must be equal). The second inequation expresses that q is the complement set of r, but uses an artificially complicated form (we employed the functional extensionality property once to $q \neq \lambda V_i.\ \lambda Z_i.\ \neg(r\ V\ Z))$. This assertion is again valid, as in Henkin semantics the set of truth values contains exactly two elements. Our problem normalises to

$$\mathcal{C}_1 : [P\ q =^{\iota\rightarrow\iota\rightarrow\iota\rightarrow o} P\ r]^T \qquad \mathcal{C}_2 : [p\ X =^{\iota\rightarrow o} \lambda Z_\iota.\ \neg(r\ X\ Z)]^T$$

where $P_{\iota\rightarrow\iota\rightarrow o)\rightarrow(\iota\rightarrow\iota\rightarrow\iota\rightarrow o)}, X_\iota$ are free variables and $q_{\iota\rightarrow\iota\rightarrow o}, r_{\iota\rightarrow\iota\rightarrow o}$ are function constants. Note that apart from self-paramodulation no rule is applicable. Especially there is no paramodulation step possible between the clauses $\mathcal{C}_1$ and $\mathcal{C}_2$, as there is no subterm of type $\iota \rightarrow o$ in $\mathcal{C}_1$ and no subterm of type $\iota \rightarrow \iota \rightarrow \iota \rightarrow o$ in $\mathcal{C}_2$, such that a type conform term rewriting becomes possible.

□

The overall problem in the first four examples is, that our calculus provides no mechanism to detect positive equations with an implicitly embedded contradiction. For example the clause $[a_o = \neg a]^T$ is implicitly contradictory with respect to the Boolean extensionality principle. Examples $\mathbf{E}_2^{Para}$ and $\mathbf{E}_4^{Para}$ show, that also functional extensionality is involved as the implicit contradiction follows here only with respect to both extensionality principles. The general problem is that in higher-order logic (when considering Henkin semantics as well as some weaker notions like discussed in Chapter 2.1) infinitely many semantical domains contain a fix-point free function, such as the negation operator or the set complement operator on a non-empty domain (such as the set of truth values or the domain of all functions from truth values to truth values, etc.).

In example $\mathbf{E}_5^{Para}$ none of the positive equational literals is unsatisfiable taken alone, but it is contradictory in connection with the other ones. Here again a refutation is not possible in calculus $\mathcal{EP}_{naive}$, as we would have to employ the functional extensionality principles to the positive equation in order to obtain a corresponding equation of appropriate type, such that the paramodulation rule becomes applicable.

*Remark 5.8 (**Fix-point free functions**).* Given a predicate type $\alpha := \alpha_1 \rightarrow \ldots \rightarrow \alpha_n \rightarrow o$ $(n \geq 0)$. The semantical domain domain$_{\alpha\rightarrow\alpha}$ contains a fix-point free function provided that it is not empty (which must be the case in Henkin Semantics, as at least the identity function, i.e., the evaluation of $\lambda P_\alpha.\ P_\alpha$, must be an element of this domain). Note, that these fix-point free functions map predicates to predicates. An example in domain$_{o\rightarrow o}$ is the negation operator. And in domain$_{((\iota\rightarrow\iota)\rightarrow o)\rightarrow((\iota\rightarrow\iota)\rightarrow o)}$ one can choose the set complement operator defined with the help of the negation operator: $\lambda S_{(\iota\rightarrow\iota)\rightarrow o}.\ \lambda F_{\iota\rightarrow\iota}.\ \neg(S\ F)$. It is easy to see, that analogously to the latter example one can easily construct a fix-point free function in any of the domains domain$_{\alpha\rightarrow\alpha}$.

These fix-point free functions cause the problems with positive equations in higher-order logic: Let term $\mathbf{T}_{\alpha\rightarrow\alpha}$ be an λ-expression that denotes such a fix-point free function in domain $D_{\alpha\rightarrow\alpha}$ (it has been illustrated above how to construct such λ-expressions). Then the positive equation $\mathbf{T}\ X_\alpha = X_\alpha$ is obviously unsatisfiable. Note that such single contradictory equations do not occur in first-order logic.

5.2 Positive Extensionality Rules

Before going further into the investigation of the extensionality problem with positive equational literals, let us first reconsider the analogous problem for negative equations. For example, $[a_o = \neg\neg a]^F$ is a negative equation literal, which is implicitly contradictory with respect

$$\frac{\mathbf{C} \vee [\mathbf{M}_o = \mathbf{N}_o]^T}{C \vee [\mathbf{M}_o \Leftrightarrow \mathbf{N}_o]^T}\ Equiv'$$

$$\frac{C \vee [\mathbf{M}_{\alpha\to\beta} = \mathbf{N}_{\alpha\to\beta}]^T \quad X \text{ new free variable}}{C \vee [\mathbf{M}\ X = \mathbf{N}\ X]^T}\ Func'$$

Figure 5.2: The extensionality rules for positive primitive equations.

to Boolean extensionality. Just as for reflexivity resolution (cf. 5.1) our calculus already provides a solution to this problem as it interprets negative equational literals automatically as unification constraints and offers an appropriate extensionality treatment by the extensionality rules *Leib, Func* and *Equiv*:

*Remark 5.9 **(Extensionality for Negative Primitive Equations)**.*
Our calculi encode unification constraints as negative primitive literals and do not differentiate between unification problems and negative primitive equations. Therefore an additional extensionality treatment for negative primitive equations is not needed as the unification algorithm has already been extended by the special extensionality rules *Leib, Func* and *Equiv.* When differentiating between negative primitive equations and unification constraints, the unit clause $[a_o = \neg\neg a]^F$ is not refutable in the resulting approach, and we would need an additional extensionality treatment for negative primitive equations.

In contrast to this pleasent result, we do have to face the lack of extensionality principles in the case of positive equational literals. And, as we have seen in the examples above, our calculus so far does not provide a solution to it. What we need is a way to test the inequality of the two sides of a positive equation with respect to the extensionality principles and also with respect to the knowledge provided by the other clauses in the search space (see for example $\mathbf{E}_5^{Para}$). This suggests to introduce analogous extensionality rules to *Func, Equiv,* and *Leib*, that operate on positive equations. To anticipate the results of the analysis of Henkin completeness, it turns out that a positive counterpart to rule *Leib* is not needed, and thus our new extensionality for primitive positive equations are *Func'* and *Equiv'* as given in Figure 5.2.

*Remark 5.10 **(Rule** Leib')*. A positive counterpart for rule *Leib* would have the following form:

$$\frac{\mathbf{C} \vee [\mathbf{M}_\alpha = \mathbf{N}_\alpha]^T \quad \alpha \in \{o, \iota\}}{\mathbf{C} \vee [\forall P_{\alpha\to o}.\ P\ M \Rightarrow P\ N]^T}\ Leib'$$

This rule is neither needed in our motivating examples nor in the completeness proof below, and as a consequence it should be possible to directly prove hat rule *Leib'* is admissible for calculus $\mathcal{EP}$.

5.3 Basic Definitions

We further extend our extensional higher-order paramodulation approach by adding rule *FlexFlex* as in Chapter 4, and lift the single clause normalisation rules to calculus level, instead of grouping exhaustive clause normalisation derivations together in rule *Cnf*.

The definitions for clause normalisation calculus $\mathcal{CNF}$ (cf. 4.2), $\mathcal{UNI}$ and $\mathcal{UNI}_f$ (cf. 4.4) need not to be modified and the lemmata 4.3 (Soundness of $\mathcal{CNF}$), 4.5, and 4.6 (properties of higher-order unification) will be employed in this section again.

Definition 5.11 (Extensional Higher-Order Paramodulation).

$\mathcal{EP}$ The calculus $\mathcal{EP}$ consists of the rules of calculus $\mathcal{ER}$ (see Figure 4.2) enriched by the paramodulation rule *Para* in Figure 5.1 and the positive extensionality rules displayed in Figure 5.2, i.e., $\mathcal{EP} := \mathcal{ER} \cup \{Para, Equiv', Func'\}$.

$\mathcal{EP}_f$ The extension $\mathcal{EP}_f$ of calculus $\mathcal{EP}$, that employs full higher-order unification instead of higher-order pre-unification, is defined as $\mathcal{EP}_f := \mathcal{EP} \cup \{FlexFlex\}$.

$\mathcal{EP}_{fc}$ The calculus $\mathcal{EP}_{fc}$, that employs stepwise instead of exhaustive clause normalisation, is defined by $\mathcal{EP}_{fc} := (\mathcal{EP}_f \backslash \{Cnf\}) \cup \mathcal{CNF}$.

For all calculi we assume that the result of each rule application is transformed into head-normal form. A set of formulae Φ is refutable in calculus $\mathcal{EP}$, iff there is a derivation $\Delta : \Phi_{cl} \vdash_{\mathcal{EP}} \Box$, where $\Phi_{cl} := \{[\mathbf{F}{\downarrow_h}]^T | \mathbf{F} \in \Phi\}$ is the set obtained from Φ by simple pre-clausification. Unification literals are still only accessible to the unification rules.

Theorem 5.12 (Soundness of Extensional Higher-Order Paramodulation). *The calculi $\mathcal{EP}$, $\mathcal{EP}_f$ and $\mathcal{EP}_{fc}$ are sound with respect to Henkin semantics.*

Proof: We already know by 5.5, that calculus $\mathcal{EP}_{naive}$ is sound. The soundness of the new positive extensionality rules is obvious in Henkin semantics (see Lemmata 2.37 and 2.43), as they simply apply the extensionality properties, which are valid in standard semantics. □

Lemma 5.13 (Proper Derivations). *For each non-proper clause $\mathcal{D}$, proper clause $\mathcal{C}$, and clause set Φ, such that $\Phi * \mathcal{D} \vdash_{\mathcal{EP}_{fc}} \mathcal{C}$, we have $\Phi \cup \mathcal{CNF}(\mathcal{D}) \vdash_{\mathcal{EP}_{fc}} \mathcal{C}$.*

Proof: Analogous to lemma 4.10. In the induction step the case $r \equiv Para$ is analogous to the cases $r \in \{Res, Fac, Prim\}$, and the cases $r \in \{Func', Equiv'\}$ are analogous to the case $r \in \mathcal{UNI}_f$. □

5.4 Lifting Properties

The clause normalisation lifting property stated in lemma 4.11 is not affected by the modification of the calculus and will be employed in the following lifting lemma for calculus $\mathcal{EP}_{fc}$. The new problem within this lemma is, that we have to take care of the additional logical connectives $=^\alpha$ in the extended signature. But fortunately rule *Prim* became automatically extended as well and allows now to introduce the logical connectives $=^\alpha$ at head position analogous to the connectives $\neg, \vee, \Pi^\alpha$ so far.

Lemma 5.14 (Lifting Lemma for $\mathcal{EP}_{fc}$). *Let Φ be a set of clauses, $\mathcal{D}_1$ be a clause, and σ a substitution. We have that:*

1. *For each derivation $\Delta_1 : \Phi_\sigma \vdash^1_{\mathcal{EP}_{fc}} \mathcal{D}_1$ exists a substitution δ, a clause $\mathcal{D}_2$, and a derivation $\Delta_2 : \Phi \vdash_{\mathcal{EP}_{fc}} \mathcal{D}_2$, such that $(\mathcal{D}_2)_\delta \equiv \mathcal{D}_1$.*

2. *For each derivation* $\Delta_1 : \Phi_\sigma \vdash_{\mathcal{EP}_{fc}} \mathcal{D}_1$ *exists a substitution* δ*, a clause* $\mathcal{D}_2$*, and a derivation* $\Delta_2 : \Phi \vdash_{\mathcal{EP}_{fc}} \mathcal{D}_2$*, such that* $(\mathcal{D}_2)_\delta \equiv \mathcal{D}_1$*.*

Proof:

(1) The proof is by case distinction on all rules in $\mathcal{EP}_{fc}$ and in all cases we will motivate that there is a derivation Δ_2 as required.

Res, Prim, Fac, Cnf, Subst The corresponding argumentations in 4.12 are not affected by the additional rules or the availability of primitive equality symbols $=^\alpha$ in the signature.

Leib',Equiv',Para The argumentations for these new rules are analogous to the previous cases.

Triv, Func, Dec, FlexRigid, FlexFlex, Equiv, Leib These cases are indeed affected by the new primitive equality symbols $= \alpha$ in the signature, as we may have a unification constraint $[\mathbf{T}_1 = \mathbf{T}_2]^F$ on the instantiated level, but an ordinary negative literal with a flexible head $[H\ \overline{\mathbf{U}^n}]^F$ on the abstract level. In order to enable the application of the particular unification rule on the abstract layer as well we can use primitive substitution rule *Prim* in connection with rule *Subst* in order to introduce the logical connective $=$ at head position, i.e., we get the abstract unification constraint $[H_1\ \overline{\mathbf{U}^n} = H_2\ \overline{\mathbf{U}^n}]^F$. The remaining argumentations are now analogous to the corresponding ones discussed in 4.12.

(2) The proof is by induction on the length of derivation Δ_1. The base case is trivial and in the induction step we first employ statement (1) and then the induction hypothesis. □

5.5 Completeness

We first analyse Henkin completeness of the extended calculus $\mathcal{EP}_{fc}$. The calculi $\mathcal{EP}_f$ and $\mathcal{EP}$ are then examined in Subsection 5.6.

Before we present the Henkin completeness theorem 5.27 for $\mathcal{EP}_{fc}$, we first adapt the two lemmata 4.15 and 4.15. The first one compares refutability of clause sets in head-normal form with the refutability of clauses in $\beta\eta$-normal form, and the second one discusses some important refutational properties.

Lemma 5.15 (Head-Normal Form). *Let* Φ *be a set of clauses. If* $\Delta : \Phi{\downarrow_{\beta\eta}} \vdash_{\mathcal{EP}_{fc}} \Box$*, then* $\Delta' : \Phi{\downarrow_h} \vdash_{\mathcal{EP}_{fc}} \Box$

Proof: The proof is analogous to lemma 4.13. In the induction step we additionally have to consider the cases where $r \in \{Para, Func', Equiv'\}$, which are analogous to the cases where $r \in \{Prim\} \cup \mathcal{CNF}$.
(One can also prove the other direction, which is not needed in this thesis.) □

Lemma 5.16. *Let* Φ *be a set of clauses and* $\mathbf{A}, \mathbf{B}$ *be formulae. We have that:*

1. *If* $\Phi * [\mathbf{A}]^T \vdash_{\mathcal{EP}_{fc}} \Box$ *and* $\Phi * [\mathbf{B}]^T \vdash_{\mathcal{EP}_{fc}} \Box$*, then* $\Phi * [\mathbf{A} \vee \mathbf{B}]^T \vdash_{\mathcal{EP}_{fc}} \Box$*.*

2. *If* $\Phi * [\mathbf{A}]^T * [\mathbf{B}]^F \vdash_{\mathcal{EP}_{fc}} \Box$ *and* $\Phi * [\mathbf{A}]^F * [\mathbf{B}]^T \vdash_{\mathcal{EP}_{fc}} \Box$*, then* $\Phi * [\mathbf{A} \Leftrightarrow \mathbf{B}]^F$

Proof: Analogous to lemma 4.15. The new rules do not affect the arguments. □

The main completeness proof of calculus $\mathcal{EP}_{fc}$ (cf. 5.27) will employ a generalised paramodulation rule *GPara* that is allowed to operate also on non-proper clauses, but which only rewrites identical terms and employs only unit equations instead of conditional equations.

Definition 5.17 (Generalised Paramodulation Rule).
The generalised paramodulation rule *GPara* is defined as follows (subterm $\mathbf{A}$ od $\mathbf{T}$ must not contain free variables that are bound outside):

$$\frac{[\mathbf{T}[\mathbf{A}_\beta]]^\alpha \vee C \quad [\mathbf{A} =^\beta \mathbf{B}]^T}{[\mathbf{T}[\mathbf{B}]]^\alpha \vee C}\ GPara$$

This rule is not restricted to proper clauses and is furthermore allowed to operate on unification constraints. Thus, *GPara* generalises rule *Para*. On the other hand *GPara* also restricts *Para* as unification is not employed and thus only identical terms can be replaced by applying this rule. Furthermore, the second premise clause is assumed to consists only of a single positive equation literal.

We want to point out, that rule *GPara* is especially designed for the completeness proof 5.27 to ensure the abstract consistency property $\nabla_{\mathfrak{e}}^s$, and the only reason why we introduce the latter restriction for rule *GPara* is, that we want to ease the proofs of the lemmata below as much as possible.

Rule *GPara* will be proven to be admissible[3] in calculus $\mathcal{EP}_{fc}$, which completes the completeness proof. For the proof of the admissibility of rule *GPara* (in fact we even proof a weak derivability property) we employ additional generalised calculus rules: generalised resolution, factorisation and primitive substitution, which are also admissible or even weakly derivable. Instead of formulating these rules as general as possible, we again (analogously to rule *GPara*) restrict ourself to the structures that are needed for our completeness proof.

We want to point out that the investigation of the generalised calculus rules also prepares the investigation of a respective non-normal form calculus.

Definition 5.18 (Generalised Resolution Rules).
The generalised resolution rules $GRes_1$, $GRes_2$, and $GRes_3$ are defined as follows (for all rules we assume $\alpha, \beta \in \{T, F\}$ with $\alpha \neq \beta$, and for $GRes_2$ we assume that $\overline{Y^n} \notin \text{free}(\mathbf{A})$):

$$\frac{[\mathbf{A}_{\overline{\gamma}\to o}\ \overline{\mathbf{T}_\gamma^n}]^\alpha \vee C \quad [\mathbf{A}_{\overline{\gamma}\to o}\ \overline{X_\gamma^n}]^\beta \vee D \quad \alpha \neq \beta}{(C \vee D)_{[\overline{T^n}/\overline{X^n}]}}\ GRes_1$$

$$\frac{[\mathbf{A}_\gamma\ \overline{Y^n}]^\alpha \vee C \quad [X_\gamma\ \overline{\mathbf{T}^n}]^\beta \vee D \quad \alpha \neq \beta}{(C \vee D)_{[\mathbf{A}/X, \overline{\mathbf{T}^n}/\overline{Y^n}]}}\ GRes_2$$

$$\frac{[\mathbf{A}_\gamma\ \overline{\mathbf{T}^n}]^\alpha \vee C \quad [X_\gamma\ \overline{Y^n}]^\beta \vee D \quad \alpha \neq \beta}{(C \vee D)_{[\mathbf{A}/X, \overline{\mathbf{T}^n}/\overline{Y^n}]}}\ GRes_3$$

[3] Remember the definition of admissible and derivable in our refutation approach: A rule r is called *admissible* in one of our resolution calculi R, iff adding rule r to R does not increase the set of refutable formulae. Furthermore, a rule r is called *derivable* in R, iff each application of rule r can be replaced by an alternative derivation in R.

These rules are not restricted to proper clauses and in this sense they extend rule *Res*. But the new rules are defined within special contexts only, and in this sense they also restrict the rule *Res*.

Definition 5.19 (Generalised Primitive Substitution Rules). The generalised primitive substitution rule is defined by (H' is a new free variable)

$$\frac{[H\ \mathbf{T}_\gamma]^\alpha \vee C \qquad \mathbf{G} \equiv \begin{cases} \neg \text{ or } (\vee\ H'_o) \text{ or } \lambda X.\ X & \text{if } \gamma \equiv o \\ \vee & \text{if } \gamma \equiv o \rightarrow o \end{cases}}{([H\ \mathbf{T}_\gamma]^\alpha \vee C)_{[\mathbf{G}/H]}}\ GPrim$$

This rule is not restricted to proper clauses, but note that it applies very special instantiations only and that these instantiations are less general as usual partial bindings. E.g., the usual imitation binding $\lambda X_o.\ \neg(Q^1\ X)$ is combined with the subsequent projection binding $\lambda Y.\ Y$ for Q^1, such that we get $\lambda X_o.\ \neg X$, or simply $\neg$ as combined binding. An imitation binding for Π is avoided as it is not needed in the completeness proof, i.e., the proof that the generalised resolution rules — and thus the generalised paramodulation rule — are weakly derivable in $\mathcal{EP}_{fc}$.

Definition 5.20 (Generalised Factorisation Rule). The generalised factorisation rule *GPrim* is defined by

$$\frac{[\mathbf{A}]^\alpha \vee [\mathbf{B}]^\alpha \vee C \quad \mathbf{A}_\sigma \equiv_\alpha \mathbf{B}_\sigma}{([\mathbf{A}]^\alpha \vee C)_\sigma}\ GFac$$

This rule is not restricted to proper clause but only factorises syntactically unifiable literals.

We will now show, that all our generalised calculus are in a weak sense — modulo subsequent clause normalisation and a kind of lifting — derivable in calculus $\mathcal{EP}_{fc}$. The proof for the weak derivability of *GPara* thereby employs the generalised resolution rule $GRes_1$. $GRes_2$ and $GRes_3$ are only needed to ensure the weak derivability property for the generalised resolution rules itself. This also holds for the generalised factorisation and primitive substitution rule.

Remark 5.21. One could omit all the generalised rules and prove the crucial substitutivity property $\nabla_{\mathfrak{e}}^s$ in the main completeness proof of $\mathcal{EP}_{fc}$. Most likely nested induction proofs will then be needed. Thereby we would loose clarity and also the interesting information on the derivability of the special form generalised calculus rules defined above.

Lemma 5.22 (Weak Derivability of *GFac*). *Let $\mathcal{C}_1,\mathcal{C}_2$ be clauses and $\mathcal{D}$ be a proper clause. If $\Delta_1 : \mathcal{C}_1 \vdash^{GFac} \mathcal{C}_2 \vdash_{\mathcal{CNF}} \mathcal{D}$, then there is a derivation $\Delta_2 : \mathcal{C}_1 \vdash_{\mathcal{EP}_{fc}} \mathcal{D}$.*

Proof: The proof is by induction on the length of the $\mathcal{CNF}$-derivation. In the base case the clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ must be proper, such that we can employ the non-generalised rules *Fac* and then full higher-order unification $\mathcal{UNI}$ to replace *GFac*. In the induction step at least one of the literals of clause rest C must be non-proper. Without loss of generality we can therefore assume, that the first step in the $\mathcal{CNF}$ derivation operates on one of the literals in C. We can now switch this first $\mathcal{CNF}$-step with the application of *GPara* and apply the induction hypotheses, which leads to the assertion. We want to point out, that we need to employ the conventions stated in Remark 4.19 in this proof. □

Lemma 5.23 (Weak Derivability of *GPrim*). *Let $\mathcal{C}_1, \mathcal{C}_2$ be clauses and $\mathcal{D}$ be a proper clause. If $\Delta_1 : \mathcal{C}_1 \vdash^{GPrim} \mathcal{C}_2 \vdash_{\mathcal{CNF}} \mathcal{D}$, then there exists a substitution σ and a derivation $\Delta_2 : \mathcal{C}_1 \vdash_{\mathcal{EP}_{fc}} \mathcal{E}$, such that $\mathcal{E}_\sigma \equiv_\alpha \mathcal{D}$.*

Proof: Analogously to *GFac* the proof is by induction on the length of the $\mathcal{CNF}$-derivation. In the base case the clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ must be proper, and thus for all instantiations we can employ the normal primititive substitution rule *Prim* instead. Note that the partial bindings introduced by *Prim* are more general than the quite special instantiations of rule *GPrim*, and consequently in all this cases we have that the result of *Prim* is indeed more general than the application of *GPrim*. The induction step is analogous to the respective argumentation for *GFac* in lemma 5.22 above. □

Lemma 5.24 (Weak Derivability of $GRes_1$, $GRes_2$, and $GRes_3$).

1. *Let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ be clauses and $\mathcal{C}_4$ be a proper clause, such that $\Delta_1 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^r \mathcal{C}_3 \vdash_{\{GFac\}} \mathcal{C}_4 \vdash_{\mathcal{CNF}} \mathcal{D}$ for $r \in \{GRes_1, GRes_2, GRes_3\}$. If both resolution literals of the generalised resolution step (i.e., the resolution literals in $\mathcal{C}_1$ and $\mathcal{C}_2$) are proper, then there exists $\Delta_2 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{D}$.*

2. *Let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ be clauses and $\mathcal{D}$ be a proper clause, such that $\Delta_1 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^r \mathcal{C}_3 \vdash_{\{GFac\}} \mathcal{C}_4 \vdash_{\mathcal{CNF}} \mathcal{D}$ for $r \in \{GRes_1, GRes_2, GRes_3\}$. Then there exists a substitution σ and a derivation $\Delta_2 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{E}$, such that $\mathcal{E}_\sigma \equiv_\alpha \mathcal{D}$.*

Proof:
(1) The proof is by induction on the length of the $\mathcal{CNF}$-derivation. In the base case the clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ must be proper (i.e., all literals and not just the resolution literals are proper) and thus the application of the generalised resolution rule can be replaced by an application of the ordinary resolution rule *Res* and subsequent eager unification. Furthermore the *GFac* steps can be replaced by an $\mathcal{EP}_{fc}$ derivation as guaranteed by lemma 5.22.

In the step case we simply switch[4] the generalised resolution step with the first clause normalisation step (which is obviously applicable either to $\mathcal{C}_1$ or $\mathcal{C}_2$ as well and does not operate on $[\mathbf{A}]^T$ or $[\mathbf{B}]^T$ as these literals are assumed to be proper). The assertion then follows by induction hypotheses.
(2) The proof is by induction on the number of logical connectives in the resolution literals. In the base case, where both resolution literals must be proper, the conclusion follows immediately by (1). In the induction step we examine the three rules separately:

$\underline{GRes_1}$: Note that in this case the clauses $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ are of form $\mathcal{C}_1 : [\mathbf{A}_{\overline{\gamma} \to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha \vee C$, $\mathcal{C}_2 : [\mathbf{A}_{\overline{\gamma} \to o}\ \overline{X^n_\gamma}]^\beta \vee D$, and $\mathcal{C}_3 : (C \vee D)_{[\overline{T^n}/\overline{X^n}]}$, where $\overline{X^n} \notin \text{free}(\mathbf{A})$. We have to consider the following cases: $[\mathbf{A}_{\overline{\gamma} \to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha$ can be (i) proper (i.e., the logical connectives occur not at head position), (ii) of form (a^α): $[\vee\ \mathbf{T}_1\ \mathbf{T}_2]^\alpha$ for $n \equiv 2$, (iii) of form (a^α): $[\neg\ \mathbf{T}]^\alpha$, (b^α): $[\Pi\ \mathbf{T}]^\alpha$, (c^α): $[(\vee\ \mathbf{M})\ \mathbf{T}]^\alpha$ for $n \equiv 1$, or (iv) of form (a^α): $[\neg\ \mathbf{M}]^\alpha$, (b^α): $[\Pi\ \mathbf{M}]^\alpha$, (c^α): $[\vee\ \mathbf{M}\ \mathbf{N}]^\alpha$ for $n \equiv 0$. As representatives we discuss here the cases where $[\mathbf{A}_{\overline{\gamma} \to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha$ is of form (iv)(c^T): $[\vee\ \mathbf{M}\ \mathbf{N}]^T$ for $n \equiv 0$, (iii)(b^T): $[\Pi\ \mathbf{T}]^T$, and (iii)(b^F): $[\Pi\ \mathbf{T}]^F$ for $n \equiv 1$. Note that case (i) is trivial and follows by (1).

[4] Note that we here need to employ the conventions of Remark 4.19.

(iv)(c^T) In this case clause $\mathcal{C}_2$ must be of form $[\vee\ \mathbf{M}\ \mathbf{N}]^F \vee D$. We first perform the following clause normalisation steps:

$$\frac{\mathcal{C}_1 : [\vee\ \mathbf{M}\ \mathbf{N}]^T \vee C}{\mathcal{C}_5 : [\mathbf{M}]^T \vee [\mathbf{N}]^T \vee C}\ \vee^T \qquad \text{and} \qquad \frac{\mathcal{C}_2 : [\vee\ \mathbf{M}\ \mathbf{N}]^F \vee D}{\begin{array}{c}\mathcal{C}_6 : [\mathbf{M}]^F \vee D\\ \mathcal{C}_7 : [\mathbf{N}]^F \vee D\end{array}}\ \vee_l^F, \vee_r^F$$

Thus, we can derive $\mathcal{C}_3$ (and consequently $\mathcal{C}_4$) also from clauses $\{\mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7\}$ with $GRes_1$:

$$\cfrac{\cfrac{\cfrac{\mathcal{C}_1 : [\vee\ \mathbf{M}\ \mathbf{N}]^T \vee C}{\mathcal{C}_5 : [\mathbf{M}]^T \vee [\mathbf{N}]^T \vee C}\ \vee^T \quad \cfrac{\mathcal{C}_2 : [\vee\ \mathbf{M}\ \mathbf{N}]^F \vee D}{\mathcal{C}_6 : [\mathbf{M}]^F \vee D}\ \vee_l^F}{\mathcal{C}_8 : [\mathbf{N}]^T \vee C \vee D}\ GRes_1 \quad \cfrac{\mathcal{C}_2 : [\vee\ \mathbf{M}\ \mathbf{N}]^F \vee D}{\mathcal{C}_7 : [\mathbf{N}]^F \vee D}\ \vee_r^F}{\begin{array}{c}\mathcal{C}_3 : C \vee D\\ \vdots\ \mathcal{CNF}\\ \mathcal{C}_4\end{array}}\ GRes_1, GFac^*$$

By induction hypotheses (applied to the latter application of $GRes_1$) we know that there is a derivation of the form

$$\cfrac{\cfrac{\cfrac{\mathcal{C}_1}{\mathcal{C}_5}\ \vee^T \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_6}\ \vee_l^F}{\mathcal{C}_8}\ GRes_1 \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_7}\ \vee_r^F}{\begin{array}{c}\vdots\ \mathcal{EP}_{fc}\\ \mathcal{C}_4'\end{array}}$$

such that $\mathcal{C}_4'$ is more general than $\mathcal{C}_4$. Our aim is to apply the induction hypotheses now also to the first application of $GRes_1$, but unfortunately the preconditions are not met, as $\mathcal{C}_8$ is not necessarily a proper clause. Therefore we first apply lemma 5.13, which gives us that

$$\cfrac{\begin{array}{c}\cfrac{\cfrac{\mathcal{C}_1}{\mathcal{C}_5}\ \vee^T \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_6}\ \vee_l^F}{\mathcal{C}_8}\ GRes_1\\ \vdots\ \mathcal{CNF}\\ \mathcal{D}_1\end{array} \quad \dots \quad \begin{array}{c}\cfrac{\cfrac{\mathcal{C}_1}{\mathcal{C}_5}\ \vee^T \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_6}\ \vee_l^F}{\mathcal{C}_8}\ GRes_1\\ \vdots\ \mathcal{CNF}\\ \mathcal{D}_n\end{array} \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_7}\ \vee_r^F}{\begin{array}{c}\vdots\ \mathcal{EP}_{fc}\\ \mathcal{C}_4'\end{array}}$$

where $\mathcal{CNF}(\mathcal{C}_8) \equiv \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$. The preconditions for the induction hypotheses are now met for $\mathcal{D}_1, \dots, \mathcal{D}_n$ and by applying the induction hypotheses for n times we get

$$\begin{array}{c}\cfrac{\cfrac{\mathcal{C}_1}{\mathcal{C}_5}\ \vee^T \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_6}\ \vee_l^F}{\begin{array}{c}\vdots\ \mathcal{EP}_{fc}\\ \mathcal{D}_1'\end{array}}\end{array} \quad \dots \quad \begin{array}{c}\cfrac{\cfrac{\mathcal{C}_1}{\mathcal{C}_5}\ \vee^T \quad \cfrac{\mathcal{C}_2}{\mathcal{C}_6}\ \vee_l^F}{\begin{array}{c}\vdots\ \mathcal{EP}_{fc}\\ \mathcal{D}_n'\end{array}}\end{array}$$

such that the $\mathcal{D}_i'$ ($i \equiv 1, \dots, n$) are more general than the $\mathcal{D}_i$. And as we already know that $\{\mathcal{D}_1, \dots, \mathcal{D}_n, \mathcal{C}_8\} \vdash_{\mathcal{ER}_{fc}} \mathcal{C}_4'$ we get with the lifting lemma for $\mathcal{ER}_{fc}$ (cf. 5.14), that there must be a derivation $\{\mathcal{D}_1', \dots, \mathcal{D}_n', \mathcal{C}_8\} \vdash_{\mathcal{ER}_{fc}} \mathcal{C}_4''$, where $\mathcal{C}_4''$ is more general than $\mathcal{C}_4'$ and $\mathcal{C}_4$, which completes the proof.

(iii)(b^T) Obviously we have that (note that $\mathcal{C}_2$ must be of form $[\Pi\ X]^F \vee D$)

$$\begin{array}{c}\dfrac{\dfrac{\mathcal{C}_1 : [\Pi\ \mathbf{T}_{\alpha\to o}]^T \vee C}{\mathcal{C}_5 : [\mathbf{T}\ Y_\alpha]^T \vee C}\,\Pi^T \qquad \dfrac{\mathcal{C}_2 : [\Pi\ X_{\alpha\to o}]^F \vee D}{\mathcal{C}_6 : [X\ \mathbf{S}_\alpha]^F \vee D}\,\Pi^F}{\mathcal{C}_3 : (C \vee D)_{[\mathbf{T}/X,\mathbf{S}/Y]}}\,GRes_2 \\ \vdots\ \mathcal{CNF} \\ \mathcal{C}_4\end{array}$$

In this derivation Y is an new variable and $\mathbf{S}$ a new Skolem term of appropriate type. Induction hypotheses is applicable and we get that

$$\begin{array}{c}\dfrac{\dfrac{\mathcal{C}_1 : [\Pi\ \mathbf{T}]^F \vee C}{\mathcal{C}_5 : [\mathbf{T}\ Y]^F \vee C}\,\neg^F \qquad \dfrac{\mathcal{C}_2 : [\Pi\ X]^T \vee D}{\mathcal{C}_6 : [X\ \mathbf{S}]^T \vee D}\,\Pi^F}{\vdots\ \mathcal{EP}_{fc}} \\ \mathcal{C}_4'\end{array}$$

where $\mathcal{C}_4'$ is more general than $\mathcal{C}_4$. This proves the assertion.

(iii)(b^F) Obviously we have that (note that $\mathcal{C}_2$ must be of form $[\Pi\ X]^F \vee D$)

$$\begin{array}{c}\dfrac{\dfrac{\mathcal{C}_1 : [\Pi\ \mathbf{T}_{\alpha\to o}]^F \vee C}{\mathcal{C}_5 : [\mathbf{T}\ \mathbf{S}_\alpha]^T \vee C}\,\Pi^T \qquad \dfrac{\mathcal{C}_2 : [\Pi\ X_{\alpha\to o}]^T \vee D}{\mathcal{C}_6 : [X\ Y_\alpha]^F \vee D}\,\Pi^F}{\mathcal{C}_3 : (C \vee D)_{[\mathbf{T}/X,\mathbf{S}/Y]}}\,GRes_3 \\ \vdots\ \mathcal{CNF} \\ \mathcal{C}_4\end{array}$$

where Y is an new variable and $\mathbf{S}$ a new Skolem term of appropriate type. Induction hypotheses is applicable and we get that

$$\begin{array}{c}\dfrac{\dfrac{\mathcal{C}_1 : [\Pi\ \mathbf{T}]^F \vee C}{\mathcal{C}_5 : [\mathbf{T}\ \mathbf{S}]^F \vee C}\,\neg^F \qquad \dfrac{\mathcal{C}_2 : [\Pi\ X]^T \vee D}{\mathcal{C}_6 : [X\ Y]^T \vee D}\,\Pi^F}{\vdots\ \mathcal{EP}_{fc}} \\ \mathcal{C}_4'\end{array}$$

where $\mathcal{C}_4'$ is more general than $\mathcal{C}_4$. This proves the assertion.

$\underline{GRes_2}$: In this case, the clauses $\{\mathcal{C}_1,\mathcal{C}_2,\mathcal{C}_3\}$ are of form $\mathcal{C}_1 : [\mathbf{A}\ \overline{Y^n}]^\alpha \vee C$, $\mathcal{C}_2 : [X\ \overline{\mathbf{T}^n}]^\beta \vee D$, $\mathcal{C}_3 : (C \vee D)_{[\mathbf{A}/X,\overline{\mathbf{T}^n}/\overline{Y^n}]}$, where $n \in \{0,1\}$. The argumentation is similar to the above case for $GRes_1$, and we have to consider the following cases: $[\mathbf{A}\ \overline{Y^n}]^\alpha$ can be (i) proper, (ii) of form (a^α): $[\vee\ Y_1\ Y_2]^\alpha$ for $n \equiv 2$, (iii) of form (a^α): $[\neg\ Y]^\alpha$, (b^α): $[\Pi\ Y]^\alpha$, (c^α): $[(\vee\ \mathbf{M})\ Y]^\alpha$ for $n \equiv 1$, or (iv) of form (a^α): $[\neg\ \mathbf{M}]^\alpha$, ($b^\alpha$): $[\Pi\ \mathbf{M}]^\alpha$, (c^α): $[\vee\ \mathbf{M}\ \mathbf{N}]^\alpha$ for $n \equiv 0$. Case (i) again follows immediately by (1), as both resolution literals must be proper. And as a representative for the other cases we exemplarily discuss the induction step for (iii)(a^F), i.e., in this case the two resolution literals $[\mathbf{A}\ \overline{Y^n}]^\alpha$ and $[X\ \overline{T^n}]^\beta$) are of form $[(\neg\ Y)]^F$ and $[X\ \mathbf{T}]^T$.

We have that

$$\dfrac{\dfrac{\mathcal{C}_1 : [\neg\ Y]^F \lor C}{\mathcal{C}_5 : [Y]^T \lor C}\,\neg^F \qquad \dfrac{\dfrac{\mathcal{C}_2 : [X\ \mathbf{T}]^T \lor D}{\mathcal{C}_6 : ([X'\ \mathbf{T}]^F \lor D)_{[\lambda Y.\ \neg(X'Y)/X]}}\,GPrim}{\mathcal{C}_7 : ([\mathbf{T}]^F \lor D)_{[\neg/X]}}\,GPrim}{\begin{array}{c}\mathcal{C}_3 : (C \lor D)_{[\neg/X,\mathbf{T}/Y]}\\ \vdots\ \mathcal{CNF}\\ \mathcal{C}_4\end{array}}\,GRes_2$$

By Induction hypotheses we get that

$$\dfrac{\dfrac{\mathcal{C}_1 : [\neg\ Y]^F \lor C}{\mathcal{C}_5 : [Y]^T \lor C}\,\neg^F \qquad \dfrac{\dfrac{\mathcal{C}_2 : [X\ \mathbf{T}]^T \lor D}{\mathcal{C}_6 : ([\neg\mathbf{T}]^T \lor D)_{[\lambda Y.\ \neg/X]}}\,GPrim}{\mathcal{C}_7 : ([\mathbf{T}]^F \lor D)_{[\neg/X]}}\,\neg^T}{\begin{array}{c}\vdots\ \mathcal{EP}_{fc}\\ \mathcal{C}'_4\end{array}}$$

where $\mathcal{C}'_4$ is more general than $\mathcal{C}_4$. By applying weak derivability of *GPrim* (cf. 5.23) two times in connection with the lifting lemma for $\mathcal{EP}_{fc}$ (cf. 5.14), we now get that $\{\mathcal{C}_1,\mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{C}'''_4$, where $\mathcal{C}'''_4$ is obviously more general than $\mathcal{C}_4$, and which proves the assertion.

$\underline{GRes_3}$: In this case the clauses $\{\mathcal{C}_1,\mathcal{C}_2,\mathcal{C}_3\}$ are of form $\mathcal{C}_1 : [\mathbf{A}\ \overline{\mathbf{T}^n}]^\alpha \lor C$, $\mathcal{C}_2 : [X\ \overline{Y^n}]^\beta \lor D$, $\mathcal{C}_3 : (C \lor D)_{[\mathbf{A}/X,\overline{\mathbf{T}^n}/\overline{Y^n}]}$, where $n \in \{0,1\}$. We have to consider the following cases (analogous to $GRes_1$): $[\mathbf{A}_{\overline{\gamma}\to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha$ can be (i) proper (i.e., the logical connectives occur not at head position), (ii) of form (a^α): $[\lor\ \mathbf{T}_1\ \mathbf{T}_2]^\alpha$ for $n \equiv 2$, (iii) of form (a^α): $[\neg\ \mathbf{T}]^\alpha$, ($b^\alpha$): $[\Pi\ \mathbf{T}]^\alpha$, (c^α): $[(\lor\ \mathbf{M})\ \mathbf{T}]^\alpha$ for $n \equiv 1$, or (iv) of form (a^α): $[\neg\ \mathbf{M}]^\alpha$, ($b^\alpha$): $[\Pi\ \mathbf{M}]^\alpha$, (c^α): $[\lor\ \mathbf{M}\ \mathbf{N}]^\alpha$ for $n \equiv 0$. Case (i) again follows by (1). As a representative for the other cases we briefly sketch induction step for (iii)(c^F), i.e., the resolution literals $[\mathbf{A}\ \overline{Y^n}]^\alpha$ and $[X\ \overline{Y^n}]^\beta$ are of form $[(\lor\ \mathbf{M})\ \mathbf{T}]^F$ and $[X\ Y]^T$. We first employ rule *GPrim* and obtain

$$\dfrac{\mathcal{C}_2 : [X\ Y]^T \lor D}{\mathcal{C}_3 : ([(\lor\ H)\ Y]^T \lor \mathcal{D})_{[(\lor\ \mathbf{M})/X]}}\,GPrim$$

The rest of the proof is analogous to the case (iv)(c^T) for $GRes_1$ as discussed before. The main difference is, that we additionally have to get rid of the initial *GPrim* step here, which can be done analogous to the case discussed for $GRes_2$ above. □

The following paramodulation lemma now shows, that the generalised paramodulation rule *GPara* is weakly derivable in $\mathcal{EP}_{fc}$, too. The proof employs the (weakly derivable) generalised resolution rules $GRes_1$ and $GRes_3$.

Lemma 5.25 (Weak Derivability of *GPara*).

1. *Let $\mathcal{C}_1$ be a clause and $\mathcal{C}_2$ be a proper clause.*

   (a) *If $\mathcal{C}_1 : [\mathbf{A}]^\alpha \lor D$ and $\mathcal{C}_2 : [\mathbf{A} = \mathbf{B}]^T$, then there exists $\Delta_1 : \{\mathcal{C}_1,\mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{D} : [\mathbf{C}]^\alpha \lor E$, and a substitution σ, such that $\mathcal{D}_\sigma \equiv_\alpha [\mathbf{B}]^\alpha \lor D$ (i.e., $\mathcal{D}$ is more general than the clause we would obtain by a respective generalised paramodulation step)*

(b) *If* $\mathcal{C}_1 : [\mathbf{A}_{\overline{\gamma}\to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha \vee D$ *and* $\mathcal{C}_2 : [(\mathbf{A}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma}) = (\mathbf{B}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma})]^T$*, where* $\overline{X^n_\gamma} \notin$ *free*$(\overline{\mathbf{T}^n_\gamma}) \cup$ *free*$(\mathbf{A})$*, then there exists* $\Delta_2 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{D} : [\mathbf{C}_{\overline{\gamma}\to o}\ \overline{\mathbf{D}^n_\gamma}]^\alpha \vee E$ *and a substitution* σ*, such that* $\mathcal{D}_\sigma \equiv_\alpha [\mathbf{B}_{\overline{\gamma}\to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha \vee D$*.*

2. *Let* $\mathcal{C}_1 : [\mathbf{T}[\mathbf{A}]_p]^\alpha \vee D$*,* $\mathcal{C}_2 : [\mathbf{A} = \mathbf{B}]^T$ *and* $\mathcal{C}_3 : [\mathbf{T}[\mathbf{B}]_p]^\alpha \vee D$*, such that there is a derivation* $\Delta_1 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^{GPara} \mathcal{C}_3 \vdash_{\mathcal{CNF}} \mathcal{C}_4$*, for a proper clause* $\mathcal{D}$ *derived by clause normalisation from* $\mathcal{C}_3$*. Then there exists a derivation* $\Delta_2 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{D}'$ *and a substitution* σ*, such that* $\mathcal{D}'_\sigma \equiv \mathcal{D}$*.*

Proof:
(1) We first concentrate on case (1a) and consider the following $\mathcal{CNF}$-derivation:

$$\begin{array}{l}
\dfrac{\mathcal{C}_2 : [\mathbf{A} = \mathbf{B}]^T}{\mathcal{C}_4 : [\mathbf{A}] \Leftrightarrow [\mathbf{B}]^T}\ Equiv' \\
\quad\quad \vdots\ \mathcal{CNF} \\
\mathcal{C}_5 : [\mathbf{A}]^F \vee [\mathbf{B}]^T \\
\mathcal{C}_6 : [\mathbf{A}]^T \vee [\mathbf{B}]^F \quad .
\end{array}$$

Depending on the polarity α of literal $[\mathbf{A}]^\alpha$ in clause $\mathcal{C}_1$, we now apply the generalised resolution rule $GRes_1$ either to $\mathcal{C}_1$ and $\mathcal{C}_5$, or to $\mathcal{C}_1$ and $\mathcal{C}_6$, thereby deriving clause $\mathcal{C}_3 : [\mathbf{B}]^\alpha \vee D$. By weak derivability of $GRes_1$ (cf. 5.24(2)) we get a derivation $\Delta_1 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{D}$ as required, i.e., $\mathcal{D}$ is more general than $[\mathbf{B}]^\alpha \vee D$. The case (1b) is analogous as

$$\begin{array}{l}
\dfrac{\mathcal{C}_2 : [(\mathbf{A}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma}) = (\mathbf{B}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma})]^T}{\mathcal{C}_4 : [(\mathbf{A}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma}) \Leftrightarrow (\mathbf{B}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma})]^T}\ Equiv' \\
\quad\quad\quad\quad \vdots\ \mathcal{CNF} \\
\mathcal{C}_5 : [\mathbf{A}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma}]^F \vee [\mathbf{B}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma})]^T \\
\mathcal{C}_6 : [\mathbf{A}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma}]^T \vee [\mathbf{B}_{\overline{\gamma}\to o}\ \overline{X^n_\gamma})]^F
\end{array}$$

and thus the application of $GRes_1$ to $\mathcal{C}_1$ and $\mathcal{C}_5$, or to $\mathcal{C}_1$ and $\mathcal{C}_6$ derives clause $\mathcal{C}_3 : [\mathbf{B}_{\overline{\gamma}\to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha \vee D$. Again the assertion follows by weak derivability of $GRes_1$ (cf. 5.24(2)).

(2) The tedious proof is by induction on the length n of the embedded $\mathcal{CNF}$ derivation $\mathcal{C}_3 \vdash_{\mathcal{CNF}} \mathcal{C}_4$.

$\underline{n = 0}$: In the base case $\mathcal{C}_3$ must be a proper clause, i.e., the clause rests D consist only of proper literals and the literal $[\mathbf{T}[\mathbf{B}]_p]^\alpha$ is proper. We now consider the possible cases for literal $[\mathbf{T}[\mathbf{A}]]^\alpha$.

1. $[\mathbf{T}[\mathbf{A}]_p]^T$ is a proper literal and $\mathcal{C}_1$ is a proper clause. Instead of generalised paramodulation we can apply in this case the standard paramodulation rule *Para* to $\mathcal{C}_1$ in order to derive $\mathcal{C}_4$ by $\Delta_2 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^{Para} [\mathbf{T}[\mathbf{B}]_p]^T \vee D_1 \vee [\mathbf{A} = \mathbf{A}]^F \vdash^{Triv} \mathcal{C}_4$. In this case σ is the empty substitution.

2. $[\mathbf{T}[\mathbf{A}]_p]^F$ is a proper literal and $\mathcal{C}_1$ is a proper clause. If $[\mathbf{T}[\mathbf{A}]_p]^F$ is not a unification constraint, then the argumentation is analogous to the above case. In case $[\mathbf{T}[\mathbf{A}]_p]^F$ is a unification constraint, it is (without loss of generality) of form $[\mathbf{T}_1[\mathbf{A}] = \mathbf{T}_2]^F$. In this case we employ a derivation that is analogous to the one already discussed in

remark 5.2:

$$\begin{array}{ll}
Leib(\mathcal{C}_1), \mathcal{CNF}: & \mathcal{D}_1 : C \vee [p\ \mathbf{T}_1[\mathbf{A}_\alpha]]^T \\
 & \mathcal{D}_2 : C \vee [p\ \mathbf{T}_2]^F \\
Para(\mathcal{D}_1, \mathcal{C}_2), Triv: & \mathcal{D}_3 : C \vee [p\ \mathbf{T}_1[\mathbf{B}_\alpha]]^T \\
Res(\mathcal{D}_3, \mathcal{D}_2), Fac, Triv: & \mathcal{D}_4 : C \vee D \vee [(p\ \mathbf{T}_1[\mathbf{B}_\alpha]) = (p\ \mathbf{T}_2)]^F \\
Dec(\mathcal{D}_4), Triv: & \mathcal{D}_5 : C \vee D \vee [\mathbf{T}_1[\mathbf{B}_\alpha] = \mathbf{T}_2]^F
\end{array}$$

Again we have that σ is the empty substitution.

3. $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ is not a proper literal, i.e., has a logical connective different from $=$ at head position. We consider the possible subterm positions p of term $\mathbf{A}$ in $\mathbf{T}$ (noted $\mathbf{T}[\mathbf{A}]_p$).

   As this would contradict our assumption ($[\mathbf{T}[\mathbf{B}]_p]^\alpha$ is a proper literal), we know that for literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ the following cases are impossible: $[\neg\ (\mathbf{M}[\mathbf{A}]_{p'})]^\alpha$, $[\vee\ (\mathbf{M}[\mathbf{A}]_{p'})\ \mathbf{N}]^\alpha$, $[\vee\ \mathbf{M}\ (\mathbf{N}[\mathbf{A}]_{p'})]^\alpha$ and $[\Pi\ (\mathbf{M}[\mathbf{A}]_{p'})]^\alpha$. The remaining cases are:

   (a) Literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ has form $[\overbrace{\neg\ \mathbf{M}}^{\mathbf{A}}]^\alpha$, $[\overbrace{\vee\ \mathbf{M}\ \mathbf{N}}^{\mathbf{A}}]^\alpha$ or $[\overbrace{\Pi\ \mathbf{M}}^{\mathbf{A}}]^\alpha$. Then $\mathcal{C}_2$ is of form $[(\neg\mathbf{M}) = \mathbf{B}]^T$, $[(\vee\ \mathbf{M}\ \mathbf{N}) = \mathbf{B}]^T$ and $[(\Pi\ \mathbf{M}) = \mathbf{B}]^T$ respectively. For all these cases the assertion now follows by (1a).

   (b) Literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ has form $[\overbrace{\neg}^{\mathbf{A}}\ \mathbf{M}]^\alpha$, $[\overbrace{\vee}^{\mathbf{A}}\ \mathbf{M}\ \mathbf{N}]^\alpha$, $[\overbrace{\Pi}^{\mathbf{A}}\ \mathbf{M}]^\alpha$ or $[\overbrace{(\vee\ \mathbf{M})}^{\mathbf{A}}\ \mathbf{N}]^\alpha$. Then analogous to above $\mathcal{C}_2$ is of form $[\neg = \mathbf{B}]^{\mathbf{T}}$, $[\vee = \mathbf{B}]^T$, $[\Pi = \mathbf{B}]^T$ and $[(\vee\ \mathbf{M}) = \mathbf{B}]^T$ respectively. Applying positive extensionality rule *Func'* to these clauses leads to $[(\neg\ X) = (\mathbf{B}\ X)]^T$, $[(\vee\ X\ Y) = (\mathbf{B}\ X\ Y)]^T$, $[(\Pi\ P) = (\mathbf{B}\ P)]^T$ and $[(\vee\ \mathbf{M}\ X) = (\mathbf{B}\ X)]^T$. For all these cases the assertion now follows by (1b).

$\underline{n > 0:}$ In the induction step we know that literal $[\mathbf{T}[\mathbf{B}]_p]^\alpha$ must have a logical connective at head position, as otherwise the length of the $\mathcal{CNF}$-derivation $\mathcal{C}_3 \vdash_{\mathcal{CNF}} \mathcal{C}_4$ is zero. We now distinguish the following two cases:

1. At least one of the literals in D is not proper. In this case, it is obvious that we can reorder the $\mathcal{CNF}$-derivation, such that one of the clause normalisation steps that modifies D comes first.[5] Without loss of generality, let us assume that $\mathcal{CNF}$-rule r transforms in step k ($1 < k \leq n$) of derivation Δ_1 the clause rest D into D'. Then we can reorder derivation Δ_1 and obtain a derivation $\Delta_3 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^{GPara} \mathcal{C}_3 \vdash^r \mathcal{C}_3' : [\mathbf{T}[\mathbf{B}]_p]^\alpha \vee D' \vdash_{\mathcal{CNF}} \mathcal{C}_4$, such that rule r is applied first. Obviously, we can also switch the first two steps in Δ_3, such that we get $\Delta_4 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^r \{\mathcal{C}_1' : [\mathbf{T}[\mathbf{A}]_p]^\alpha \vee D', \mathcal{C}_2\} \vdash^{GPara} \mathcal{C}_3' \vdash_{\mathcal{CNF}} \mathcal{C}_4$. Now induction hypotheses is applicable and we get that there is $\mathcal{EP}_{fc}$-derivation Δ_5 : $\{\mathcal{C}_1', \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{C}_4$.

2. D consist only of proper literals. We again examine the structure of literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ and distinguish between all possible subterm positions p of term $\mathbf{A}$ in $\mathbf{T}$.

   (a) Literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ is of form $[\neg\ (\mathbf{M}[\mathbf{A}]_{p'})]^\alpha$, $[\vee\ (\mathbf{M}[\mathbf{A}]_{p'})\ \mathbf{N}]^\alpha$, $[\vee\ \mathbf{M}\ (\mathbf{N}[\mathbf{A}]_{p'})]^\alpha$, or $[\Pi\ (\mathbf{M}[\mathbf{A}]_{p'})]^\alpha$. As the proofs are analogous we only present the case where $\mathcal{C}_1$ is of form $[\vee\ \mathbf{M}\ (\mathbf{N}[\mathbf{A}]_{p'})]^T \vee D$. Consider now derivation Δ_1, which obviously applies

[5] Note that we here need to employ the conventions of Remark 4.19.

the $\mathcal{CNF}$-rule $\vee^T$ first within the normalisation process (note that D contains only proper literals). Thus we have $\Delta_1 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^{GPara} \mathcal{C}_3 \vdash^{\vee^T} \mathcal{C}_3' : [\mathbf{M}]^T \vee [\mathbf{N}[\mathbf{B}]_p]^T \vee D \vdash_{\mathcal{CNF}} \mathcal{C}_4$. We can obviously switch the first two steps in Δ_1, such that we get an alternative derivation[6] $\Delta_1' : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^{\vee^T} \{\mathcal{C}_1' : [\mathbf{M}]^T \vee [\mathbf{N}[\mathbf{A}]_p]^T \vee D, \mathcal{C}_2\} \vdash^{GPara} \mathcal{C}_3' \vdash_{\mathcal{CNF}} \mathcal{C}_4$. By induction hypotheses we now get that there is a derivation $\Delta_2' : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash^{\vee^T} \{\mathcal{C}_1', \mathcal{C}_2\} \vdash_{\mathcal{EP}_{fc}} \mathcal{C}_4$.

(b) Literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ is of form $[\overbrace{\neg\ \mathbf{M}}^{\mathbf{A}}]^\alpha$, $[\overbrace{\vee\ \mathbf{M}\ \mathbf{N}}^{\mathbf{A}}]^\alpha$ or $[\overbrace{\Pi\ \mathbf{M}}^{\mathbf{A}}]^\alpha$. Then $\mathcal{C}_2$ is of form $[(\neg\ \mathbf{M}) = \mathbf{B}]^\alpha$, $[(\vee\ \mathbf{M}\ \mathbf{N}) = \mathbf{B}]^\alpha$ or $[(\Pi\ \mathbf{M}) = \mathbf{B}]^\alpha$. In all cases the assertion follows immediately by (1a).

(c) Literal $[\mathbf{T}[\mathbf{A}]_p]^\alpha$ is of form $[\overbrace{\ \neg\ }^{\mathbf{A}}\ \mathbf{M}]^\alpha$, $[\overbrace{\ \vee\ }^{\mathbf{A}}\ \mathbf{M}\ \mathbf{N}]^\alpha$, $[\overbrace{\ \Pi\ }^{\mathbf{A}}\ \mathbf{M}]^\alpha$ or $[\overbrace{(\vee\ \mathbf{M})}^{\mathbf{A}}\ \mathbf{N}]^\alpha$. Then $\mathcal{C}_2$ is of form $[\neg = \mathbf{B}]^\alpha$, $[\vee = \mathbf{B}]^T$, $[\Pi = \mathbf{B}]^T$ and $[(\vee\ \mathbf{M}) = \mathbf{B}]^T$ respectively. Applying positive extensionality rule *Func'* to these clauses leads to $[(\neg\ X) = (\mathbf{B}\ X)]^\alpha$, $[(\vee\ X\ Y) = (\mathbf{B}\ X\ Y)]^\alpha$, $[(\Pi\ P) = (\mathbf{B}\ P)]^\alpha$ and $[(\vee\ \mathbf{M}\ X) = (\mathbf{B}\ X)]^\alpha$. For all these cases the assertion now follows by (1b).

□

As □ is just a special proper clause we immediately get the following corollary.

Corollary 5.26 (Admissibility of Generalised Paramodulation Rule). *The generalised paramodulation rule* GPara *is admissible in calculus* $\mathcal{EP}_{fc}$.

Theorem 5.27 (Completeness of $\mathcal{EP}_{fc}$). *The calculus $\mathcal{EP}_{fc}$ is complete with respect to Henkin models.*

Proof: Let Γ_Σ be the set of Σ-sentences which cannot be refuted by the calculus $\mathcal{EP}_{fc}$ ($\Gamma_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) | \Phi_{cl} \not\vdash_{\mathcal{EP}_{fc}} \square\}$), then we show that Γ_Σ is a saturated abstract consistency class for Henkin models with primitive equality (cf. Definition 3.18). This entails completeness with the model existence theorem for Henkin models with primitive equality (cf. Theorem 3.29).

First we have to verify that Γ_Σ validates the abstract consistency properties ∇_c, $\nabla_\neg$, ∇_β, $\nabla_\vee$, $\nabla_\wedge$, $\nabla_\forall$, $\nabla_\exists$, $\nabla_\mathfrak{b}$, $\nabla_\mathfrak{q}$, and that Γ_Σ is saturated. For all of these cases, the proofs are identical to the respective argumentations in theorem 4.16. The only difference is that we employ lemmata 5.14 and 5.16 instead of 4.12 and 4.15. Thus, all we need to ensure is the validity of the additional abstract consistency properties $\nabla_\mathfrak{e}^r$ and $\nabla_\mathfrak{e}^s$ for primitive equality.

$\nabla_\mathfrak{e}^r$ $\quad \neg(\mathbf{A} =^\alpha \mathbf{A}) \notin \Phi$.

$\nabla_\mathfrak{e}^s$ $\quad$ If $\mathbf{F}[\mathbf{A}]_p \in \Phi$ and $\mathbf{A} = \mathbf{B} \in \Phi$, then $\Phi * \mathbf{F}[\mathbf{B}]_p \in \Gamma_\Sigma$.

($\nabla_\mathfrak{e}^r$) We have that $[\mathbf{A} =^\alpha \mathbf{A}]^F \vdash^{Triv} \square$, and thus $\neg(\mathbf{A} =^\alpha \mathbf{A})$ cannot be in Φ.
($\nabla_\mathfrak{e}^s$) Analogous to the cases in 4.16 we show the contrapositive of the assertion, i.e., we assume that there is derivation $\Delta_1 : \Phi_{cl} * [\mathbf{F}[\mathbf{B}]_p]^T \vdash_{\mathcal{EP}_{fc}} \square$, and then ensure that $\Delta_2 : \Phi_{cl} * [\mathbf{F}[\mathbf{A}]_p]^T * [\mathbf{A} = \mathbf{B}]^T \vdash_{\mathcal{EP}_{fc}} \square$. Now consider the following $\mathcal{EP}_{fc}$-derivation:

$$\frac{[\mathbf{F}[\mathbf{A}]_p]^T \quad [\mathbf{A} = \mathbf{B}]^T}{[\mathbf{F}[\mathbf{B}]_p]^T}\ GPara$$

[6] Note that we here need to employ the conventions of Remark 4.19.

By Corollary 5.26 the generalised paramodulation rule *GPara* is admissible for calculus $\mathcal{EP}_{fc}$, which guarantees, that there is a derivation $\Delta_2 : \Phi_{cl} * [\mathbf{F}[\mathbf{A}]_p]^T * [\mathbf{A} = \mathbf{B}]^T \vdash_{\mathcal{EP}_{fc}} \Box$, that does not apply *GPara*. This proves the assertion. □

5.6 Theorem Equivalence

We shall now prove that $\mathcal{EP}_{fc}$ and $\mathcal{EP}_f$ are theorem equivalent. Theorem equivalence of $\mathcal{EP}_{fc}$ and $\mathcal{EP}$ is only presented as a conjecture.

Lemma 5.28 (Proper Clauses in $\mathcal{EP}$ and $\mathcal{EP}_f$). *For each non-proper clause $\mathcal{C}$ and clause set Φ, such that $\Phi * \vdash_{\mathcal{EP}_{fc}} \mathcal{C}$, we have $\Phi \vdash_{\mathcal{EP}_f} \mathcal{C}$.*

Proof: Analogous to lemma 4.20. In the induction step, the case for rule *Para* is treated analogously to the cases $r \in \{Res, Fac, Prim\}$, and the rules *Func'* and *Equiv'* are treated analogously to the unification rules. □

As □ is also a proper clause we immediately get the following corollary:

Corollary 5.29 (Theorem Equivalence of $\mathcal{EP}_{fc}$ and $\mathcal{EP}_f$). *The calculi $\mathcal{EP}_{fc}$ and $\mathcal{EP}_f$ are theorem equivalent.*

Conjecture 5.30 (Theorem Equivalence of $\mathcal{EP}$ and $\mathcal{EP}_{fc}$ (or $\mathcal{EP}_f$)). *The calculi $\mathcal{EP}$ and $\mathcal{EP}_{fc}$ (or $\mathcal{EP}_f$) are theorem equivalent.*

The proof for the latter conjecture will most likely be analogous to the proof of the theorem equivalence for calculi $\mathcal{ER}_{fc}$ and $\mathcal{ER}$. This conjecture also gains evidence by the examples carried out with the LEO-prover [BK98b] for extensional higher-order resolution as well as the new challenging examples for extensional higher-order paramodulation discussed in Chapter 8.

Chapter 6

Extensional Higher-Order RUE-Resolution: $\mathcal{ERUE}$

6.1 Resolution on Unification Constraints

In this section we will extend the **R**esolution by **U**nification and **E**quality approach of Digricoli [Dig79] to higher-order logic. The key idea of the resulting calculus $\mathcal{ERUE}$ is to allow resolution and factorisation rules also to operate on unification constraints.[1] More precisely, our approach allows to compute partial E-unifiers with respect to a specified theory E (where E is given in form of a set of unitary or conditional equations in clause form in the search space) by employing resolution on unification within constraint the calculus itself. This is due to the fact that the extensional higher-order resolution approach already realises a test calculus for general higher-order E-pre-unification (or general higher-order E-unification in case we also add rule *FlexFlex*; cf. Remark 4.8). Furthermore, each partial instantiation can be applied to a clause with rule *Subst*. Finally, like in the traditional first-order RUE-resolution approach, the non-solved unification constraints are encoded as still open unification constraints within the particular clauses itself.

Remark 6.1 ***(Incompleteness of a naive RUE-Resolution approach).*** Similar to our naive adaptation of the first-order paramodulation approach, we obtain a Henkin incomplete RUE-resolution approach, if we do not additionally add the positive extensionality axioms. This is illustrated by example $\mathbf{E}_1^{Para}$ already used in the incompleteness proof 5.6 for $\mathcal{EP}$. In a naive $\mathcal{ERUE}$ approach, i.e., without positive extensionality rules, no single rule would be applicable to the contradictory unit clause $\mathcal{C}_1 : [a = \neg a]^T$.

6.2 Basic Definitions

Similar to the approaches $\mathcal{ER}$ and $\mathcal{EP}$ discussed in the previous chapters, we further extend the extensional higher-order RUE-resolution approach by adding rule *FlexFlex* and lifting the single clause normalisation rules to calculus level.

Definition 6.2 (Extensional Higher-Order RUE-Resolution).

[1] The idea to resolve on unification constraints arose while discussing the necessity of Rule *Leib* within extensional higher-order resolution with Frank Pfenning during a stay at Carnegie Mellon University.

$\mathcal{ERUE}$ The calculus $\mathcal{ERUE}$ consists of the rules of calculus $\mathcal{ER}$ (see Figure 4.2) enriched by the positive extensionality rules displayed in Figure 5.2. The most important aspect is that we allow to resolve and factorise on unification constraints. Furthermore we want to point out that unification constraints are assumed to be symmetric which could also be formulated by the following rule:

$$\frac{\mathcal{C}_1 : C \lor [\mathbf{A} = \mathbf{B}]^F}{\mathcal{C}_2 : C \lor [\mathbf{B} = \mathbf{A}]^F}\ Sym$$

We employ this convention despite the fact that the symmetry rule Sym is derivable (c.f. remark 6.3) in $\mathcal{ERUE}$ — as well as in $\mathcal{ER}$ and $\mathcal{EP}$ — as it shortens and eases derivations and is acceptable with respect to its complexity in practice.

$\mathcal{ERUE}_f$ The extension $\mathcal{ERUE}_f$ of calculus $\mathcal{ERUE}$ that employs full higher-order unification instead of higher-order pre-unification is defined as $\mathcal{ERUE}_f := \mathcal{ERUE} \cup \{FlexFlex\}$.

$\mathcal{ERUE}_{fc}$ The calculus $\mathcal{ERUE}_{fc}$ that employs stepwise instead of exhaustive clause normalisation is defined by $\mathcal{ERUE}_{fc} := (\mathcal{ERUE}_f \backslash \{Cnf\}) \cup \mathcal{CNF}$.

We assume furthermore, that the result of each rule application is transformed into head-normal form. A set of formulae Φ is refutable in calculus $\mathcal{ERUE}$, iff there is a derivation $\Delta : \Phi_{cl} \vdash_{\mathcal{ERUE}} \Box$, where $\Phi_{cl} := \{[\mathbf{F}]^T | \mathbf{F} \in \Phi\}$ is the set obtained from Φ by simple pre-clausification.

*Remark 6.3 (**Symmetric Unification Constraints**).* The following derivation shows that the symmetry rule Sym is derivable in $\mathcal{ERUE}$ ($\mathcal{ER}$ and $\mathcal{EP}$):

$$\begin{array}{ll}
Leib(\mathcal{C}_1): & \mathcal{D}_1 : C \lor [p\ \mathbf{A}]^T \\
 & \mathcal{D}_2 : C \lor [p\ \mathbf{B}]^F \\
Res(\mathcal{D}_2, \mathcal{D}_1), Fac, Triv: & \mathcal{D}_3 : C \lor [p\ \mathbf{B} = p\ \mathbf{A}]^F \\
Dec(\mathcal{D}_3), Triv: & \mathcal{C}_2 : C \lor [\mathbf{B} = \mathbf{A}]^F
\end{array}$$

Theorem 6.4 (Soundness of Extensional Higher-Order RUE-Resolution). *The calculi $\mathcal{ERUE}_{fc}$, $\mathcal{ERUE}_f$, and $\mathcal{ERUE}$ are sound with respect to Henkin semantics.*

Proof: Soundness of most of our calculus rules have already been discussed in lemmata 4.9 and 5.12. We additionally have to verify that resolution and factorisation on unification constraints is sound wrt. Henkin semantics. Note that unification constraints are treated as ordinary negative literals with a primitive equality symbol at head position. Thus, there is nothing new to show here and we can employ the standard argumentation for soundness of the resolution and factorisation rule. □

Lemma 6.5 (Proper Derivations). *For each non-proper clause $\mathcal{C}$, proper clause $\mathcal{C}'$, and clause set Φ, such that $\Phi * \mathcal{C} \vdash_{\mathcal{ERUE}_{fc}} \mathcal{C}'$, we have that $\Phi \cup \mathcal{CNF}(\mathcal{C}) \vdash_{\mathcal{EP}_{fc}} \mathcal{C}'$.*

Proof: Analogous to lemmata 4.10 and 5.13. It does not cause any problems that we allow to resolve on unification constraints. □

6.3 Lifting Properties

The clause normalisation lifting lemma 4.11 is not affected by the modifications to the calculus and can be safely employed in the following lifting lemma for calculus $\mathcal{ERUE}_{fc}$.

The proof of the adapted main lifting lemma is analogous to the one presented for $\mathcal{ER}_{fc}$.

Lemma 6.6 (Lifting Lemma for $\mathcal{ERUE}_{fc}$).
Let Φ be a set of clauses, $\mathcal{D}_1$ be a clause and σ a substitution. We have that:

1. *For each derivation $\Delta_1 : \Phi_\sigma \vdash^1_{\mathcal{ERUE}_{fc}} \mathcal{D}_1$, there exists a substitution δ, a clause $\mathcal{D}_2$, and a derivation $\Delta_2 : \Phi \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D}_2$, such that $(\mathcal{D}_2)_\delta \equiv \mathcal{D}_1$.*

2. *For each derivation $\Delta_1 : \Phi_\sigma \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D}_1$ there exists a substitution δ, a clause $\mathcal{D}_2$, and a derivation $\Delta_2 : \Phi \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D}_2$, such that $(\mathcal{D}_2)_\delta \equiv \mathcal{D}_1$. (Note that this claim is stronger than: Φ is refutable by $\mathcal{ERUE}_{fc}$, provided that Φ_σ is.)*

Proof:
(1) $\mathcal{ERUE}_{fc}$ only slightly modifies calculus $\mathcal{EP}_{fc}$ as it does not employ rule *Para* but a slightly extended resolution instead. This does not cause new problems and the argumentation is analogous to 5.14(1). The main idea is again to employ rule *Prim* in order to introduce logical connectives at head position, such that the needed structure becomes available at the abstract level as well.

(2) Analogous to 5.14(2) □

6.4 Completeness

Analogous to the Subsections 4.4 and 5.5, we analyse in this subsection Henkin completeness of calculus $\mathcal{ERUE}_{fc}$. The calculi $\mathcal{ERUE}_f$ and $\mathcal{ERUE}$ are then examined in Subsection 6.5. Like in the case of extensional higher-order resolution, theorem equivalence between $\mathcal{ERUE}$ and $\mathcal{ERUE}_{fc}$ has not been proven formally yet and will only be presented as a conjecture.

We first adapt the two lemmata 5.16 and 5.24. The former compares refutability of clause sets in head-normal form with refutability of clauses in $\beta\eta$-normal form and the latter one discusses some important refutational properties.

Lemma 6.7 (Head-Normal Form).
Let Φ be a set of clauses. If $\Phi{\downarrow_{\beta\eta}} \vdash_{\mathcal{ERUE}_{fc}} \Box$, then $\Phi{\downarrow_h} \vdash_{\mathcal{ERUE}_{fc}} \Box$.

Proof: The proof is analogous to lemma 4.13. In the induction step we additionally have to consider the cases where $r \in \{Para, Func', Equiv'\}$, which are analogous to the cases where $r \in \{Prim\} \cup \mathcal{CNF}$. □

Lemma 6.8. *Let Φ be a set of Σ-sentences and Φ_{cl} be the corresponding set of pre-clauses. Furthermore let $\mathbf{A}, \mathbf{B}$ be formulae and $\mathcal{C}, \mathcal{D}$ be clauses. We have that:*

1. *If $\Phi_{cl} * [\mathbf{A}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$ and $\Phi_{cl} * [\mathbf{B}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$, then $\Phi_{cl} * [\mathbf{A} \vee \mathbf{B}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$.*

2. *If $\Phi_{cl} * [\mathbf{A}]^T * [\mathbf{B}]^F \vdash_{\mathcal{ERUE}_{fc}} \Box$ and $\Phi_{cl} * [\mathbf{A}]^F * [\mathbf{B}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$, then $\Phi_{cl} * [\mathbf{A} \Leftrightarrow \mathbf{B}]^F$.*

Proof: Analogous to lemma 4.15 and 5.16. The new or slightly modified rules do not affect the argumentations. □

As in Chapter 5, we will now show that the generalised rules *GFac*, *GPrim*, $GRes_1$, $GRes_2$ and $GRes_3$ are in a weak sense (modulo subsequent clause normalisation and a kind of lifting) derivable in calculus $\mathcal{ERUE}_{fc}$. Derivability of $GRes_1$ will then be employed to establish the admissibility of generalised resolution rule *GPara* for calculus $\mathcal{ERUE}_{fc}$. We will finally use the latter result in the main completeness proof for calculus $\mathcal{ERUE}_{fc}$ to prove the crucial substitutivity property $\nabla^s_{\mathfrak{e}}$ of primitive equality.

Lemma 6.9 (Weak Derivability of *GFac*). *Let $\mathcal{C}_1,\mathcal{C}_2$ be clauses and $\mathcal{D}$ be a proper clause. If $\Delta_1 : \mathcal{C}_1 \vdash^{GFac} \mathcal{C}_2 \vdash_{\mathcal{CNF}} \mathcal{D}$, then there is a derivation $\Delta_2 : \mathcal{C}_1 \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D}$.*

Proof: Analagous to the corresponding Lemma 5.22 for $\mathcal{EP}_{fc}$; resolution and factorisation on unification constraints does not cause serious new problems. □

Lemma 6.10 (Weak Derivability of *GPrim*). *Let $\mathcal{C}_1,\mathcal{C}_2$ be clauses and $\mathcal{D}$ be a proper clause. If $\Delta_1 : \mathcal{C}_1 \vdash^{GPrim} \mathcal{C}_2 \vdash_{\mathcal{CNF}} \mathcal{D}$, then there exists a substitution σ and a derivation $\Delta_2 : \mathcal{C}_1 \vdash_{\mathcal{ERUE}_{fc}} \mathcal{E}$, such that $\mathcal{E}_\sigma \equiv_\alpha \mathcal{D}$.*

Proof: Analagous to the corresponding Lemma 5.23 for $\mathcal{EP}_{fc}$; resolution and factorisation on unification constraints causes no extra problem. □

Lemma 6.11 (Weak Derivability of $GRes_1$, $GRes_2$, and $GRes_3$).

1. *Let $\mathcal{C}_1,\mathcal{C}_2,\mathcal{C}_3$ be clauses and $\mathcal{C}_4$ be a proper clause, such that $\Delta_1 : \{\mathcal{C}_1,\mathcal{C}_2\} \vdash^r \mathcal{C}_3 \vdash_{\{GFac\}} \mathcal{C}_4 \vdash_{\mathcal{CNF}} \mathcal{D}$ for $r \in \{GRes_1, GRes_2, GRes_3\}$. If both resolution literals of the generalised resolution step are proper, then there exists $\Delta_2 : \{\mathcal{C}_1,\mathcal{C}_2\} \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D}$.*
2. *Let $\mathcal{C}_1,\mathcal{C}_2,\mathcal{C}_3,\mathcal{C}_4$ be clauses and $\mathcal{D}$ be a proper clause, such that $\Delta_1 : \{\mathcal{C}_1,\mathcal{C}_2\} \vdash^r \mathcal{C}_3 \vdash_{\{GFac\}} \mathcal{C}_4 \vdash_{\mathcal{CNF}} \mathcal{D}$ for $r \in \{GRes_1, GRes_2, GRes_3\}$. Then there exists a substitution σ and a derivation $\Delta_2 : \{\mathcal{C}_1,\mathcal{C}_2\} \vdash_{\mathcal{ERUE}_{fc}} \mathcal{E}$, such that $\mathcal{E}_\sigma \equiv_\alpha \mathcal{D}$.*

Proof: The argument is analogous to 5.24 and resolution on unification constraints does cause new problems. □

The following paramodulation Lemma 6.12 now shows that the generalised paramodulation rule *GPara* is admissible in $\mathcal{ERUE}_{fc}$.

Whereas Lemma 6.12(1) is again analogous to Lemma 5.25(1), we can prove in 6.12(2) only admissibility instead of the weak derivability property in 5.25(2). This is because in 5.25 we were able to reduce the applications of the generalised paramodulation rule either to the generalised resolution rules and thus finally to the proper resolution rule *Res* or to the proper paramodulation rule *Para*. Whereas the reduction to rule *Res* is analogous here as well, we cannot employ the reduction to the proper paramodulation *Para*. Instead we have to show that alternative reductions are possible which employ the RUE-resolution idea to resolve against unification constraints. The latter causes the loss of the weak derivability property. But fortunately admissibility is sufficient for our purposes.

Lemma 6.12 (Admissibility of Generalised Paramodulation Rule).

1. *Let $\mathcal{C}_1$ be a clause and $\mathcal{C}_2$ be a proper clause.*

(a) *If* $\mathcal{C}_1 : [\mathbf{A}]^\alpha \vee D$ *and* $\mathcal{C}_2 : [\mathbf{A} = \mathbf{B}]^T$*, then there exists* $\Delta_1 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D} : [\mathbf{C}]^\alpha \vee E$*, and a substitution* σ*, such that* $\mathcal{D}_\sigma \equiv_\alpha [\mathbf{B}]^\alpha \vee D$ *(i.e.,* $\mathcal{D}$ *is more general than the clause we would obtain by a respective generalised paramodulation step)*

(b) *If* $\mathcal{C}_1 : [\mathbf{A}_{\overline{\gamma} \to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha \vee D$ *and* $\mathcal{C}_2 : [(\mathbf{A}_{\overline{\gamma} \to o}\ \overline{X^n_\gamma}) = (\mathbf{B}_{\overline{\gamma} \to o}\ \overline{X^n_\gamma})]^T$ *where* $\overline{X^n_\gamma} \notin$ *free*$(\overline{\mathbf{T}^n_\gamma}) \cup$ *free*$(\mathbf{A})$*, then there exists* $\Delta_2 : \{\mathcal{C}_1, \mathcal{C}_2\} \vdash_{\mathcal{ERUE}_{fc}} \mathcal{D} : [\mathbf{C}_{\overline{\gamma} \to o}\ \overline{\mathbf{D}^n_\gamma}]^\alpha \vee E'$ *and a substitution* σ*, such that* $\mathcal{D}_\sigma \equiv_\alpha [\mathbf{B}_{\overline{\gamma} \to o}\ \overline{\mathbf{T}^n_\gamma}]^\alpha \vee D$.

2. *Let* $\mathcal{C}_1[A/B]_\mathcal{P}$ *be a clause that is obtained from clause* $\mathcal{C}$ *be replacing the occurrences of term* $\mathbf{B}$ *at positions* $p \in \mathcal{P}$ *by term* $\mathbf{A}$ *and let* Φ *be a set of clauses. If* $\Delta_1 : \Phi * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_\mathcal{P} * [\mathbf{A} = \mathbf{B}]^T \vdash^{GPara} \Phi * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_\mathcal{P} * [\mathbf{A} = \mathbf{B}]^T * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}} \vdash_{\mathcal{ERUE}_{fc}} \Box$*, then there is a derivation* $\Delta_2 : \Phi * [\mathbf{A} = \mathbf{B}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$.

Proof:

(1) For the cases (1a) and (1b) the proof is analogous to the corresponding argumentation in lemma 5.25(1). The only difference is that we here employ Lemma 6.11 instead of 5.24.
(2) The proof is by induction on the length of Δ_1 and the base step is the most complicate one as we cannot reduce *GPara* paramodulation steps to *Para* steps, but instead have to replace them by pure $\mathcal{ERUE}_{fc}$ derivations.

$\underline{n \equiv 1:}$ If $\Box \in \Phi * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_\mathcal{P} * [\mathbf{A} = \mathbf{B}]^T$ the assertion follows trivially. Therefore let us assume that $\Box \equiv \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}}$. Hence $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_\mathcal{P}$ has form $\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} \vee [\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}]^\alpha$, where all but one literal must be flex-flex unification constraints. Without loss of generality we assume that this is the literal $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}]^\alpha$. Then the new clause $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}}$ looks like $\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} \vee [\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2\setminus\{p'\}}]^\alpha$, where position p' in literal $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2\setminus\{p'\}}]^\alpha$ specifies the position where the replacement has been taken place. Obviously this clause must consist only of flex-flex unification constraints, and hence, polarity α must be F, as otherwise it would be different from $\Box$. Thus, the literal $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2\setminus\{p'\}}]^\alpha$ is a flex-flex constraint. We now consider the position p' in literal $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}]^F$, where the generalised paramodulation step has been applied to. The first two possibilities concentrate on replacements that include the head position in this literal.

- If p' specifies the replacement of the whole atom of the literal in focus, i.e., $\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2} \equiv \mathbf{A}$, then we get the assertion by (1a).
- If p' specifies the replacement of a proper prefix term of the atom $\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}$ (i.e., $\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}$ looks like $(\mathbf{A}\ \overline{\mathbf{U}^n})$), then we first apply the positive functional extensionality rule *Func'* for n-times to $[\mathbf{A} = \mathbf{B}]^T$, thereby generating a proper clause $[\mathbf{A}\ \overline{X^n} = \mathbf{B}\ \overline{X^n}]^T$. Now (1b) is applicable, which gives us the assertion.

Next we examine the cases where a proper subterm of the atom of $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}]^F$ is replaced. As the replacement must lead to a flex-flex unification constraint we already know that $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}]^F$ must be a unification constraint and thus must have form $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_3} = \mathbf{T}_2[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_4}]^F$.

- If p' refers to proper subterm of either $\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_3}$ or $\mathbf{T}_2[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_4}$, then it must be the case that the generalised paramodulation step was not necessary as the literal $[\mathbf{T}[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_2}]^F$ already is a flex-flex unification constraint and thus $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_\mathcal{P} \equiv \Box$, which trivially gives us the assertion.

- Without loss of generality let us assume that p' refers to a prefix of term $\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_3}$. We then know that $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_3} = \mathbf{T}_2[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_4}]^F$ must have form $[\mathbf{A}\ \overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})} = H\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}}]^F$, where $n, m \geq 0$. As the replacement of $\mathbf{A}$ by $\mathbf{B}$ introduces a flex-flex constraint, we know that term $\mathbf{B}$ must have a flexible head, i.e., $\mathbf{B}$ has form $(F\ \overline{\mathbf{R}^l})$, such that $l \geq 0$. Now consider term $\mathbf{A}$: If the head of $\mathbf{A}$ is a variable then again the replacement of $\mathbf{A}$ is not necessary as we already have a flex-flex constraint without employing generalised paramodulation rule. If on the other hand $\mathbf{A}$ has a rigid head, i.e., $\mathbf{A}$ has form $(a\ \overline{\mathbf{W}^k})$ for $k \geq 0$, then the proof is a bit more complicate as we have to construct a refutation using $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}}$ and $[\mathbf{A} = \mathbf{B}]^T$ without employing generalised paramodulation. Before discussing this derivation let us sum up the structural restrictions that are given: The clause $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}}$ looks like $\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} \vee [((a\ \overline{\mathbf{W}^k})\ \overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})}) = (H\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}})]^F$ and clause $[\mathbf{A} = \mathbf{B}]^T$ has form $[(a\ \overline{\mathbf{W}^k}) =^{\gamma_3 \to o} (F\ \overline{\mathbf{R}^l})]^T$, where a is a predicate constant and F a predicate variable of appropriate type. We apply rule *FlexRigid* to the former clause and positive extensionality rule *Func'* for n-times to the latter clause, such that we get:

$$\begin{aligned}\mathcal{C}_2 : \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} &\vee [((a\ \overline{\mathbf{W}^k})\ \overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})}) =^o (H\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}})]^F \\ &\vee [H = \lambda \overline{Z^m}.\ a\ \overline{(K\ \overline{Z^m})^{k+n}}]^F \\ \mathcal{C}_3 : [(a\ \overline{\mathbf{W}^k})\ \overline{Y^n} &=^o (F\ \overline{\mathbf{R}^l})\ \overline{Y^n}]^T\end{aligned}$$

The variable K in $\mathcal{C}_2$ and the variables $\overline{Y^n}$ in $\mathcal{C}_3$ are new free variables. Next, we instantiate the imitation binding in clause $\mathcal{C}_2$ with rule *Subst*. (Thereby we assume that variable H does not occur in any of the literals in $\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1}$. If on the other hand H occurs in $\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1}$, then we only get a problem if H occurs at literal head positions. In these cases we can use an analogous argumentation to the following one for these literals as well.)

$$\mathcal{C}_4 : \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} \vee [(a\ \overline{\mathbf{W}^k})\ \overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})} =^o (a\overline{(K\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}})^{k+n}})]^F$$

By resolving[2] between $\mathcal{C}_4$ and $\mathcal{C}_3$ we get (note that the unification terms of both unification constraints must have the same type)

$$\begin{aligned}\mathcal{C}_5 : \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} \vee [((a\ \overline{\mathbf{W}^k})\ \overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})} &=^o (a\overline{(K\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}})^{k+n}})) \\ =^o ((a\ \overline{\mathbf{W}^k})\ \overline{Y^n} &=^o (F\ \overline{\mathbf{R}^l})\ \overline{Y^n})]^F\end{aligned}$$

We now apply the decomposition rule (2 times) and immediately delete the trivial pair $[=\,=\,=]^F$ with rule *Triv.*

$$\begin{aligned}\mathcal{C}_5 : \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1} &\vee [((a\ \overline{\mathbf{W}^k})\ \overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})}) = ((a\ \overline{\mathbf{W}^k})\ \overline{Y^n})]^F \\ &\vee [(\overline{a(K\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}})^{k+n}}) = ((F\ \overline{\mathbf{R}^l})\ \overline{Y^n})]^F\end{aligned}$$

We know that $\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}_1}$ contains only *flex-flex*-constraints and that F and $\overline{Y^n}$ do not occur in them. And as the latter two unification constraints of $\mathcal{C}_5$ are obviously solvable with substitution

$$[\overline{(\mathbf{U}[\mathbf{A}/\mathbf{B}]^n_{\mathcal{P}_{4_n}})}/\overline{Y^n}, \lambda \overline{Z^{l+n}}.\ (a\overline{(K\ \overline{\mathbf{V}[\mathbf{A}/\mathbf{B}]^m_{\mathcal{P}_{5_m}}})^{k+n}})/F]$$

[2] We do not need to switch the latter unification constraint here as $\mathcal{C}_4$ and $\mathcal{C}_3$ already have the right constellation. But generally it might be necessary to employ the symmetry rule to clause $\mathcal{C}_3$.

we know that $\mathcal{C}_5 \vdash_{\mathcal{UNI}_f} \Box$, which proves the assertion.

$\underline{n > 1:}$ In the step case we consider the second derivation step in $\Delta_1 : \Phi * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}} * [\mathbf{A} = \mathbf{B}]^T \vdash^{GPara} \Phi * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}} * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}} \vdash^{r} \Phi * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}} * \mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}} * \mathcal{C}_2 \vdash_{\mathcal{ERUE}_{fc}} \Box$, i.e., we focus on the first step following the generalised paramodulation step. If the premise clause(s) within the application of rule r are different from $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}}$ we can obviously switch the first two derivation steps of Δ_1, such that we get the assertion trivially by employing the induction hypothesis.[3] This even holds if the application of rule r indeed uses $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}}$ but operates on a literal that was not affected by the initial generalised paramodulation step, i.e., on a literal different from the position where p refers to. Thus, for all of the following cases let us assume that clause $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}\setminus\{p\}}$ is of form $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^{\alpha} \vee \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''}$ (for respective position lists P', P'' and position p', i.e, p' and p are equal is one removes the first element from position p, specifying the subterm that was modified within the initial generalised paramodulation step) and that r operates in the second derivation step on literal $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^{\alpha}$. Note that under this assumption the non-modified original clause $\mathcal{C}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}}$ is of form $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'}]^{\alpha} \vee \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''}$.

$r \in \{Res, Fac\}$: As both rules do not depend on the term structure of the resolution or factorisation literals, we can in both cases switch the initial derivation steps, such that the assertion easily follows by induction hypothesis. We will briefly illustrate this here for rule *Res*; the argumentation for *Fac* is analogous. Let us assume that there is a clause $\mathcal{C}_2 \in \Phi$ that is of form $[\mathbf{T}_2]^{\beta} \vee C'_2$ and that

$$\frac{[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^{\alpha} \vee \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''} \quad [\mathbf{T}_2]^{\beta} \vee C'_2}{\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''} \vee C'_2 \vee [(\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}) = \mathbf{T}_2]^F}\; Res$$

Then an analogous resolution proof step is possible between $\mathcal{C}_2$ and the non-modified clause $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'}]^{\alpha} \vee \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''}$

$$\frac{[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'}]^{\alpha} \vee \mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''} \quad [\mathbf{T}_2]^{\beta} \vee C'_2}{\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''} \vee C'_2 \vee [(\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'}) = \mathbf{T}_2]^F}\; Res$$

We can obviously apply generalised paramodulation rule *GPara* to the latter clause (at position p' in the left term of the new unification constraint) thereby generating exactly the same clause as in the above original derivation.

$$\frac{\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''} \vee C'_2 \vee [\mathbf{T}_2 = (\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'})]^F \quad [\mathbf{A} = \mathbf{B}]^T}{\mathcal{C}'_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}''} \vee C'_2 \vee [(\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}) = \mathbf{T}_2]^F}\; GPara$$

Now the assertion follows by induction hypothesis.

$r \in \{Prim, Func', Equiv'\}$: The three cases are similar and therefore we only discuss rule *Prim* here. Thereby we consider the position p' of the subterm that has been rewritten in the generalised paramodulation step. (i) If the position p' in literal $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^{\alpha}$ refers to a proper subterm, then we get the assertion analogously to the cases for $r \in \{Res, Fac\}$ above by switching the first two derivation steps

[3] Note that we here need to employ the conventions of Remark 4.19.

and employing the induction hypothesis. (ii) In the other case position p' refers to a flexible prefix term of $\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}$, i.e., this literal has form

$$[(\ldots\overbrace{(\ldots(H\ \mathbf{U}_1)\ldots\mathbf{U}_k)}^{\mathbf{A}}\ldots\mathbf{U}_n)]^\alpha \qquad (*)$$

for $0 \leq k \leq n$. In case $k \equiv n$ we can replace the initial paramodulation step in Δ_1 by an alternative one not employing generalised paramodulation by (1a) and get the assertion by induction hypothesis. In case $k \leq n$ we first apply positive extensionality rule *Func'* for $(k-n)$-times to the clause $[\mathbf{A} = \mathbf{B}]^T$ leading to $[\mathbf{A}\ \overline{Y^{k-n}} = \mathbf{B}\ \overline{Y^{k-n}}]^T$. This time we can employ (1b) to replace the initial paramodulation step in Δ_1 by an alternative one. Again, the assertion follows by induction hypothesis.

$r \in \mathcal{UNI}_f$ If position p' in literal $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^\alpha$ refers to a flexible prefix term as illustrated in $(*)$ above, we get the assertion analogously to the previous case by (1a) and induction hypothesis, or by (1b) in combination with an appropriate modification of clause $[\mathbf{A} = \mathbf{B}]^T$ with rule *Func'*. If on the other hand position p refers to a proper subterm of $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^\alpha$, then we get the result by employing derivation that is analogous to one already employed in Remark 5.2 (which shows that paramodulation into unification constraints is derivable).

$r \equiv Cnf$: We again differentiate between the following two cases: (i) position p' in $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^\alpha$ refers to proper subterm of the literals atom and (ii) position p' in $[\mathbf{T}_1[\mathbf{A}/\mathbf{B}]_{\mathcal{P}'\setminus\{p'\}}]^\alpha$ refers to a prefix term of the atom (like in $(*)$ above). In case (i) the assertion follows by induction hypothesis after switching the first two derivation steps.[4]. This is possible as derivation step r is obviously also applicable to the initial clause first, such that an subsequent application of rule *Para* leads to $\mathcal{C}_2$. In case (ii) the argumentation is analogous to the case (ii) for $r \in \{Prim, Func', Equiv'\}$ discussed above, i.e., we get the assertion by employing either (1a) and induction hypothesis or by (1b) in combination with an n-times application of rule *Func'* to literal $[\mathbf{A} = \mathbf{B}]^T$ and induction hypothesis.

□

Theorem 6.13 (Completeness of $\mathcal{ERUE}_{fc}$). *The calculus $\mathcal{ERUE}_{fc}$ is complete with respect to Henkin models.*

Proof: Let Γ_Σ be the set of Σ-sentences which cannot be refuted by the calculus $\mathcal{ERUE}_{fc}$ ($\Gamma_\Sigma := \{\Phi \subseteq cwff_o(\Sigma) | \Phi_{cl} \not\vdash_{\mathcal{ERUE}_{fc}} \Box\}$), then we show that Γ_Σ is a saturated abstract consistency class for Henkin models with primitive equality (cf. Def. 3.18). This entails completeness with the model existence theorem for Henkin models with primitive equality (cf. Theorem 3.29).

First we have to verify that Γ_Σ validates the abstract consistency properties ∇_c, $\nabla_\neg$, ∇_β, $\nabla_\vee$, $\nabla_\wedge$, $\nabla_\forall$, $\nabla_\exists$, $\nabla_\mathfrak{b}$, $\nabla_\mathfrak{q}$ and that Γ_Σ is saturated. For all of these cases the proofs are identical to the corresponding argumentations in theorem 4.16. The only difference is that we employ the lemmata 6.6 and 6.8 instead of 4.12 and 4.15. Thus, all we need to ensure is the validity of the additional abstract consistency properties $\nabla_\mathfrak{e}^r$ and $\nabla_\mathfrak{e}^s$ for primitive equality.

[4] Note that we here need to employ the conventions of Remark 4.19

$\nabla_{\mathfrak{e}}^{r}$ $\neg(\mathbf{A} =^{\alpha} \mathbf{A}) \notin \Phi$.

$\nabla_{\mathfrak{e}}^{s}$ If $\mathbf{F}[\mathbf{A}]_p \in \Phi$ and $\mathbf{A} = \mathbf{B} \in \Phi$, then $\Phi * \mathbf{F}[\mathbf{B}]_p \in \Gamma_\Sigma$.

($\nabla_{\mathfrak{e}}^{r}$) We have that $[\mathbf{A} =^{\alpha} \mathbf{A}]^F \vdash^{Triv} \Box$, such that $\neg(\mathbf{A} =^{\alpha} \mathbf{A}) \notin \Phi$.
($\nabla_{\mathfrak{e}}^{s}$) Analogously to 5.27 we show the contrapositive of the assertion. Therefor we assume, that there is derivation $\Delta_1 : \Phi_{cl} * [\mathbf{F}[\mathbf{B}]_p]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$, and show, that there exists $\Delta_2 : \Phi_{cl} * [\mathbf{F}[\mathbf{A}]_p]^T * [\mathbf{A} = \mathbf{B}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$. Consider the following $\mathcal{ERUE}_{fc}$-derivation:

$$\frac{[\mathbf{F}[\mathbf{A}]_p]^T \quad [\mathbf{A} = \mathbf{B}]^T}{[\mathbf{F}[\mathbf{B}]_p]^T}\ GPara$$

By lemma 6.12(2) we know that the generalised paramodulation rule *GPara* is admissible for calculus $\mathcal{ERUE}_{fc}$, such that must be a pure $\mathcal{ERUE}_{fc}$ derivation $\Delta_2 : \Phi_{cl} * [\mathbf{F}[\mathbf{A}]_p]^T * [\mathbf{A} = \mathbf{B}]^T \vdash_{\mathcal{ERUE}_{fc}} \Box$ that avoids rule *GPara*. □

6.5 Theorem Equivalence

We now prove that $\mathcal{ERUE}_{fc}$ and $\mathcal{ERUE}_{f}$ are theorem equivalent. Theorem equivalence of $\mathcal{ERUE}_{fc}$ and $\mathcal{ERUE}$ is then only presented as a conjecture.

Lemma 6.14 (Proper Clauses in $\mathcal{ERUE}$ and $\mathcal{ERUE}_f$). *For each non-proper clause $\mathcal{C}$ and clause set Φ, such that $\Phi * \vdash_{\mathcal{ERUE}_{fc}} \mathcal{C}$, we have that $\Phi \vdash_{\mathcal{ERUE}_f} \mathcal{C}$.*

Proof: Analogously to lemmata 4.20 and 5.28. Resolution on unification constraints does not cause additional problems. □

As □ is also a proper clause we immediately get the following corollary:

Corollary 6.15 (Theorem Equivalence of $\mathcal{ERUE}_{fc}$ and $\mathcal{ERUE}_f$). *The calculi $\mathcal{ERUE}_{fc}$ and $\mathcal{ERUE}_f$ are theorem equivalent.*

Conjecture 6.16 (Theorem Equivalence of $\mathcal{ERUE}$ and $\mathcal{ERUE}_{fc}$ (or $\mathcal{ERUE}_f$)). *The calculi $\mathcal{ERUE}$ and $\mathcal{ERUE}_{fc}$ (or $\mathcal{ERUE}_f$) are theorem equivalent.*

Again the author expects that the proof for this conjecture will be analogous to the proof of the theorem equivalence for calculi $\mathcal{ER}_{fc}$ and $\mathcal{ER}$ (or $\mathcal{EP}_{fc}$ and $\mathcal{EP}$). This conjecture gains some evidence by the examples carried out with the Leo-prover [BK98b] for extensional higher-order resolution as well as the new challenging examples discussed for extensional higher-order paramodulation in Chapter 8. There is no example known to the author that demonstrates the necessity of rule *FlexFlex*.

Chapter 7

The LEO-System

In this chapter we present the theorem prover LEO, which realises the extensional higher-order resolution approach $\mathcal{ER}$. LEO is implemented in CLOS [Ste90], a object-oriented extension of LISP, and its input language is $\mathcal{POST}$, which is also used in the ΩMEGA-system [BCF+97].

Despite the theoretical completeness results on $\mathcal{ER}$, the most recent implementation (LEO03) of LEO is not Henkin complete. The main lesson learned from the implementation and the case studies is that developing a theoretically Henkin complete approach to mechanise classical type theory is one thing, but implementing it, such that non-trivial theorems can be proven in practice, is certainly a much more challenging task.

The difference between unification and proof search disappears in extensional higher-order theorem proving and all rules have to be integrated at the same level in order to realise arbitrary calls from the overall proof search to unification and from unification again to the overall proof search. Implementing calculus $\mathcal{ER}$ in such a theoretically Henkin complete but practically naive way is certainly not very complicated, however, the search spaces generated in such a system would be enormous. Therefore LEO employs the extensionality rules only in a restricted way and furthermore guides their application by additional heuristics. This enables LEO to prove simple theorems which require the application of the extensionality principles, such as the theorem discussed in Subsection 7.2.4, which is currently not automatically provable in any other higher-order theorem prover. The drawback is, that these restrictions and heuristics are the main sources of LEO's incompleteness.

LEO adapts well known ideas and techniques from first-order automated theorem proving and modifies them with respect to the special requirements and aspects of the extensional higher-order resolution approach. More precisely, LEO is based on an extended SOS-architecture that provides two additional stores. The two new constructions are the store of extensionally interesting clauses and the store of unification continuations; the ideas of these concepts will be illustrated in detail below.

The most important feature of the LEO-system is that it employs, like traditional first-order resolution, eager unification as a kind of filter, i.e., LEO tries to get rid of newly generated clauses with a non-solvable unification constraint as early as possible. The main problem thereby is the undecidability of higher-order (pre-)unification and the general need for recursive calls from unification to the general refutation process in order to unify terms also with respect to the extensionality principles. A compromise between completeness and practicability is needed for LEO in order to realise the eager unification idea and hence LEO should be viewed as higher-order theorem prover specialised in extensionality reasoning rather than a general purpose automated theorem-prover for classical type theory.

The LEO-system has been implemented by the author mainly during a stay at Carnegie Mellon University (see [Ben97]). The implementation is still quite prototypical and leaves much room for improvements.

LEO's main strength is not to compete with other theorem-provers, such as TPS [ABI$^+$96, AINP90], but rather more to complement their weakness in the treatment of the extensionality principles. TPS, for instance, can in contrasts to LEO explore rather deep search spaces and has its strengths, e.g., in the handling of the primitive substitution principles or the handling of definitions [BA98]. *Real mathematics* will only be mechanisable by combining specialised automated higher-order and first-order theorem provers guided and controlled, e.g., by a cognitively more adequate proof planning approach (see [Bun88, CrS98, Mel95, Mel94]) or an intelligent agent mechanism like [BS99, BS98a].

We shall now discuss LEO's extended SOS-architecture and its main loop in a greater detail. A case study will then be sketched and typical examples that can be solved with the LEO system will be given.

7.1 Basic Data-Structures and Algorithms

KEIM-Toolbox The implementation of LEO employs the KEIM-toolbox [HKK$^+$94] for deduction systems. This toolbox, which also underlies the mathematical assistant system ΩMEGA [BCF$^+$97], provides many useful data structures (e.g., higher-order terms, literals, clauses, and substitutions) and basic algorithms (e.g., application of substitution, subterm replacement, copying, and renaming). Thus, the usage of KEIM allows for a rather quick implementation of new higher- or first-order theorem proving systems. In addition to the code provided by the KEIM-package, LEO consists of 6900 lines of LISP code. Whereas the KEIM-toolbox indeed turned out to be very useful to support a prototypical development of a system like LEO, the experience with LEO also showed, that the datastructures offered by KEIM are in many cases too extensive and costly. Especially the implementation of LEO's latest version, LEO3, which employs the fully polymorphic data structures offered by KEIM3 (the latest version of KEIM) is roughly 5-10 times slower than the former version LEO1, which was based on the non-polymorphic data structures of KEIM2.

Higher-Order Unification The author (in cooperation with Karsten Konrad) implemented higher-order unification by rather closely realising the rules of [SG89, Sny91]. In all branching cases (branching may only happen in the *FlexRigid* or *FlexFlex* case) this algorithm employs the variable binding mechanism offered by the KEIM-package. This mechanism avoids copying of terms and maintains variable instantiations by pointers instead of explicitly instantiating, i.e., modifying, the term data structures. Unfortunately the current implementation does not allow for an explicit maintenance of the variable bindings itself, such that a heuristically guided higher-order unification algorithm cannot be realised in this setting (the aim was to realise a most general, heuristically guided backtracking in the unification search tree, where an user definable heuristics selects — probably in dependence of the current proof goal — the particular partial binding to be examined first when branching in the *FlexRigid*-case). As KEIM is written in Lisp, the implemented unification algorithm can furthermore not exploit cuncurrency like, e.g., employed in the unification algorithm of LEO's brother HOT [Kon98], which is written in the programming language OZ [Sb98, SSW94]. HOT translates and realises the main ideas of calculus $\mathcal{ER}$ in a tableaux setting.

Higher-Order Term indexing In most state of the art theorem provers for first-order logic, term indexing techniques [Gra95] are successfully employed as strong filters in unification, subsumption, and the computation of resolution partners. The basis for higher-order term indexing has been laid by the masters thesis of Lars Klein [Kle97], supervised by Michael Kohlhase, which adapts first-order term indexing ideas as presented in [Gra95] to λ-terms. The key idea of this adaptation is to employ the simplification part of higher-order unification, which is in contrast to general higher-order unification decidable, as the overall filtering criterion. When sending a request for a given λ-term $\mathbf{T}$ to this higher-order term indexing approach, it filters out those terms $\mathbf{T}'$, given in its database, for which pure simplification can not detect a counterargument to its unifiability with $\mathbf{T}$. This obviously realises an imperfect filter, as each delivered $\mathbf{T}'$ may still fail to unify with $\mathbf{T}$. But because of the undecidability of general higher-order unification it is easy to see that a perfect eager unification filter cannot be achieved.

Subsumption The subsumption filter [Tam98, BHJB92, Fit96], which is well known from resolution based first-order theorem proving, checks whether a given clause $\mathcal{C} : N_1 \vee \ldots \vee N_n$ is more general than another clause $\mathcal{D} : M_1 \vee \ldots \vee M_m$. To be more precise, subsumption checks whether there exists a substitution σ, such that the particular literals of $\mathcal{C}\sigma$ are entailed in $\mathcal{D}$. If so, then the more specialised clause $\mathcal{D}$ can be removed from the search space without affecting completeness. In practice, many resolution based first-order theorem prover spend most of their computation time in filtering out subsumed clauses in order to minimise the search space. As matching is involved, term indexing techniques are thereby employed to lower the computation costs and to speed up the subsumption tests.

LEO employs a higher-order subsumption test that is, apart from the technical details, very similar to the ones employed in first-order provers. Instead of first-order matching, the criteria for comparing the single literals is higher-order simplification matching, i.e., matching with respect to the deterministic higher-order simplification rules. As matching is known to be decidable up to fourth-order logic [Pad95], it is theoretically possible to develop and employ a much stronger subsumption filter in LEO. This has been avoided so far, as the respective matching algorithms seem to be rather complex. But at least higher-order pattern unification or matching [Mil91], which is decidable in linear time [Qia93], should be employed in LEO. Consequently, the higher-order subsumption filter that is currently employed in LEO is — just as the higher-order unification filter — a quite imperfect one, as not all subsumed clauses can be determined in the search space.

But note that even if higher-order matching would be generally decidable for all orders (which is still an open and challenging question), we still could not develop a perfect higher-order subsumption filter for LEO, as this would require a higher-order extensional subsumption approach instead of one that is based solely on syntactical higher-order matching. The problem of extensional higher-order subsumption will be illustrated in detail in Section 7.3.

7.2 Extended SOS Architecture and Main Loop

7.2.1 Problems

The most important aspects and problems for an implementation of the LEO-system are:

1. **Extensional higher-order (pre-)unification** The necessary combination of the traditional syntactical higher-order unification rules with the new extensionality rules[1] *Equiv* and *Leib* is probably the most challenging aspect in the implementation of $\mathcal{ER}$, as a non-restricted application of this rules obviously leads to a search space explosion. But because of the goal directed application of the extensionality principles in $\mathcal{ER}$, even such a general integration would still have advantages over an unrestricted usage of the infinetely many extensionality rules in the traditional constrained resolution approach (cf. the discussion in Section 1.4). Anyhow, the challenge in LEO is to find suitable heuristics to restrict the application of the extensionality rules in calculus $\mathcal{ER}$ and to avoid too many recursive calls from within higher-order unification to the overal search process.

2. **Eager unification is essential but undecidable** In contrast to Huet's original constraint resolution approach, eager unification becomes essential in $\mathcal{ER}$ and cannot be generally delayed. The reason is, that whenever both extensionality principles are involved in the search for a proof, then calculus $\mathcal{ER}$ needs to employ recursive calls to the overall search procedure from within a unification attempt, i.e., eager unification is essential in $\mathcal{ER}$ and LEO. Therefore we need to develop a mechanism, which allows to interrupt eager unificatioon attempts. For example, when reaching a user-definable unification search depth limit the unification process terminates and is resumed later.

3. **Primitive substitution is infinitely branching** The primitive substitution problem is well known in resolution based higher-order theorem proving and is thus not a specific problem of calculus $\mathcal{ER}$. Informally the problem is that one generally has to ensure that free predicate variables can be instantiated with arbitrary formulae. In order to illustrate the problem, assume that the single unit clause $[P\ a]^F$ (e.g., obtained from the negation and normalisation of the theorem $\exists P_{\iota\to o}.\ P\ a_\iota$) is given in the search space, where $P_{\iota\to o}$ is a free variable. The only way to refute this unit clause is to guess the *right* instantiation for P; in our case $\lambda X_\iota.\ \neg P'\ X$. With this instantiation we get another unit clause $[\neg(P'a)]^F$, which normalises to $[P'a]^T$. Now the empty clause can easily be derived by resolving between $[P\ a]^F$ and $[P'a]^T$.

   Fortunately, a blind guess of instantiating formulae can be avoided as it is sufficent to subsequently instantiante the free variable heads with partial bindings imitating the logical connectives in the signature (which in some sense subsequently enumerates all formulas schemes). But as experienced in the TPS-project (cf. [ABI+96]) it can nevertheless be very fruitful for some proof attempts to instantiate free variables immediately with special formulae, such as Leibniz Equality.

   It is easy to see that the primitive substitution principle causes an enormous explosion of the search space when applied in an unrestricted and non-guided way.

Apart from these challenging aspects of the implementation of LEO, there are many other, mostly technical problems (see [Ben97]).

7.2.2 Extended SOS Architecture

LEO's basic architecture adapts the well known set of support approach (cf. [McC94]) with respect to the special requirements and challenges of calculus $\mathcal{ER}$. The four cornerstones of

[1] We here consider the modified rule *Func* as belonging to syntactical higher-order unification.

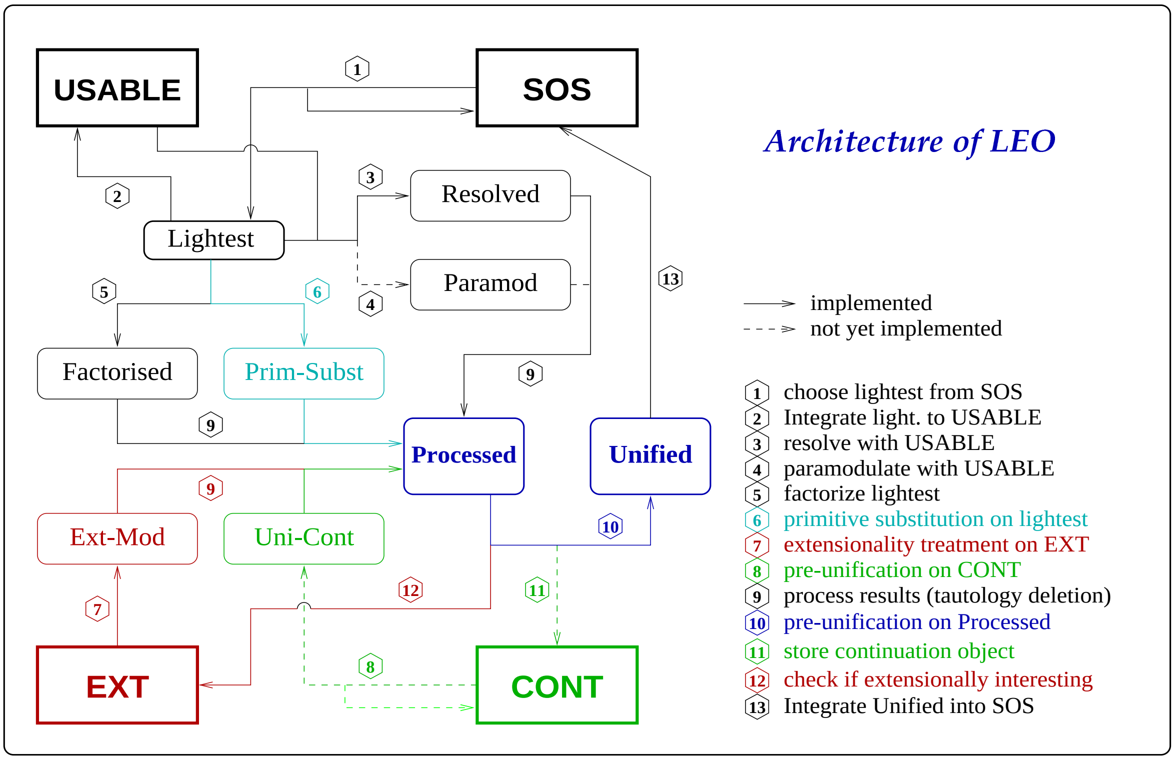

USABLE
SOS
Architecture of LEO
Resolved
Lightest
Paramod
Factorised
Prim-Subst
Processed
Unified
Ext-Mod
Uni-Cont
EXT
CONT
1
2
3
4
5
6
7
8
9
10
11
12
13
implemented
not yet implemented
1 choose lightest from SOS
2 Integrate light. to USABLE
3 resolve with USABLE
4 paramodulate with USABLE
5 factorize lightest
6 primitive substitution on lightest
7 extensionality treatment on EXT
8 pre-unification on CONT
9 process results (tautology deletion)
10 pre-unification on Processed
11 store continuation object
12 check if extensionally interesting
13 Integrate Unified into SOS

Figure 7.1: LEO's main architecture

LEO's architecture (see the figure 7.1) are:

USABLE The set of all usable clauses, which initially only contains satisfiable clauses, i.e., the clauses stemming from the assumptions of the theorem to prove (same idea as in first-order resolution [McC94]).

SOS The set of support, which initially only contains the clauses belonging to the negated theorem (same idea as in first-order resolution [McC94]).

EXT The set of all *extensionally interesting clauses*, i.e., clauses which are assumed to be unifiable, when the new extensionality rules *Equiv* and *Leib* are taken into account (but probably not by pure higher-order (pre-)unification alone). Initially this set is empty.

CONT The set of all continuations created by the higher-order unification algorithm when reaching the search depth limit in a particular branch of the unification search tree. The idea of this objects is to allow for the continuation of interrupted unification attempts at a later time.

7.2.3 LEO's heuristically guided Main Loop

As a suitable realisation of $\mathcal{ER}$ we suggest a computation consisting of the steps 1–13 as described below (the **Initialise** step is applied only at the very beginning of the proof attempt and is not part of the main loop). The main loop, whose dataflow is graphically illustrated in Figure 7.1, is executed until an empty clause, i.e., a clause consisting only of *flexflex*-unification constraints, is detected. The described algorithm is an abstract and slightly idealised presentation of the working principles of the currently actual version (LEO3) of LEO. Furthemore, this description of LEO's working mechanism abstracts from many technical details and mentions only the most important user definable flags and heuristics that influence its problem solving behaviour.

Initialise The proof problem formalised in $\mathcal{POST}$-syntax is read from a file. Then the specified assumptions and the assertion are pre-clausified, i.e., the assumptions become positive unit (pre-)clauses and the assertion becomes a negative unit (pre-) clause. The assumption clauses are passed to `USABLE` and the assertion clause to `SOS`. Thereby the `SOS` store is automatically sorted (with respect to a heuristics H1, that is based on the clauses weights; this weights are computed from the accumulated weights of the term-symbols and from the clause age, i.e., the number of the loop in which the clause has been created).

Step 1 (Choose Lightest) LEO chooses the lightest, i.e., topmost, clause from `SOS`. If this clause is a pre-clause, i.e., not in proper clause normal-form, then LEO applies clause normalisation to it and integrates the resulting proper clauses into `SOS` while employing subsumption (depending on the flag-setting forward and/or backward subsumption; see also step 13). Within the clause normalisation process the positive primitive equations are replaced by respective Leibniz equations. Negative primitive quations are not expanded but immediately encoded as extensional unification constraints. Furthermore, identical literals are automatically factorised.

In case the lightest clause was not proper, LEO chooses the next clause from SOS and proceeds with this as described. Otherwise the lightest clause is passed to the store `Lightest`.

Step 2 (Insert to `USABLE`) LEO inserts the lightest clause into `USABLE` while employing forward and/or backward subsumption depending on the flag-setting.

Step 3 (Resolve) The lightest clause is resolved against all clauses in `USABLE` and the results are stored in `Resolved`.

Step 4 (Paramodulat) Paramodulation is applied between all clauses in `USABLE` and the lightest clause, and the results are stored in `Paramod`. (This step is currently not realised in LEO3)

Step 5 (Factorise) The lightest clause is factorised and the resulting clauses are stored in `Factorised`.

Step 6 (Primitive Substitution) LEO applies the primitive substitution principle to the lightest clause. The particular logical connectives to be imitated in this step are specified by a flag. The resulting clauses are stored in `Prim-Subst`.

Step 7 (Extensionality Treatment) The heuristically sorted (with a heuristics H2) store `EXT` contains extensionally interesting clauses (i.e., clauses with unification constraints that may have additional pre-unifiers, if the extensionality rules are taken into account). LEO chooses the topmost clause and applies the compound extensionality treatment to all extensionally interesting literals of it. Assume the chosen clause has form $\mathcal{C}_1$:

$D \vee [\mathbf{L}_{\alpha\to\beta} = \mathbf{R}_{\alpha\to\beta}]^F$, where the latter literal is extensionally interesting (determined by a heuristics H3), then the compound extensionality treatment proceeds as follows:

1. First rule *Func* is applied exhaustively to $\mathcal{C}_1$. This means that *Func* is subsequently applied until the unification constraint has form $[((\mathbf{L}_{\alpha\to\beta}\ s^1)\ldots s^n) = ((\mathbf{R}_{\alpha\to\beta}\ s^1)\ldots s^n)]^F$, where both hand sides are of primitive type $\tau \in \{\iota, o\}$.
2. If $\tau \equiv o$, then the compound extensionality treatment applies rule *Equiv*, and if $\tau \equiv \iota$, then rule *Leib* is applied.

The resulting clauses are stored in `Ext-Mod`.

Step 8 (Continue Unification) The heuristically sorted (with a heuristics H4) store `CONT` contains continuations of interrupted higher-order pre-unification attempts from the previous loops (cf. step 10). If the actual unification search depth limit (specified by a flag, whose value can be dynamically increased during proof attempts) allows for a deeper search in the current loop, then the additional search for unifiers will be performed. The resulting, instantiated clauses are passed to `Uni-Cont` and the new continuations are sorted (wrt. heuristics H4) and integrated into `CONT`. (This step is currently not realised in LEO3)

Step 9 (Collect Results) In this step LEO collects all clauses, that have been generated within the current loop, from the stores `Resolved`, `Paramod`, `Factorised`, `Prim-Subst`, `Ext-Mod`, and `Uni-Cont`, eliminates obvious tautologies, and stores the remaining clauses in `Processed`.

Step 10 (Pre-Unify) LEO tries to pre-unify the clauses in `Processed`. Thus, it applies the pre-unification rules exhaustively, thereby spanning a unification tree until it reaches the unification search depth limit specified by a special flag. The unification search depth limit specifies how many subsequent *FlexRigid*-branchings may at most occur in each path through the unification search tree. The pre-unified, i.e., instantiated clauses, are passed to `Unified`. The main idea of this step is to filter out all those clauses with syntactically non-solvable unification constraints (modulo the allowed search depth limit). But note that there are exemptions, which are determined in steps 11 and 12. That means, that not all syntactically non-unifiable clauses are removed from the search space as this would, e.g., also remove the extensionally interesting clauses.

Step 11 (Safe Continuations) Each time a pre-unification attempt in step 10 is interrupted by reaching the unfication search depth limit, a respective continuation is created. This object stores the state of the interrupted unification search process, i.e., it contains the particular unification constraints as given at the point of interruption together with the remaining literals of the clause in focus and some information on the interrupted unification process (in principle a continuation is just a new clause with some additional information). Continuations allow to continue the interrupted unification process at any later time. The set of all such continuations is integrated (employing sorting heuristics H4) in the sorted store `CONT`. (This step is not realised in LEO3.)

Step 12 (Safe Extensionally Interesting Clauses) In the pre-unification process in step 10 LEO analyses the unification pairs in focus (with heuristics H3) in order to estimate whether this unification constraint and thus this clause is extensionally interesting, i.e., probably solvable with respect to both extensionality principles. The motivation thereby is obvious: we want to avoid recursive calls to overall search procedure for all generated unification pairs and instead employ such calls only selectively. We present two exemplifying criteria for H3: (i) never apply the extensionality treatment to two constants of type ι that clash (which are quite a lot unification pairs in practice) and (ii) never apply the extensionality treatment to *flex-flex*-constraints.[2] All extensionally interesting clauses are passed to `EXT`, which is heuristically sorted with a heuristics H5. While integrating the clauses to `EXT` forward and/or backward subsumption is applied in order to minimise the number of clauses in this store (cf. the discussion of syntactical and extensional subsumption below).

Step 13 (Integrate to `SOS`) In the last step LEO integrates all pre-unified clauses in `Unified` into the sorted (with heuristics H1) store `SOS`. Forward and/or backward subsumption is employed depending on the flag-setting.

7.2.4 An illustrated Example

Let us illustrate LEO's working with the following example (this example is graphically illustrated in Figure 7.5 in Subsection 7.5.2 and also mentioned as example $\mathbf{E}_3^{ext}$ in Section 8.1):

$$p\ a \wedge p\ b \Rightarrow p\ (a \wedge b)$$

where $p_{o\to o}$, a_o and b_o are constants. Despite it's simplicity, this theorem cannot be solved automatically by any other theorem proving system known the author (apart from the tableaux based higher-order theorem prover HOT which is based on the tableaux calculus $\mathcal{HTE}$ [Koh98a]; $\mathcal{HTE}$ [Koh98a] translates the calculus $\mathcal{ER}$ presented in this thesis in a tableaux context). Negation and clause normalisation introduces the following three clauses:

$$\mathcal{C}_1 : [p\ a]^T \qquad \mathcal{C}_2 : [p\ b]^T \qquad \mathcal{C}_3 : [p\ (a \wedge b)]^F$$

In Figure 7.5 these clauses are represented by the nodes 5, 7 and 12.

In order to derive the empty clause LEO proceeds as follows: First, it inserts $\mathcal{C}_3$ into `SOS`, and the two others into `USABLE`. In the first loop $\mathcal{C}_3$ is chosen as the lightest clause (step 1), inserted into `USABLE` (step 2), and resolved against $\mathcal{C}_1$ and $\mathcal{C}_2$ (step 3), thereby yielding the clauses $\mathcal{C}_4 : [p\ (a \wedge b) = p\ a]^F$ and $\mathcal{C}_5 : [p\ (a \wedge b) = p\ b]^F$. Paramodulation (step 4) is not employed in LEO3 and factorisation (step 5) as well as primitive substitution (step 6) are not applicable. Thus, the clauses passed to `Processed` (step 7) are $\mathcal{C}_4$ and $\mathcal{C}_5$. The stores `EXT` (step 8) is still empty and continuation of (pre-)unification (step 9) is not employed in LEO. Next LEO tries to pre-unify both clauses in `Processed` (step 10). After decomposition the unification process ends up with a clash in both cases: $\mathcal{C}_6 : [(a \wedge b) = a]^F$ and $\mathcal{C}_7 : [(a \wedge b) = b]^F$ such that no clause is passed to `Unified`. As continuation of (pre-)unification (step 9) is not employed in LEO no clause is passed to `CONT`: the maximal unification search depth limit (usually 5 *FlexRigid* branchings) is not reached in both unification processes. Then the non-unifiable clauses $\mathcal{C}_6$ and $\mathcal{C}_7$ are analysed whether they are extensionally interesting (step 12). As unification constraints between terms of Boolean type are generally classified as

[2] Whereas the criterions (i) and (ii) mentioned here are fair, LEO currently employs additional criterions which are not so obvious and which are most likely not fair, e.g., never apply the extensionality treatment to *flex-flex*-pairs.

extensionally interesting, both clauses are subsequently stored in EXT. Unified is empty and thus no clause is passed to SOS (step 13), such that SOS is empty at the end of the first loop.

Nothing happens and no new clause is generated in the next loops until LEO is allowed to perform its extensionality treatment. The loops in which the extensionality treatment is applied are specified by a flag, and here we assume that the extensionality treatment is employed in every 6th loop. When reaching this loop LEO, e.g., chooses $\mathcal{C}_6$ from EXT (in step 7) and applies its compound extensionality treatment, thereby generating clause $\mathcal{C}_8 : [(a \land b) \equiv a]^F$, which passes the unification filter (in step 9) as it contains no further unification constraints, and which is inserted to Processed. From Processed $\mathcal{C}_8$ is finally passed (in step 10) to SOS.

In the 7th loop clause $\mathcal{C}_8$ is chosen from SOS and immediately normalised (step 1), leading to $\mathcal{C}_9 : [a]^T$, $\mathcal{C}_{10} : [a]^T \lor [b]^T$, and $\mathcal{C}_{11} : [a]^F \lor [b]^F$. These clause are then integrated to SOS, and as subsumption is applied, the clause $\mathcal{C}_{10}$ is removed.

In the following two loops LEO subsequently chooses $\mathcal{C}_9$ and $\mathcal{C}_{11}$ as new lightest clauses, and proceeds with its processing as described. The only interesting clause generated in these two loops is $\mathcal{C}_{12} : [b]^F$ (by resolution between $\mathcal{C}_9$ and $\mathcal{C}_{11}$). This clause is itself integrated to USABLE in one of the following loops.

In the 12th loop LEO then chooses clause $\mathcal{C}_7$ from EXT and applies its compound extensionality treatment to it (in step 7), thereby generating clause $\mathcal{C}_{13} : [(a \land b) \equiv b]^F$, which is passed to SOS at the end of this loop.

Clause normalisation of $\mathcal{C}_{13}$ at the beginning of loop 13 leads to $\mathcal{C}_{14} : [b]^T$, $\mathcal{C}_{15} : [a]^T \lor [b]^T$ and $\mathcal{C}_{16} : [a]^F \lor [b]^F$; both are immediately integrated (modulo subsumption) into SOS. $\mathcal{C}_{14}$ is chosen as the new lightest clause, and as the complementary clause $\mathcal{C}_{12}$ is already contained in USABLE, the empty clause is derived in this loop by resolution and (trivial) unification.

7.2.5 Realisation of the Challenges and the main Sources of Incompleteness

Within the current version of LEO the basic challenges mentioned in 7.2.1 are realised as follows:

1. **Extensional higher-order (pre-)unification** Instead of a full integration of the new extensionality rules into the employed higher-order pre-unification algorithm, we decided to separate the extensionality treatment from higher-order pre-unification (see steps 7, 10, and 12 above). Thereby the pre-unification filter simply passes extensionality interesting clauses to the store EXT in order to prevent from being removed from the the search space. The store EXT is itself heuristically sorted, such that the (heuristically) most interesting clauses (this notion clearly depends on the quality of the criterions employed by the sorting heuristics H5) are considered first in the exensionality treatment. This mechanism allows to delay the extensionality treatment in order to avoid an early search space explosion. It is obvious, that because of the possibly delay of the extensionality treatment, the sorting heuristics H5 for EXT strongly influences LEO's overall performance on examples, which indeed require the application of the extensionality principles. The extensionality treatment itself is exhaustive, i.e., it is applied to all extensionally interesting literals of the chosen clause from store EXT in once.

   Whereas the original motivation for the separation of the extensionality treatment was to avoid a complete re-implementation of the pre-unification algorithm provided by the KEIM-package and to examine first in a prototypical implementation of LEO whether the $\mathcal{ER}$ approach is in principle able to solve some simple theorems requiring the extensionality principles, it turned out that this separation and the possibility to delay the extensionality treatment is very fruitful if not even essential in practice.

Another interesting experience gained from practice is, that when allowing an extensionality treatment only in each, e.g., 6th loop, and when restricting primitive substitution to the logical connective $\neg$, LEO refutes many of the examples mentioned in Section 7.4 as follows: In the first 2 or 3 loops LEO performs a couple of interesting resolution, factorisation and primitive substitution steps. In the following loops he then often produces less interesting clauses which are often already subsumed by already given ones. Sometimes the SOS even becomes empty, just like in the example discussed in the previous section. But, meanwhile the store EXT contains some extensionally interesting clauses and from this LEO chooses then in the 6th loop the most promising one, applies the extensionality treatment thereby often generating new interesting facts which then subsequently positively influence the amount of interesting clauses derived in the following loops. This illustrates that LEO indeed applies the extensionality principles in a goal directed way, which contrasts ith the traditional approaches, as they would — provided that the extensionality axioms are added to the search space — always operate on the extensionality axioms and/or their consequences.

2. **Eager unification is essential but undecidable** The current version of LEO employs higher-order (pre-)unification only with respect to a certain search depth, which is specified by a flag. When reaching this depth LEO simply terminates its search in the current path and backtracks to the next one. It has already been pointed out by other authors (see for instance [Wol93, Pau94, BS94, Nad87]) that non-terminating higher-order unification constraints rarely occur in practice, such that the syntactical unifiability of many clauses can indeed be decided even when restricting the unification search depth to a certain limit. And for the rather simple examples discussed here, a unification search depth of 5 allowed nested branchings with rule *FlexRigid* turns out to be sufficient.[3] But not that LEO's architecture and main loop already presents with the (currently not realised) steps 8 and 11 a general solution to the problem. This solution consists in the generation of continuations for interrupted unification processes. The generated continuations[4] are heuristically sorted and stored in CONT, from where they can be chosen in a later loop of LEO. Thus, the idea is to realise a delayed continuation of interrupted unification processes with respect to a steadily increasing unification search depth in order to overcome the problems induced by the undecidability of higher-order (pre-)unification. The KEIM-toolbox already provides data structures and algorithms for unification continuations. Even though they are not yet employed in LEO — mainly as there was no need for increasing the unification search depth within the currently considered example domains (e.g., simple examples about sets as discussed in Subsection 7.4) and as it seems to be more promising to concentrate on the improvement of other aspects in LEO prototypical implementation first.

3. **The primitive substitution principle is infinitely branching** When using its

[3] Peter Andrews mentioned that his experience with the TPS-system is similar: Most of the examples examined within the TPS-system can been proven when restricting the unification depth in this sense. A unification depth of more than 15 *FlexRigid* branchings seems to be rarely needed in practice.

[4] Note that continuations are basically just ordinary clauses with additional information on the interrupted unification process (e.g., the particular search depth).

default settings LEO3 only imitates the logical connective $\neg$ when applying the primitive substitution principle in step 6. Analogously to the restriction of the unification search depth, this restriction turns out to be sufficient for proving simple examples about sets discussed in Subsection 7.4. Whereas the additional imitation of the logical connective $\vee$ only slightly influences the time LEO needs to solve these problems, the situation is quite different when imitating also universal quantifiers, e.g., all quantifiers up to order 5.

For more complicated examples it is certainly not sufficient to restrict the primitive substitution rule to the connective $\neg$ as, e.g., example THM15b discussed in [ABI+96] illustrates. The proof of this example in the TPS-system requires the primitive substitution of primitive or Leibniz equality (in LEO the latter would in fact mean to subsequently build up the respective instantiation by single primitive substitution steps imitating $\vee$ and $\neg$ in a row).

In order to improve LEO's treatment of the primitive substitution principle one can introduce a delay mechanism for primitive substitution similar to the stores `EXT` and `CONT`. The idea is to start a proof attempt by, e.g., only imitating $\neg$ and $\vee$, and then to subsequently add primitive substitutions of universal quantifiers in a delayed manner.

We briefly summarise the main sources of incompleteness in LEO3, i.e., the current version of LEO:

1. the restricted (pre-)unification depth in step 10 of the main loop,
2. the restricted application of the primitive substitution principles in step 6,
3. the heuristics employed for determining of extensionally interesting clauses in step 12, which have not been proven to ensure a Henkin completeness yet,
4. the compound extensionality treatment in step 7, which has not been proven to ensure a Henkin completeness yet,
5. the heuristics employed to sort the stores `SOS`, `EXT` (and `CONT`) which have not been proven to ensure fairness yet.

7.3 New Insights gained from LEO

7.3.1 Extensionality and Term indexing

Syntactical higher-order term indexing (see Subsection 7.1) must not be employed in the computation of resolution partners as otherwise no resolution step on only extensionally but not syntactically unifiable literals would be suggested.

This drawback is illustrated by the example discussed in Subsection 7.2.4: the key steps in this refutation — namely the resolution steps between $\mathcal{C}_3 : [p\ (a \wedge b)]^F$ and $\mathcal{C}_1 : [p\ a]^T$ or $\mathcal{C}_2 : [p\ b]^T$ at the very beginning — will not be performed in case syntactical higher-order term indexing is employed as a filter within the computation of possible resolution partners. Thus, LEO would fail to prove this theorem.

What we would need in extensional higher-order theorem proving is term indexing *modulo extensionality the extensionality principles*, which is obviously undecidable.

7.3.2 Extensionality and Subsumption

With respect to our higher-order subsumption filter — we face a problem, similar to the one sketched for the higher-order term indexing filter above. LEO's subsumption filter is a quite weak one, as it only looks for matchable literals on the basis of higher-order simplification, i.e., only employs first-order matching, when examining whether one clause subsumes another one. Analogous to the situation in higher-order term indexing, this filter does not take the extensionality principles into account. Clearly, this does not affect the completeness of our approach and the only effect of employing a weak subsumption filter when inserting clauses to a clause store is, that probably much more clauses are inserted as needed to ensure the completeness of the system. Especially the store EXT is affected, which maintains the extensionality interesting unification constraints. As the compound extensionality treatment is strongly delayed in LEO, it is consequently very important to avoid as many as possible subsumed clauses in store EXT. We illustrate this problem by the following two clauses (p, q and r are constants):

$$\mathcal{C}_1 : [(\lambda X_\iota.\ p_{\iota\to o}\ X) = (\lambda X_\iota.\ q_{\iota\to o}\ X \land r_{\iota\to o}\ X)]^F$$

$$\mathcal{C}_2 : [(\lambda X_\iota.\ r_{\iota\to o}\ X \land q_{\iota\to o}\ X) = (\lambda X_\iota.\ p_{\iota\to o}\ X)]^F$$

It is obvious that with respect to a syntactical higher-order matching approach neither one of both clauses is subsumed by the other. Even though, when applying the compound extensionality treatment and normalisation to them, they introduce identical proper clauses (modulo the names of the new Skolem constant s^i) into the search space:

$$\mathcal{C}_1 \rightsquigarrow \quad \mathcal{C}_3 : [p\ s^1]^F \lor [q\ s^1]^F \lor [r\ s^1]^F \quad \mathcal{C}_4 : [p\ s^1]^T \lor [q\ s^1]^T \quad \mathcal{C}_5 : [p\ s^1]^T \lor [r\ s^1]^T$$

$$\mathcal{C}_2 \rightsquigarrow \quad \mathcal{C}_6 : [p\ s^2]^F \lor [q\ s^2]^F \lor [r\ s^2]^F \quad \mathcal{C}_7 : [p\ s^2]^T \lor [q\ s^2]^T \quad \mathcal{C}_8 : [p\ s^2]^T \lor [r\ s^2]^T$$

The problem is that $\mathcal{C}_1$ and $\mathcal{C}_2$ subsume each other only with respect to a notion of extensional higher-order subsumption, but not with respect to a notion that is only based on syntactical higher-order matching or simplification. And, as extensional higher-order matching or unification is undecidable extensional higher-order subsumption seems not to be feasible in practice.

A practical solution that has yet not been examined in LEO, could be to employ the extensionality treatment before inserting a extensionally interesting clause into the store EXT and to insert the resulting clauses (clauses $\mathcal{C}_3 \ldots \mathcal{C}_8$ in our example) instead of the extensionally interesting clauses itself. One could then try to develop a higher-order subsumption approach that abstracts from the names of the Skolem constants introduced by the extensionality treatment and employ it as a filter for store EXT. In our example, this could prevent LEO from inserting all the clauses $\mathcal{C}_3 \ldots \mathcal{C}_8$ into EXT and to filter out, for instance, $\mathcal{C}_3, \mathcal{C}_4$ and $\mathcal{C}_5$. In such a modified approach, the idea of the store EXT would be to delay and control the insertion of the clauses obtained from the compound extensionality treatment rather than to delay the extensionality treatment of the extensionally interesting clauses itself.

7.4 Case Study

Boolean Properties of Sets A case study on the examples provided by the article *Boolean Properties of Sets* [TS89] has demonstrated that LEO sometimes outperforms state of the art theorem provers when reasoning about simple examples from set theory. Article [TS89] presents 97 quite trivial theorems about sets. To provide an impression, we present the examples 28, 80, 99, 104 and 111 (the examples in [TS89] are numbered, but not each natural number is associated with an example — this explains the example numbers greater than 97).

Examples:

set28) If $X \subseteq Y$ and $Y \subseteq X$, then $X = Y$

set80) If $(X \cap Y) \cup (X \backslash Y) = X$

set99) $(X \dot{-} Y) \dot{-} Z = X \dot{-} (Y \dot{-} Z)$

set104) Xmisses $\emptyset$

set111) $(X \cap Y)$ *misses* $(X\ Y)$

To formalise these theorems and to mechanise their proofs in a first-order theorem prover one has to axiomatise set theory in the system. In a case study carried out in the ILF-project [Dah97], Tarski Grothendieck set theory was employed. The results of this case study are presented in detail at `http://www-irm.mathematik.hu-berlin.de/~ilf/miz2atp/mizstat.html` and are summarised by Figure 7.2. We want to point out that examples 99, 104 and 111 could not be proven by any of the employed first-order theorem provers.

Figure 7.2 also relates the ILF case study to the respective case study carried out with LEO. Instead of employing Tarski Grothendieck set theory the above examples were encoded in this case study in classical type theory. This encoding fully exploits the expressiveness of the higher-order language based on the λ-calculus and describes sets by characteristic functions. For instance, the set $\{X_\iota | p\ X\}$ is encoded by the λ-expression $\lambda X_\iota.\ p\ X$ or simply p (by η-reduction). Based on this idea one can for instance define the notions $\in, \subseteq, \cap, \cup, \backslash$, $\dot{-}$, meets and *misses* as follows:

$$\begin{array}{rcl}
\in_{\alpha\to((\alpha\to o)\to o)} & := & \lambda X_\alpha.\ \lambda S_{\alpha\to o}.\ S\ X \\
\subseteq_{(\alpha\to o)\to(\alpha\to o)\to o} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \forall X_\alpha.\ X \in M\ \Rightarrow X \in N \\
\cap_{(\alpha\to o)\to(\alpha\to o)\to(\alpha\to o)} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \lambda X_\alpha.\ X \in M \wedge X \in N \\
\cup_{(\alpha\to o)\to(\alpha\to o)\to(\alpha\to o)} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \lambda X_\alpha.\ X \in M \vee X \in N \\
\backslash_{(\alpha\to o)\to(\alpha\to o)\to(\alpha\to o)} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \lambda X_\alpha.\ X \in M \wedge X \notin N \\
\dot{-}_{(\alpha\to o)\to(\alpha\to o)\to(\alpha\to o)} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ (M\backslash N) \cup (N\backslash M) \\
\mathrm{meets}_{(\alpha\to o)\to(\alpha\to o)\to o} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \exists X_\alpha.\ X \in M \wedge X \in N \\
\mathrm{misses}_{(\alpha\to o)\to(\alpha\to o)\to o} & := & \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \neg(M\ \mathrm{meets}\ N)
\end{array}$$

LEO's (version 1.0) detailed performance in seconds when solving the theorems of the article *Boolean Properties of Sets* [TS89] is presented in Figure 7.3. The reader may wonder how some problems can be solved without consuming any time. This comes from the fact, that the initial definition expansion (which is done by ΩMEGA) and initial clause normalisation was not measured in this case study. And surprisingly some examples, which seem to be hard to solve for the first-order provers (e.g., set104), can be proven just by normalising the expanded formula. A simple example is the theorem $\forall X_\iota.\ X \notin \emptyset$, which expands and normalises to $[\bot]^T$ (LEO immedately detects such contradictory clauses).

LEO was able to prove 95 of the 97 theorems and the only reason why the system cannot prove theorems 56 and 57 is, that after applying its extensionality treatment in one of the first loops in the proof attempt, LEO simply generates too many — in their sum contradictory — first-order clauses to be refuted in its prototypical implementation. By simply combining LEO with a sophisticated first-order theorem prover like SPASS (e.g., within the ΩMEGA-system), such that LEO could pass all the essentially first-order clauses generated by its extensionality treatment to this first-order reasoner, this two problems should easily be solvable as well. LEO's task in such a combined system would be to concentrate on the extensionality aspects and the first-order reasoner would perform brute-force search on the results in order to check if the extensionality reasoning performed so far by LEO is already sufficient to ensure the contradiction by straightforward first-order reasoning.

Solved theorems (of 97)	
Waldmeister (pure equality prover, only Th 72 and 99 have been tried)	1
Spass v0.78 (on Ultra Sparc 170)	72
Setheo v3.3 ("on" PVM)	76
CM v10-15-97 (ME Prover in Prolog)	72
CM v10-15-97 (with special cost function [hdef(d1,6,1,6)])	76
CM v9-22-97 (with definition expansion in the theorem)	79
Otter (auto)	60
Gandalf v. c-1.0b	47
Spass v0.54	52
Setheo	53
All Together	94
LEO	95

Figure 7.2: Case study with LEO on simple example about sets

set ex.	8	9	10	12	13	15	17	18	19	20	23	24	25	27	28
sec	.03	.02	.04	.05	.06	.04	.05	1.90	1.80	1.93	.05	.60	5.89	0	.22
set ex.	29	30	31	32	33	34	35	37	38	39	40	41	42	44	45
sec	.12	.05	.02	.13	.07	.12	.11	0	.02	.14	.08	.14	.11	.26	2.06
set ex.	46	47	48	49	50	51	52	53	54	55	58	59	60	61	62
sec	.07	.08	.12	.03	.04	.20	1.98	.21	.17	.17	.04	1.56	.02	.02	.03
set ex.	64	65	67	68	69	70	71	72	73	74	75	76	77	78	79
sec	.13	.03	.14	.04	.05	.17	.17	.57	.04	.03	.02	.04	.10	.05	.09
set ex.	80	81	82	83	84	85	86	87	88	89	90	91	92	93	95
sec	.08	.18	.09	.09	2.36	.14	.17	.14	.14	.19	.27	.16	.24	.05	.15
set ex.	96	97	98	99	100	101	102	104	110	111	112	113	114	115	116
sec	.16	.54	.47	2.00	25.66	.10	.06	0	.24	.02	.03	.07	.18	.50	.14
set ex.	117	118	119	120	121										
sec	.18	1.67	1.64	.25	.15										

Figure 7.3: LEO's performance in seconds on simple examples about sets

LEO-Command	Description
beta-normalise	β-normalises a clause
cd	changes the current directory
decompose	applies the decomposition rule *Dec* to a clause
delete-clause	deletes a clause from the current environment
end-report	closes the report stream
execute-log	subsequently executes the commands in a report file
exit	quits the current interpreter session
ext	applies the compound extensionality treatment to a clause
factorise	applies the factorisation rule *Fac* to a clause
help	lists a short description of all commands
imitate	applies the *FlexRigid* rule to a clause (only with imitation bindings)
new-log	opens a new report-file
para	applies the paramodulation rule *Para* to two clauses
pre-unifiers	computes and displays the syntactical pre-unifiers of a clause
pre-unify	pre-unifies a clause
prim-subst	applies the primitive substitution *Prim* rule to a clause
proj-imi	applies the *FlexRigid* rule to a clause
project	applies the *FlexRigid* rule to a clause (only with projection bindings)
prove	calls LEO in automatic mode to the current system state
read-problem	reads a problem file containing a proof problem in $\mathcal{POST}$ syntax, applies pre-clausification and stores the clauses either in the set of support or the set of usable clauses
resolve	applies the resolution rule *Res* to two clauses
set-flag	modifies the setting of a global flag
set-tactic	modifies the setting of a global tactic
show-clause	displays a clause
show-clauses	displays the content of clause stores
show-derivation	displays a linearised derivation of a clause
show-flags	displays the settings of all global flags
show-problem	displays the given problem ($\mathcal{POST}$ input)
show-proof	displays a found proof in linearised form
show-tactics	displays the available tactics (heuristics)
show-vars	displays the settings of all global variables
step-log	interactively executes the commands stored in a log file
subsumes	determines whether a clause subsumes another clause
write-derivation	writes a derivation of a clause in a file
write-louiproof	writes a proof in LOUI format in a file
write-proof	writes a proof in a file

Figure 7.4: Some interactive commands provided by the LEO-system

7.5 Additional Features of Leo

7.5.1 The Interactive Theorem Prover Leo

Leo can also be employed as an interactive theorem prover for extensional higher-order resolution. The main idea thereby is to provide a system that enables the user to analyse and illustrate the working principles of extensional higher-order resolution step by step. Especially in Leo's first implementation stages the interactive features turned out to be a very useful tool for experimenting with the calculus $\mathcal{ER}$. Figure 7.4 presents the most important interactive commands of Leo.

7.5.2 Loui as a Graphical Interface for Leo

Loui is a generic graphical user interface developed by Stephan Hess within his masters thesis [Hes99] and it is described in detail in [SHB+98, SHB+99]. Among the systems which already employ Loui as an interface or which are currently connected to it are: the mathematical assistant ΩMEGA [BCF+97], the induction theorem prover Inka [AHMS99, HS96], the higher-order proof planner λ-Clam [RSG98], and the Leo system described in this thesis. Apart from other useful information, Loui displays a proof graph (more precisely, a proof tree with co-references to already given nodes in the tree) and a linearised proof protocol.

Leo's connection to Loui allows for the graphical presentation of extensional higher-order resolution derivations and proofs with the gain for the user, that the structure of interactively or automatically created derivations become much more transparent to him.

Figure 7.5 presents Loui's visualisation for the proof of the small but challenging example $(p_{o\to o}a_o \land p_{o\to o}b_o \Rightarrow p_{o\to o}(a_o \land b_o)$ discussed in Subsection 7.2.4. Figure 7.6 presents another simple example about sets, which states that the power set of the empty set is the set that contains only the empty set.

In the Loui interface triangles represent initially given clauses stemming either directly from the assumptions or from the negated theorem. Derived clauses are presented as round nodes with links to their ancestors and their successors. Rhombi symbolise co-references to already displayed sub-derivations in order to avoid their duplication in the display.

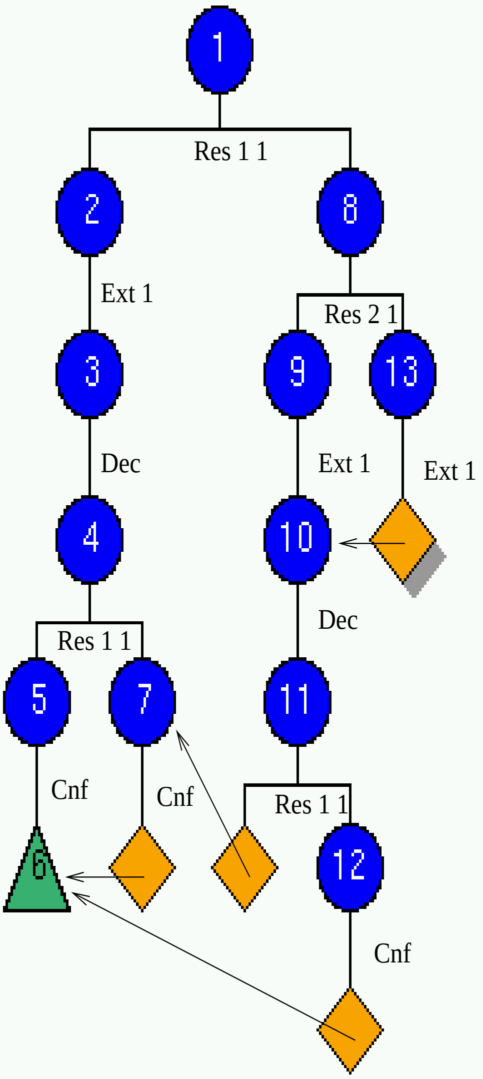
7.5.3 Integration to ΩMEGA

The Leo prover is integrated into the mathematical assistant system ΩMEGA [BCF+97], and the main aspects of this integration are:

- Leo and ΩMEGA rely on the same data structures and run in a single Lisp process. Hence, Leo can be viewed as ΩMEGA's logical engine, which can be employed to solve minor subproblems and to support ΩMEGA's proof planner [CrS98, Mel95, Mel94].
- Leo can be employed as an integrated higher-order theorem prover just like any other of the (currently) 10 external systems, that have been integrated to ΩMEGA.
- Leo can be employed as an interactive theorem prover for extensional higher-order resolution in ΩMEGA. This is supported by the Loui-interface, which provides menu entries supporting an easy selection for each of Leo's interactive commands (cf. Figure 7.4).

Within the ΩMEGA system, Leo and Tps [ABI+96, AINP90], which is the only other higher-order theorem prover integrated to ΩMEGA (the integration of Tps and ΩMEGA is decribed

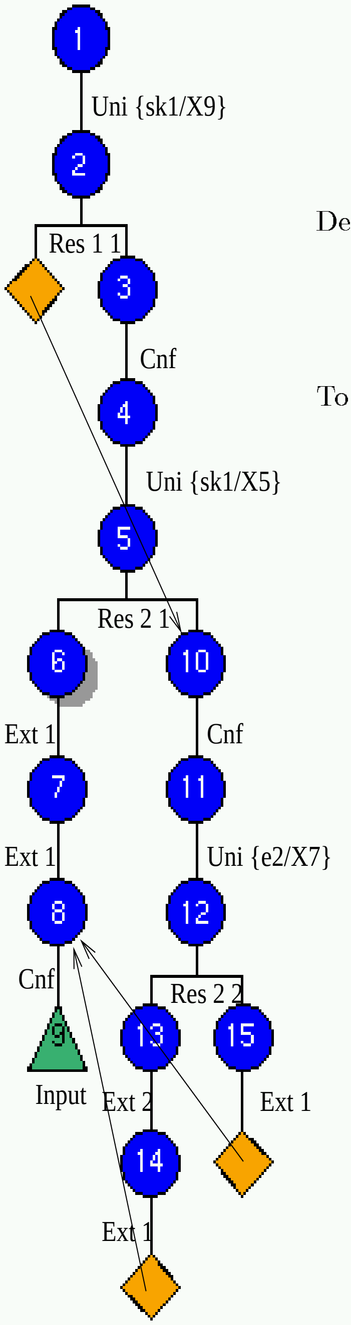
EXAMPLE: $p_{o\to o}\ a_o \wedge p_{o\to o}\ b_o \Rightarrow p_{o\to o}\ (a_o \wedge b_o)$



$1:\ \Box$
$2:\ [b]^T$
$3:\ [b = (a \wedge b)]^F$
$4:\ [(p\ b) = (p\ (a \wedge b))]^F$
$5:\ [p\ b]^T$
$6:\ [((p\ a) \wedge (p\ b)) \Rightarrow (p\ (a \wedge b))]^F$
$7:\ [p\ (a \wedge b)]^F$
$8:\ [b]^F$
$9:\ [a]^F \vee [b]^F$
$10:\ [a = (a \wedge b)]^F$
$11:\ [(p\ a) = (p\ (a \wedge b))]^F$
$12:\ [p\ a]^T$
$13:\ [a]^T$

Figure 7.5: LOUI-Visualisation of LEO's proof for example $\mathbf{E}_3^{ext}$

$$\boxed{\text{EXAMPLE: } \wp(\emptyset) = \{\emptyset\}}$$



Def.:

$$\wp := \lambda A_{\alpha\to o}.\ \lambda B_{\alpha\to o}.\ \overbrace{\forall.\ X_\alpha B\ X \Rightarrow A\ X}^{B\ \subseteq\ A}$$
$$\{\} := \lambda X_\beta.\ \lambda Y_\beta.\ Y = X$$
$$\emptyset := \lambda Y_\gamma.\ \mathbf{F}_o$$

To show:

$$\begin{array}{l} \neg((\lambda B_{\alpha\to o}.\ (\forall X_\alpha.\ (B\ X) \Rightarrow \bot)) = \\ \quad (\lambda B_{\alpha\to o}.\ B = \lambda Y_\alpha.\ \bot)) \end{array}$$

$$\begin{array}{ll}
1: & \square \\
2: & [e^1_{\alpha\to o}\ X^9_\alpha = e^1_{\alpha\to o}\ sk^1]^F \\
3,4: & [e^1_{\alpha\to o}\ X^8_\alpha]^T \\
5: & [e^1_{\alpha\to o}\ X^5_\alpha = e^1_{\alpha\to o}\ sk^1]^F \vee [e^1_{\alpha\to o}\ X^6_\alpha]^F \\
6: & [e^1_{\alpha\to o}\ X^2_\alpha]^F \vee [e^1_{\alpha\to o}\ X^3_\alpha]^F \\
7: & [e^1_{\alpha\to o}\ X^1_\alpha]^F \vee [e^1_{\alpha\to o} = \lambda Z.\ \bot]^T \\
8,9: & [(\lambda B_{\alpha\to o}.\ (\forall X_\alpha.\ (B\ X) \Rightarrow \bot)) = \\
 & \quad (\lambda B_{\alpha\to o}.\ B = \lambda Y_\alpha.\ \bot)]^F \\
10,11: & [e^1_{\alpha\to o}\ sk^1]^T \\
12: & [e^1_{\alpha\to o}\ e^2 = e^1_{\alpha\to o}\ X^7_\alpha]^F \vee [e^1_{\alpha\to o}\ sk^1]^T \\
13: & [e^1_{\alpha\to o}\ e^2]^T \vee [e^1_{\alpha\to o}\ sk^1]^T \\
14: & [e^1_{\alpha\to o} = \lambda Y_\alpha.\ \bot]^F \vee [e^1_{\alpha\to o}\ sk^1]^T \\
15: & [e^1_{\alpha\to o}\ X^4_\alpha]^F \vee [e^1_{\alpha\to o}\ sk^1]^T
\end{array}$$

Figure 7.6: LOUI-Visualisation of LEO's proof for example $\mathbf{E}_6^{ext}$.

in [BBS99, BS98b]) complement each other. The Tps system can explore rather deep search spaces and employs a clever mechanism for selectively expanding definitions [BA98]. But the extensionality treatment in Tps is rather weak (e.g., Tps can not solve our little problem illustrated in Figure 7.5 and Subsection 7.2.4).

Chapter 8

Examples

This chapter presents various examples that illustrate the basic ideas of the calculi $\mathcal{ER}$, $\mathcal{EP}$, and $\mathcal{ERUE}$ and compare refutations in the different calculi with each other. These examples especially demonstrate the importance of an appropriate extensionality treatment in higher-order theorem proving.

8.1 Extensionality in $\mathcal{ER}$

This section discusses five simple, but with respect to their mechanisation nevertheless challenging examples for an higher-order automated theorem prover. As already mentioned in Subsection 7.2.4, example $\mathbf{E}_3^{ext}$ is currently not automatically provable in any higher theorem prover apart form Leo. Note that all refutations are quite trivial and straightforward in the extensional higher-order resolution approach $\mathcal{ER}$.

$\mathbf{E}_1^{ext}$ $a_o \equiv b_o \Rightarrow (\forall P_{o \to o}.\ Pa \Rightarrow Pb)$

The non-trivial direction of the extensionality property for truth values: if a_o is equivalent to b_o then a_o is Leibniz equal to b_o. In the following refutation $p_{o \to o}$ is a new Skolem constant).

$$\begin{array}{lllll}
Cnf(\neg \mathbf{E}_1^{ext}): & \mathcal{C}_1 : [p\ a]^T & \mathcal{C}_2 : [p\ b]^F & \mathcal{C}_3 : [a]^T \vee [b]^F & \mathcal{C}_4 : [b]^T \vee [a]^F \\
Res(\mathcal{C}_1, \mathcal{C}_2): & \mathcal{C}_5 : [p\ a = p\ b]^F & & & \\
Dec(\mathcal{C}_5), Triv: & \mathcal{C}_6 : [a = b]^F & & & \\
Equiv(\mathcal{C}_6), Cnf: & \mathcal{C}_7 : [a]^F \vee [b]^F & \mathcal{C}_8 : [a]^T \vee [b]^T & &
\end{array}$$

The rest of the refutation with clauses $\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_7$, and $\mathcal{C}_8$ is obvious.

$\mathbf{E}_2^{ext}$ $p_{o \to o}\ (a_o \wedge b_o) \Rightarrow p\ (b \wedge a)$.

Any property which holds for $a \wedge b$ also holds for $b \wedge a$. This statement can also be read as: whenever $a \wedge b$ is in a set of truth values $p_{o \to o}$, then also $b \wedge a$ is in $p_{o \to o}$.

$$\begin{array}{lllll}
Cnf(\neg \mathbf{E}_2^{ext}): & \mathcal{C}_1 : [p\ (a \wedge b)]^F & \mathcal{C}_2 : [p\ (b \wedge a)]^F & & \\
Res(\mathcal{C}_1, \mathcal{C}_2): & \mathcal{C}_3 : [p\ (a \wedge b) = p\ (b \wedge a)]^F & & & \\
Dec(\mathcal{C}_3), Triv: & \mathcal{C}_4 : [a \wedge b = b \wedge a]^F & & & \\
Equiv(\mathcal{C}_4), Cnf: & \mathcal{C}_5 : [a]^F \vee [b]^F & \mathcal{C}_6 : [a]^T \vee [b]^T & \mathcal{C}_7 : [a]^T & \mathcal{C}_8 : [b]^T
\end{array}$$

The rest of the refutation is obvious: resolve $\mathcal{C}_5$ against $\mathcal{C}_7$ and $\mathcal{C}_8$.

$\mathbf{E}_3^{ext}$ $(p_{o\to o}\ a_o \land p\ b_o) \Rightarrow p\ (b \land a)$

If the truth values a_o and b_o are in set $p_{o\to o}$, then $a_o \land b_o$ is in $p_{o\to o}$ as well.

$$\begin{array}{llll}
Cnf(\neg\mathbf{E}_3^{ext}): & \mathcal{C}_1 : [p\ a]^T & \mathcal{C}_2 : [p\ b]^T & \mathcal{C}_3 : [p\ (a \land b)]^F \\
Res(\mathcal{C}_3, \mathcal{C}_1): & \mathcal{C}_4 : [p\ (a \land b) = p\ a]^F & & \\
Res(\mathcal{C}_3, \mathcal{C}_2): & \mathcal{C}_5 : [p\ (a \land b) = p\ b]^F & & \\
Dec(\mathcal{C}_4), Triv: & \mathcal{C}_6 : [a \land b = a]^F & & \\
Dec(\mathcal{C}_5), Triv: & \mathcal{C}_7 : [a \land b = b]^F & & \\
Equiv(\mathcal{C}_6), Cnf: & \mathcal{C}_8 : [a]^F \lor [b]^F & \mathcal{C}_9 : [a]^T \lor [b]^T & \mathcal{C}_{10} : [a]^T \\
Equiv(\mathcal{C}_7), Cnf: & \mathcal{C}_{11} : [a]^F \lor [b]^F & \mathcal{C}_{12} : [a]^T \lor [b]^T & \mathcal{C}_{13} : [b]^T
\end{array}$$

The rest of the refutation is obvious: Resolve $\mathcal{C}_8$ against $\mathcal{C}_{10}$ and $\mathcal{C}_{13}$.

$\mathbf{E}_4^{ext}$ $(\forall X_\iota.\ \forall P_{\iota\to o}.\ (P\ (m_{\iota\to\iota}X) \Rightarrow P\ (n_{\iota\to\iota}X)))$
$\Rightarrow (\forall Q_{(\iota\to\iota)\to o}.\ Q\ (\lambda X_\iota.\ mX) \Rightarrow Q\ (\lambda X_\iota.\ nX))$

This formula is an instance of the ξ-rule $(\forall X_\iota.\ m_{\iota\to\iota}X \doteq n_{\iota\to\iota}X) \Rightarrow (\lambda X_\iota.\ mX) \doteq (\lambda X_\iota.\ nX)$ for Leibniz equality; for details on the ξ-rule, e.g., see [HS86].

$$\begin{array}{lll}
Cnf(\neg\mathbf{E}_4^{ext}): & \mathcal{C}_1 : [P\ (m\ X)]^F \lor [P\ (n\ X)]^T & \\
 & \mathcal{C}_2 : [q\ (\lambda X.\ m\ X)]^T & \mathcal{C}_3 : [q\ (\lambda X.\ n\ X)]^F
\end{array}$$

Unfortunately the idea to resolve $\mathcal{C}_2$ and $\mathcal{C}_3$ immediately against $\mathcal{C}_1$ does not lead to successful refutation as the resulting unification constraints are not solvable. Therefore we choose another way and resolve between $\mathcal{C}_2$ and $\mathcal{C}_3$ (in the following derivation $p_{\iota\to o}$, $q_{(\iota\to\iota)\to o}$, and s_i are new Skolem constants):

$$\begin{array}{lll}
Res(\mathcal{C}_2, \mathcal{C}_3): & \mathcal{C}_4 : [q\ (\lambda X.\ m\ X) = q\ (\lambda X.\ n\ X)]^F & \\
Dec(\mathcal{C}_4), Triv: & \mathcal{C}_5 : [\lambda X.\ m\ X = \lambda X.\ n\ X]^F & \\
Func(\mathcal{C}_5): & \mathcal{C}_6 : [m\ s = n\ s]^F & \\
Leib(\mathcal{C}_6), Cnf: & \mathcal{C}_7 : [p\ (m\ s)]^T & \mathcal{C}_8 : [p\ (n\ s)]^F
\end{array}$$

We made a detour to the pre-unification part of the calculus and modified the clauses $\mathcal{C}_2$ and $\mathcal{C}_3$ in an extensionally appropriate way and $\mathcal{C}_2$ and $\mathcal{C}_3$ have now their counterparts in $\mathcal{C}_7$ and $\mathcal{C}_8$ But in contrast to $\mathcal{C}_2$ and $\mathcal{C}_3$ the new clauses can successfully be resolved against $\mathcal{C}_1$.

In our refutation $q_{(\iota\to\iota)\to o}, s_\iota, p_{\iota\to o}$ are new Skolem constants.

$\mathbf{E}_5^{ext}$ $(\forall X_\iota.\ \forall P_{\iota\to o}.\ P(m_{\iota\to\iota}X) \Rightarrow P(n_{\iota\to\iota}X)) \Rightarrow (\forall Q_{(\iota\to\iota)\to o}.\ Qm \Rightarrow Qn)$

This is an instance of the non-trivial direction of the functional extensionality axiom for type $\iota \to \iota$ (in the following derivation $p_{\iota\to o}$, $q_{(\iota\to\iota)\to o}$, and s_i are new Skolem constants):

$$\begin{array}{lll}
Cnf(\neg\mathbf{E}_5^{ext}): & \mathcal{C}_1 : [P\ (m\ X)]^F \lor [P\ (n\ X)]^T & \\
 & \mathcal{C}_2 : [q\ m]^T & \mathcal{C}_3 : [q\ n]^F \\
Res(\mathcal{C}_2, \mathcal{C}_3): & \mathcal{C}_4 : [q\ m = q\ n]^F & \\
Dec(\mathcal{C}_4), Triv: & \mathcal{C}_5 : [m = n]^F & \\
Func(\mathcal{C}_5): & \mathcal{C}_6 : [m\ s = n\ s]^F & \\
Leib(\mathcal{C}_6), Cnf: & \mathcal{C}_7 : [p\ (m\ s)]^T & \mathcal{C}_8 : [p\ (n\ s)]^F
\end{array}$$

The rest of the refutation is (like above) obvious: resolve $\mathcal{C}_7$ and $\mathcal{C}_8$ against $\mathcal{C}_1$.

In our refutation $q_{(\iota\to\iota)\to o}, s_\iota, p_{\iota\to o}$ are new Skolem constants.

$\mathbf{E}_6^{ext}$ $\wp(\emptyset) = \{\emptyset\}$

A (similar) proof for this example in calculus $\mathcal{ER}$ is also illustrated by Figure 7.6.

$$\begin{array}{ll}
Cnf(\neg\mathbf{E}_6^{ext}): & \mathcal{C}_1 : [(\lambda B_{o\alpha}.\ (\forall X_\alpha.\ (B\ X) \Rightarrow \bot)) = (\lambda B_{o\alpha}.\ B = \lambda Y_\alpha.\ \bot)]^F \\
Func(\mathcal{C}_1), Equiv, Cnf: & \mathcal{C}_2 : [e^1\ X]^F \vee [e^1 = \lambda Z.\ \bot]^T \\
 & \mathcal{C}_3 : [e^1\ s]^T \vee [e^1 = \lambda Z.\ \bot]^F \\
 & \mathcal{C}_4 : [e^1\ s]^T \vee [e^1\ X]^F \\
Func'(\mathcal{C}_2), Equiv', Cnf: & \mathcal{C}_5 : [e^1\ X]^F \vee [e^1\ Y]^F \\
Fac(\mathcal{C}_5), \mathcal{UNI}: & \mathcal{C}_6 :: [e^1\ X]^F \\
Func'(\mathcal{C}_3), Equiv, Cnf: & \mathcal{C}_7 : \mathcal{C}_2 : [e^1\ s]^T \vee [e^1\ e^2]^T \\
Res(\mathcal{C}_4, \mathcal{C}_7), Fac, \mathcal{UNI}: & \mathcal{C}_8 : [e^1\ s]^T \\
Res(\mathcal{C}_6, \mathcal{C}_8): \Box &
\end{array}$$

8.2 Decomposition in $\mathcal{ER}$

Example $\mathbf{E}^{Dec}$ below demonstrates the basic ideas of extensional higher-order resolution and illustrates why this approach can also be seen as test calculus for extensional higher-order E-unification. This example furthermore focuses on the role of the decomposition rule in connection with the extensionality rules and compares the slightly modified decomposition rule *Dec* employed in this thesis with the rule *Dec'* as used in [BK98a]:

$$\frac{\mathbf{C} \vee [h\ \overline{\mathbf{U}^n} = h\ \overline{\mathbf{V}^n}]^F}{\mathbf{C} \vee [\mathbf{U}^1 = \mathbf{V}^1]^F \vee \ldots \vee [\mathbf{U}^n = \mathbf{V}^n]^F}\ Dec'$$

Our example extends $\mathbf{E}_5^{ext}$ from the previous section: suppose we have four function constants, $f_{(\alpha\to\alpha)\to\alpha\to\alpha}$, $g_{(\alpha\to\alpha)\to\alpha\to\alpha}$, $h_{\alpha\to\alpha}$, and $j_{\alpha\to\alpha}$, and we know that f equals g and h equals j (in the E-unification perspective we can assume that this two equations define our theory E). We want to prove that under this assumptions (under this theory) application $(f\ h)$ equals application $(g\ j)$. Depending on the actual encoding of the assumptions and the assertion, a respective proof is either trivial or quite complicate to mechanise in classical type theory. A formulation that is not trivial to mechanise (as the application of the extensionality principles is required) and which uses Leibniz equality is:

$$\mathbf{E}^{Dec} \quad (\forall X_{\alpha\to\alpha}.\ \forall Y_\alpha.\ (f\ X\ Y) \doteq (g\ X\ Y)) \wedge (\forall Z_\alpha.\ (h\ Z) \doteq (j\ Z)) \Rightarrow (f\ h) \doteq (g\ j)$$

When expanding the definition of $\doteq$, negating the theorem and applying pre-clausification we obtain the following pre-clauses for this example:

$$\begin{array}{l}
\mathcal{C}_1 : [\forall P_{\alpha\to o}.\ \forall X_{\alpha\to\alpha}.\ \forall Y_\alpha.\ \neg(P\ (f\ X\ Y)) \vee (P\ (g\ X\ Y))]^T \\
\mathcal{C}_2 : [\forall Q_{\alpha\to o}.\ \forall Z_\alpha.\ \neg(Q\ (h\ Z)) \vee (Q\ (j\ Z))]^T \\
\mathcal{C}_3 : [\neg(\forall R_{(\alpha\to\alpha)\to o}.\ (\neg(R\ (f\ h))) \vee (R\ (g\ j)))]^T
\end{array}$$

Clause normalisation leads to:

$$\begin{array}{ll}
\mathcal{C}_4 : [P\ (f\ X\ Y)]^F \vee [P\ (g\ X\ Y))]^T & \mathcal{C}_5 : [Q\ (h\ Z)]^F \vee [Q\ (j\ Z)]^T \\
\mathcal{C}_6 : [r\ (f\ h)]^T & \mathcal{C}_7 : [r\ (g\ j)]^F
\end{array}$$

The reader may check that resolving $\mathcal{C}_6$ and $\mathcal{C}_7$ against $\mathcal{C}_4$ does not lead to successful refutation (c.f. discussion of example **E1** in [BK98a]). Instead we resolve between $\mathcal{C}_6$ and $\mathcal{C}_7$ and proceed by employing the difference reduction idea.

The first refutation we present here employs rule *Dec*. Below we will present a second refutation that alternatively use rule *Dec'*.

$$\begin{array}{lll}
Res(\mathcal{C}_6,\mathcal{C}_7): & \mathcal{C}_8 : [r\ (f\ h) = r\ (g\ j)]^F & \\
Dec(\mathcal{C}_8), Triv: & \mathcal{C}_9 : [f\ h = g\ j]^F & \\
Dec(\mathcal{C}_9): & \mathcal{C}_{10} : [f = g]^F \vee [h = j]^F & \\
2 \times Func(\mathcal{C}_{10}): & \mathcal{C}_{11} : [f\ t\ s = g\ t\ s]^F \vee [h\ u = j\ u]^F & \\
2 \times Leib(\mathcal{C}_{11}): & \mathcal{C}_{12} : [\forall P_{\alpha\to o}.\ \neg(P\ (f\ t\ s)) \vee (P\ (g\ t\ s))]^F & \\
 & \quad \vee [\forall Q_{\alpha\to o}.\ \neg(Q\ (h\ u)) \vee (Q\ (j\ u))]^F & \\
Cnf(\mathcal{C}_{12}): & \mathcal{C}_{13} : [p\ (f\ t\ s)]^T \vee [q\ (h\ u)]^T & \mathcal{C}_{14} : [p\ (f\ t\ s)]^T \vee [q\ (j\ u)]^F \\
 & \mathcal{C}_{15} : [p\ (g\ t\ s)]^F \vee [q\ (h\ u)]^T & \mathcal{C}_{16} : [p\ (g\ t\ s)]^F \vee [q\ (j\ u)]^F \\
Prim(\mathcal{C}_4): & \mathcal{C}_{17} : [P'\ (f\ X\ Y)]^T \vee [P'\ (g\ X\ Y))]^F & \\
Prim(\mathcal{C}_5): & \mathcal{C}_{18} : [Q'\ (h\ Z)]^T \vee [Q'\ (j\ Z)]^F &
\end{array}$$

Within this derivation $t_{\alpha\to\alpha}, s_\alpha, u_\alpha, p_{\alpha\to o}$ and $q_{\alpha\to o}$ are new Skolem terms and $P'_{\alpha\to o}, Q'_{\alpha\to o}$ are new predicate variables. The rest of the refutation employs straightforward resolution between the clauses $\mathcal{C}_{13}$–$\mathcal{C}_{16}$ and our assumption clauses $\mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_{17}$ and $\mathcal{C}_{18}$.

A alternative refutation that uses the rule *Dec'* is a bit more tricky:

$$\begin{array}{lll}
Res(\mathcal{C}_6,\mathcal{C}_7): & \mathcal{C}_8 : [r\ (f\ h) = r\ (g\ j)]^F & \\
Dec'(\mathcal{C}_8): & \mathcal{C}_9 : [f\ h = g\ j]^F & \\
Func(\mathcal{C}_9): & \mathcal{C}_{10} : [f\ h\ s = g\ j\ s]^F & \\
Leib(\mathcal{C}_{10}): & \mathcal{C}_{11} : [\forall P_{\alpha\to o}.\ \neg(P\ (f\ h\ s)) \vee (P\ (g\ j\ s))]^F & \\
Cnf(\mathcal{C}_{11}): & \mathcal{C}_{12} : [p\ (f\ h\ s)]^T & \mathcal{C}_{13} : [p\ (g\ j\ s)]^F \\
Res(\mathcal{C}_{12},\mathcal{C}_4): & \mathcal{C}_{14} : [P\ (g\ X\ Y)]^T \vee [p\ (f\ h\ s) = P\ (f\ X\ Y)]^F & \\
Res(\mathcal{C}_{13},\mathcal{C}_4): & \mathcal{C}_{15} : [P'\ (f\ X\ Y))]^F \vee [p\ (g\ j\ s) = P'\ (g\ X\ Y)]^F &
\end{array}$$

Note that the unification constraints in $\mathcal{C}_{14}$ and $\mathcal{C}_{15}$ are solvable by first imitating p and then either projecting to the argument of P or imitating to the argument of p. In our refutation we choose for $\mathcal{C}_{14}$ the partial projection solution $\sigma_1 := [(\lambda X_\alpha.\ p\ X)/P, h/X, s/Y]$ and for $\mathcal{C}_{15}$ the partial imitation solution $\sigma_2 := [(\lambda X_\alpha.\ p\ (g\ j\ s))/P']$. Both solutions can be computed in $\mathcal{ER}$ by applying eager unification. We propagate this solutions back to the non-unification literals of the clauses and proceed as follows:

$$\begin{array}{lll}
\mathcal{UNI}(\mathcal{C}_{14}): & \mathcal{C}_{16} : [p\ (g\ h\ s)]^T & \\
\mathcal{UNI}(\mathcal{C}_{15}): & \mathcal{C}_{17} : [p\ (g\ j\ s)]^F & \\
Res(\mathcal{C}_{16},\mathcal{C}_{17}): & \mathcal{C}_{18} : [p\ (g\ h\ s) = p\ (g\ j\ s)]^F & \\
2 \times Dec'(\mathcal{C}_{18}): & \mathcal{C}_{19} : [h\ s = j\ s]^F & \\
Leib(\mathcal{C}_{19}): & \mathcal{C}_{20} : [\forall Q_{\alpha\to o}.\ \neg(Q\ (h\ s)) \vee (Q\ (j\ s))]^F & \\
Cnf(\mathcal{C}_{20}): & \mathcal{C}_{21} : [q\ (h\ s)]^T & \mathcal{C}_{22} : [q\ (j\ s)]^T \\
Res(\mathcal{C}_{21},\mathcal{C}_5): & \mathcal{C}_{23} : [Q\ (j\ Z)]^T \vee [q\ (h\ s) = Q\ (h\ Z)]^F & \\
Res(\mathcal{C}_{22},\mathcal{C}_5): & \mathcal{C}_{24} : [Q'\ (h\ Z')]^F \vee [q\ (j\ s) = Q'\ (j\ Z')]^F & \\
Res(\mathcal{C}_{23},\mathcal{C}_{24}): & \multicolumn{2}{l}{[Q\ (j\ Z) = Q'\ (h\ Z')]^F \vee [q\ (h\ s) = Q\ (h\ Z)]^F \vee [q\ (j\ s) = Q'\ (j\ Z')]^F}
\end{array}$$

With unifier $\sigma := [(\lambda X_\alpha.\ q\ (h\ s))/Q, (\lambda X_\alpha.\ q\ X)/Q', s/Z']$ this unification constraint is obviously solvable, such that the empty clause is obtained by applying pre-unification.

This discussion raises the question whether rule *Dec* or rule rule *Dec'* is better suitable within calculus $\mathcal{ER}$. This problem can and should be clarified by experiments in practice. As an alternative to [BK98a] and in order to simplify the formal completeness proofs at certain points we decided in this paper to use *Dec*.

8.3 Leibniz Equality and Alternative Definitions in $\mathcal{ER}$

The examples discussed in this section focus on the equivalence of Leibniz equality and alternative definitions for equality in classical type theory.

The definitions for equality that are compared are:

Leibniz Equality

$$\dot{=}^{\alpha} := \lambda X_{\alpha}.\ \lambda Y_{\alpha}.\ \forall P_{\alpha\to o}.\ PX \Rightarrow PY$$

Reflexivity Definition

$$\ddot{=}^{\alpha} := \lambda X_{\alpha}.\ \lambda Y_{\alpha}.\ \forall Q_{\alpha\to\alpha\to o}.\ (\forall Z_{\alpha}.\ (Q\ Z\ Z)) \Rightarrow (Q\ X\ Y)$$

Modified Leibniz Equality

$$\dddot{=}^{\alpha} := \lambda X_{\alpha}.\ \lambda Y_{\alpha}.\ \forall P_{\alpha\to o}.\ ((a_o \lor \neg\ a_o) \land P\ X) \Rightarrow ((b_o \lor \neg\ b_o) \land P\ Y)$$

Leibniz equality employs the substitutivity principle to define equality, whereas the second alternative definition, which is presented and discussed in Andrews textbook [And86] at page 155, employs the reflexivity principle. The third (artificial) definition illustrates that there are infinetely many modifications of Leibniz equality (and analogously of the Reflexivity Definition) which all denote the same relation in Henkin semantics, namely equality. The examples furthermore illustrates that it is impossible to decide whether a given formula denotes the equality relation as this requires to decide whether two arbitrary formulae are equivalent.

$\mathbf{E}_1^{\dot{=}}$ **(i)** $(u \dot{=}^{\alpha} v) \Rightarrow (u \ddot{=}^{\alpha} v)$ and **(ii)** $(u \ddot{=}^{\alpha} v) \Rightarrow (u \dot{=}^{\alpha} v)$

In case (i) we use the following refutation ($q_{\alpha\to\alpha\to o}$ is a new Skolem constant):

$$\begin{array}{ll}
\mathcal{CNF}(i): & \mathcal{C}_1 : [P\ u]^F \lor [P\ v]^T \\
 & \mathcal{C}_2 : [q\ Z\ Z]^T \\
 & \mathcal{C}_3 : [q\ u\ v]^F \\
Res(\mathcal{C}_1,\mathcal{C}_3): & \mathcal{C}_4 : [P\ u]^F \lor [P\ v = q\ u\ v]^F \\
\mathcal{UNI}(\mathcal{C}_4)\ with\ [(\lambda X.\ q\ u\ X)/P]: & \mathcal{C}_5 : [q\ u\ v]^T \\
Res(\mathcal{C}_2,\mathcal{C}_5), Triv: & \Box
\end{array}$$

In case (ii) we proceeed as follows ($p_{\alpha\to o}$ is a new Skolem constant):

$$\begin{array}{ll}
\mathcal{CNF}(i): & \mathcal{C}_1 : [Q\ z\ z]^F \lor [Q\ u\ v]^T \\
 & \mathcal{C}_2 : [p\ u]^T \\
 & \mathcal{C}_3 : [p\ v]^F \\
Prim(\mathcal{C}_1), Subst: & \mathcal{C}_4 : [Q'\ z\ z]^T \lor [Q'\ u\ v]^F \\
Res(\mathcal{C}_1,\mathcal{C}_3): & \mathcal{C}_5 : [Q\ z\ z]^F \lor [Q\ u\ v = p\ v]^F \\
\mathcal{UNI}(\mathcal{C}_5)\ with\ [(\lambda X, Y.\ p\ Y)/P]: & \mathcal{C}_6 : [p\ z]^F \\
Res(\mathcal{C}_4,\mathcal{C}_2): & \mathcal{C}_7 : [Q'\ z\ z]^T \lor [Q'\ u\ v = p\ u]^F \\
\mathcal{UNI}(\mathcal{C}_5)\ with\ [(\lambda X, Y.\ p\ X)/P]: & \mathcal{C}_8 : [p\ z]^T \\
Res(\mathcal{C}_6,\mathcal{C}_8), Triv: & \Box
\end{array}$$

Alternatively one can directly prove $\doteq\doteq\ddot{=}$, but the proof is a little more bulky.

$\mathbf{E}_2^{\doteq}$ **(i)** $(u \doteq^{\alpha} v) \Rightarrow (u \overset{\cdots}{=}^{\alpha} v)$ and **(ii)** $(u \overset{\cdots}{=}^{\alpha} v) \Rightarrow (u \doteq^{\alpha} v)$

In case (i) we proceed as follows:

$$\begin{array}{ll}
\mathcal{CNF}(i): & \mathcal{C}_1 : [P\ u]^F \vee [P\ v]^T \\
 & \mathcal{C}_2 : [p\ u]^T \vee [a]^T \\
 & \mathcal{C}_3 : [p\ u]^T \vee [a]^F \\
 & \mathcal{C}_4 : [p\ v]^F \vee [a]^T \\
 & \mathcal{C}_5 : [p\ v]^F \vee [a]^F \\
Res(\mathcal{C}_2,\mathcal{C}_3), Fac, 2 \times Triv: & \mathcal{C}_6 : [p\ u]^T \\
Res(\mathcal{C}_4,\mathcal{C}_5), Fac, 2 \times Triv: & \mathcal{C}_7 : [p\ v]^F \\
Res(\mathcal{C}_1,\mathcal{C}_6): & \mathcal{C}_8 : [P\ v]^T \vee [P\ u = p\ u]^F \\
Res(\mathcal{C}_8,\mathcal{C}_7): & \mathcal{C}_9 : [P\ u = p\ u]^F \vee [P\ v = p\ v]^F \\
\mathcal{UNI}(\mathcal{C}_9) \text{ with } [p/P]: & \Box
\end{array}$$

In case (ii) we employ the following derivation:

$$\begin{array}{ll}
\mathcal{CNF}(i): & \mathcal{C}_1 : [a]^F \vee [P\ u]^F \vee [b]^F \vee [P\ v]^T \\
 & \mathcal{C}_2 : [a]^F \vee [P\ u]^F \vee [b]^T \vee [P\ v]^T \\
 & \mathcal{C}_3 : [a]^T \vee [P\ u]^F \vee [b]^F \vee [P\ v]^T \\
 & \mathcal{C}_4 : [a]^T \vee [P\ u]^F \vee [b]^T \vee [P\ v]^T \\
 & \mathcal{C}_5 : [p\ u]^T \\
 & \mathcal{C}_6 : [p\ v]^F \\
Res(\mathcal{C}_1,\mathcal{C}_2), 3 \times Fac, 4 \times Triv: & \mathcal{C}_7 : [a]^F \vee [P\ u]^F \vee [P\ v]^T \\
Res(\mathcal{C}_3,\mathcal{C}_4), 3 \times Fac, 4 \times Triv: & \mathcal{C}_8 : [a]^T \vee [P\ u]^F \vee [P\ v]^T \\
Res(\mathcal{C}_7,\mathcal{C}_8), 2 \times Fac, 3 \times Triv: & \mathcal{C}_9 : [P\ u]^F \vee [P\ v]^T \\
Res(\mathcal{C}_9,\mathcal{C}_5): & \mathcal{C}_{10} : [P\ v]^T \vee [P\ u = p\ u]^F \\
Res(\mathcal{C}_{10},\mathcal{C}_6): & \mathcal{C}_{11} : [P\ u = p\ u]^F \vee [P\ v = p\ v]^F \\
\mathcal{UNI}(\mathcal{C}_{11}) \text{ with } [p/P]: & \Box
\end{array}$$

Alternatively one can directly prove that $\doteq^{\alpha}\doteq^{\alpha\to\alpha\to o}\overset{\cdots}{=}^{\alpha}$). Again the proof is a little more bulky.

8.4 Reasoning about Sets with Leibniz Equality

The case study carried out with Leo on the examples presented in the Mizar article *Boolean Properties of Sets* [TS89] has already been discussed in Section 7.4. This case study makes use of the expressivness of classical type theory and encodes sets as characteristic functions instead of employing, e.g., Tarski Grothendieck set theory [Try89].

The particular proofs generated by Leo are in most cases quite short and elegant. Very interesting is, that some proofs are found immediately by definition expansion and clause normalisation.

Leo's performance together with the detailled proofs is reported at `http://www.ags.uni-sb.de/projects/deduktion/projects/hot/leo/`.

Currently the author experiments with the examples of the article Basic Properties of Sets [Byl89].

8.5 Positive Extensionality Rules in $\mathcal{EP}$ and $\mathcal{ERUE}$

The examples discussed in this section demonstrate that the positive extensionality rules (or extensionality axioms which we want to avoid) are unavoidable in order to reach Henkin completeness approaches for primitive equality. As discussed in detail in chapter 2.8 none of these examples can be proven in $\mathcal{EP}$ or $\mathcal{ERUE}$ without employing the additional extensionality rules.

$\mathbf{E}_1^{Para}$ $\mathcal{C}_1 : [a = \neg a]^T$

Refutation in $\mathcal{EP}$ and $\mathcal{ERUE}$

$$\begin{array}{lll}
Equiv'(\mathcal{C}_1), \mathcal{CNF}: & \mathcal{C}_2 : [A]^F \vee [A]^F & \mathcal{C}_3 : [A]^T \vee [A]^T \\
Fac(\mathcal{C}_2), \mathcal{UNI}, Fac(\mathcal{C}_3), \mathcal{UNI}: & \mathcal{C}_4 : [A]^F & \mathcal{C}_5 : [A]^T \\
Res(\mathcal{C}_4, \mathcal{C}_5), \mathcal{UNI}: & \mathcal{C}_6 : \square &
\end{array}$$

As rule *Para* is not needed this refutation is also possible in calculus $\mathcal{ERUE}$.

$\mathbf{E}_2^{Para}$ $\mathcal{C}_1 : [G\ X =^{\iota \to o} p]^T$

Refutation in $\mathcal{EP}$ and $\mathcal{ERUE}$

$$\begin{array}{lll}
Func'(\mathcal{C}_1): & \mathcal{C}_2 : [G\ X\ Y =^o p\ Y]^T & \\
Equiv'(\mathcal{C}_1): & \mathcal{C}_3 : [G\ X\ Y]^F \vee [p\ Y]^T & \mathcal{C}_4 : [G\ X\ Y]^T \vee [p\ Y]^F \\
Prim(\mathcal{C}_3), Subst: & \mathcal{C}_5 : [G'\ X\ Y]^T \vee [p\ Y]^T & \\
Prim(\mathcal{C}_4), Subst: & \mathcal{C}_6 : [G''\ X\ Y]^F \vee [p\ Y]^F & \\
Fac(\mathcal{C}_5), \mathcal{UNI}: & \mathcal{C}_7 : [p\ Y]^T & \\
Fac(\mathcal{C}_6), \mathcal{UNI}: & \mathcal{C}_8 : [p\ Y]^F & \\
Res(\mathcal{C}_7, \mathcal{C}_8), \mathcal{UNI}: & \mathcal{C}_9 : \square &
\end{array}$$

As rule *Para* is not needed this proof is also possible in calculus $\mathcal{ERUE}$.

$\mathbf{E}_3^{Para}$ $\mathcal{C}_1 : [m = \lambda X_o.\ (\exists X_o.\ X \wedge \neg X)]^T$

Refutation in $\mathcal{EP}$ and $\mathcal{ERUE}$

$$\begin{array}{lll}
Func'(\mathcal{C}_1): & \mathcal{C}_2 : [M\ Y_o = (\exists X_o.\ X \wedge \neg X)]^T & \\
Equiv'(\mathcal{C}_2): & \mathcal{C}_3 : [M\ Y]^F \vee [s]^T & \mathcal{C}_4 : [M\ Y]^F \vee [s]^F \\
Prim(\mathcal{C}_3), Subst: & \mathcal{C}_5 : [H\ Y]^T \vee [s]^T & \\
Fac(\mathcal{C}_4), \mathcal{UNI}, Fac(\mathcal{C}_5), \mathcal{UNI}: & \mathcal{C}_6 : [s]^T & \mathcal{C}_7 : [s]^F \\
Res(\mathcal{C}_6, \mathcal{C}_7), \mathcal{UNI}: & \mathcal{C}_8 : \square &
\end{array}$$

where s_o is a Skolem constant for X. As rule *Para* is not needed this proof is also possible in calculus $\mathcal{ERUE}$.

$\mathbf{E}_4^{Para}$ $\mathcal{C}_1 : [m = \lambda X_o.\ \neg(m\ X)]^T$

Refutation in $\mathcal{EP}$ and $\mathcal{ERUE}$

$$\begin{array}{lll}
Func'(\mathcal{C}_1): & \mathcal{C}_2 : [m\ Y = \neg(m\ Y)]^T & \\
Equiv'(\mathcal{C}_2): & \mathcal{C}_3 : [m\ Y]^T \vee [m\ Y]^T & \mathcal{C}_4 : [m\ Y]^F \vee [m\ Y]^F \\
Fac(\mathcal{C}_3), \mathcal{UNI}, Fac(\mathcal{C}_4), \mathcal{UNI}: & \mathcal{C}_5 : [m\ Y]^T & \mathcal{C}_6 : [m\ Y]^F \\
Res(\mathcal{C}_5, \mathcal{C}_6), \mathcal{UNI}: & \mathcal{C}_7 : \square &
\end{array}$$

As rule *Para* is not needed this proof is also possible in calculus $\mathcal{ERUE}$.

$\mathbf{E}_5^{Para}$ $\mathcal{C}_1 : [P\ q =^{\iota\to\iota\to\iota\to o} P\ r]^T$ $\qquad \mathcal{C}_2 : [q\ X =^{\iota\to o} \neg(r\ X)]^T$
where $P_{(\iota\to\iota\to o)\to(\iota\to\iota\to\iota\to o)}, X_\iota$ are free variables and $q_{\iota\to\iota\to o}, r_{\iota\to\iota\to o}$ are function constants.

Refutation in $\mathcal{EP}$

$$\begin{array}{ll}
3 \times Func'(\mathcal{C}_1): & \mathcal{C}_3 : [P\ q\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3 =^o P\ r\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^T \\
Func'(\mathcal{C}_2): & \mathcal{C}_4 : [q\ X\ Z_\iota =^o \neg(r\ X\ Z)]^T \\
Para(\mathcal{C}_4, \mathcal{C}_3): & \mathcal{C}_5 : [P\ r\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^T \vee [P\ q\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3 =^o (q\ X\ Z =^o \neg(r\ X\ Z))]^F
\end{array}$$

By pre-unifying $\mathcal{C}_5$ one can compute the following unifier for its unification constraint: $[(\lambda U_{\iota\to\iota\to o}, V_\iota, W_\iota, T_\iota.\ U\ V\ W =^o \neg(r\ V\ W))/P, V/Y^1, W/Y^2]$. Thus by eager unification applied to $\mathcal{C}_5$ with get:

$$\mathcal{UNI}(\mathcal{C}_5), Subst: \quad \mathcal{C}_6 : [r\ X\ Z =^o \neg(r\ X\ Z)]^F$$

The rest of the refutation is analogous to $\mathbf{E}_1^{Para}$.

Refutation in $\mathcal{ERUE}$

$$\begin{array}{lll}
3 \times Func'(\mathcal{C}_1): & \mathcal{C}_3 : [P\ q\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3 =^o P\ r\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^T & \\
Func'(\mathcal{C}_2): & \mathcal{C}_4 : [q\ X\ Z_\iota =^o \neg(r\ X\ Z)]^T & \\
Equiv'(\mathcal{C}_3): & \mathcal{C}_5 : [P\ q\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^F \vee [P\ r\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^T & \\
 & \mathcal{C}_6 : [P\ q\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^T \vee [P\ r\ Y_\iota^1\ Y_\iota^2\ Y_\iota^3]^F & \\
Equiv'(\mathcal{C}_4), Cnf: & \mathcal{C}_7 : [q\ X\ Z_\iota]^F \vee [r\ X\ Z]^F & \mathcal{C}_8 : [q\ X\ Z_\iota]^T \vee [r\ X\ Z]^T
\end{array}$$

The rest of the refutation is straightforward resolution on $\mathcal{C}_5, \ldots, \mathcal{C}_8$.

8.6 Comparing $\mathcal{EP}$ and $\mathcal{ERUE}$

Properties of Primitive Equality The following examples demonstrate that primitive equality denotes an extensional congruence relation in all Henkin models (i.e., the intended equality relation). The single properties to be checked are reflexivity ($\mathbf{E}_r^=$), symmetry ($\mathbf{E}_s^=$), transitivity ($\mathbf{E}_t^=$), congruence ($\mathbf{E}_c^=$) and extensionality ($\mathbf{E}_e^=$).

$\mathbf{E}_r^=$ $\forall A_\alpha.\ A = A.$
Negation and clause normalisation leads to (a_α is a new Skolem constant):

$$\mathcal{C}_1 : [a = a]^F$$

Refutation in $\mathcal{EP}$: immediately by unification.
Refutation in $\mathcal{ERUE}$: immediately by unification.

$\mathbf{E}_s^=$ $\forall A_\alpha, B_\alpha.\ (A =^\alpha B) \Rightarrow (B =^\alpha A).$
Negation and clause normalisation leads to ($a_\alpha, b_\alpha, p_{\alpha\to o}$ are new Skolem constants):

$$\mathcal{C}_1 : [a =^\alpha b]^T \qquad \mathcal{C}_2 : [b =^\alpha a]^F$$

Refutation in $\mathcal{EP}$:

$$\begin{array}{lll}
Leib(\mathcal{C}_2), \mathcal{CNF}: & \mathcal{C}_3 : [p\ a]^T & \mathcal{C}_4 : [p\ b]^F \\
Para(\mathcal{C}_1, \mathcal{C}_3), Triv: & \mathcal{C}_5 : [p\ b]^T & \\
Res(\mathcal{C}_4, \mathcal{C}_5), Triv: & \Box &
\end{array}$$

Refutation in $\mathcal{ERUE}$:

$$Res(\mathcal{C}_1,\mathcal{C}_2), Triv: \ \Box \text{ (possible beause of symmetry convention in } \mathcal{ERUE})$$

$\mathbf{E}_t^=$ $\forall A_\alpha.\ \forall B_\alpha.\ \forall C_\alpha.\ (A = B) \wedge (B = C) \Rightarrow (A = C)$.
Negation and clause normalisation leads to ($a_\alpha, b_\alpha, c_\alpha, p_{\alpha\to o}$ are new Skolem constants):

$$\mathcal{C}_1 : [a = b]^T \qquad \mathcal{C}_2 : [b = c]^T \qquad \mathcal{C}_3 : [a = c]^F$$

Refutation in $\mathcal{EP}$:

$$\begin{array}{lll}
Leib(\mathcal{C}_3), \mathcal{CNF}: & \mathcal{C}_4 : [p\ a]^T & \mathcal{C}_5 : [p\ c]^F \\
Para(\mathcal{C}_1,\mathcal{C}_4), Triv: & \mathcal{C}_6 : [p\ b]^T & \\
Para(\mathcal{C}_2,\mathcal{C}_6), Triv: & \mathcal{C}_7 : [p\ c]^T & \\
Res(\mathcal{C}_5,\mathcal{C}_7), Triv: & \Box &
\end{array}$$

Refutation in $\mathcal{ERUE}$:

$$\begin{array}{ll}
Res(c3, c2): & \mathcal{C}_4 : [(a = c) = (b = c)]^F \\
2 \times Dec(c4), Triv: & \mathcal{C}_5 : [a = b]^F \\
Res(c1, c5), \mathcal{UNI}: & \mathcal{C}_6 : \Box
\end{array}$$

$\mathbf{E}_c^=$ $\forall F_{\alpha\to\beta}.\ \forall A_\alpha.\ \forall B_\alpha.\ (A =^\alpha B) \Rightarrow (F\ A =^\beta F\ B)$.
Negation an clause normalisation leads to ($a_\alpha, b_\alpha, f_{\alpha\to\beta}, p_{\beta\to o}$ are new Skolem constants):

$$\mathcal{C}_1 : [a = b]^T \qquad \mathcal{C}_2 : [f\ a = f\ b]^F$$

Refutation in $\mathcal{EP}$:

$$\begin{array}{lll}
Leib(\mathcal{C}_2), \mathcal{CNF}: & \mathcal{C}_3 : [p\ (f\ a)]^T & \mathcal{C}_4 : [p\ (f\ b)]^F \\
Para(\mathcal{C}_1,\mathcal{C}_3), Triv: & \mathcal{C}_5 : [p\ (f\ b)]^T & \\
Res(\mathcal{C}_5,\mathcal{C}_4), Triv: & \Box &
\end{array}$$

Refutation in $\mathcal{ERUE}$:

$$\begin{array}{ll}
Dec(c2), Triv: & \mathcal{C}_3 : [a = b]^F \\
Res(c1, c3), \mathcal{UNI}: & \mathcal{C}_4 : \Box
\end{array}$$

$\mathbf{E}_e^=$ $\forall F_{\alpha\to\beta}.\ \forall G_{\alpha\to\beta}.\ (\forall A_\alpha.\ (F\ A =^\beta G\ A)) \Rightarrow (F = G)$.
Negation and clause normalisation leads to ($f_{\alpha\to\beta}, g_{\alpha\to\beta}, p_{\beta\to o}, s_\beta$ are new Skolem constants):

$$\mathcal{C}_1 : [f\ A = g\ A]^T \qquad \mathcal{C}_2 : [f = g]^F$$

Refutation in $\mathcal{EP}$:

$$\begin{array}{lll}
Func(\mathcal{C}_2): & \mathcal{C}_3 : [f\ s = g\ s]^F & \\
Leib(\mathcal{C}_3), \mathcal{CNF}: & \mathcal{C}_4 : [p\ (f\ s)]^T & \mathcal{C}_5 : [p\ (g\ s)]^F \\
Para(\mathcal{C}_1,\mathcal{C}_4), \mathcal{UNI}: & \mathcal{C}_6 : [p\ (g\ s)]^T & \\
Res(\mathcal{C}_6,\mathcal{C}_5), \mathcal{UNI}: & \Box &
\end{array}$$

Refutation in $\mathcal{ERUE}$:

$$\begin{array}{ll}
Func(\mathcal{C}_2): & \mathcal{C}_3 : [f\ s = g\ s]^F \\
Res(c1, c3), \mathcal{UNI}: & \mathcal{C}_4 : \Box
\end{array}$$

Note that the $\mathcal{ERUE}$ refutations are in all cases shorter and more elegant.

Especially $\mathbf{E}_e^=$ shows that in extensional higher-order paramodulation it might be useful to allow paramodulation also on unification constraints in order to get shorter and simpler proofs whithout recursive calls from within the unification process. On the other hand it seems to be quite complicated to guide an approach that simultaneously paramodulates into unification constraints and employs recursive calls to the overall refutation procedure from within unification.

Primitive Equality and Leibniz Equality We analyse in calculi $\mathcal{EP}$ and $\mathcal{ERUE}$ whether Leibniz equality and primitive equality denote the same relation.

$\mathbf{E}_1^=$ We prove that $\dot{=}^\alpha =^{\alpha\to\alpha\to o} =^\alpha$, i.e. that $(\lambda X_\alpha.\ \lambda Y_\alpha.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y) =^{\alpha\to\alpha\to o} =^\alpha$

Refutation in $\mathcal{EP}$ (u_α, v_α and $p_{\alpha\to o}, q_{\alpha\to o}, r_{\alpha\to o}$ are Skolem constants)

$\mathcal{CNF}(\mathbf{E}_1^=)$:	$\mathcal{C}_1 : [(\lambda X_\alpha.\ \lambda Y_\alpha.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y) =^{\alpha\to\alpha\to o} =^\alpha]^F$
$2 \times Func(\mathcal{C}_1)$:	$\mathcal{C}_2 : [(\forall P_{\alpha\to o}.\ P\ u \Rightarrow P\ v) =^o (u =^\alpha v)]^F$
$Equiv(\mathcal{C}_2), \mathcal{CNF}$:	$\mathcal{C}_3 : [p\ u]^T \vee [u =^\alpha v]^F$
	$\mathcal{C}_4 : [p\ u]^F \vee [u =^\alpha v]^F$
	$\mathcal{C}_5 : [P\ v]^T \vee [u =^\alpha v]^T \vee [P\ u]^F$
$Res(\mathcal{C}_3, \mathcal{C}_4), Triv$:	$\mathcal{C}_6 : [u =^\alpha v]^F \vee [u =^\alpha v]^F$
$2 \times Leib(\mathcal{C}_6)$:	$\mathcal{C}_7 : [\forall Q_{\alpha\to o}.\ Q\ u \Rightarrow Q\ v]^F \vee [\forall R_{\alpha\to o}.\ R\ u \Rightarrow R\ v]^F$
$\mathcal{CNF}(\mathcal{C}_7)$:	$\mathcal{C}_8 : [q\ u]^T \vee [r\ u]^T$
	$\mathcal{C}_9 : [q\ u]^T \vee [r\ v]^F$
	$\mathcal{C}_{10} : [q\ v]^F \vee [r\ u]^T$
	$\mathcal{C}_{11} : [q\ v]^F \vee [r\ v]^F$
$Prim(\mathcal{C}_5)$:	$\mathcal{C}_{12} : [P\ v]^T \vee [u =^\alpha v]^T \vee [P\ u]^F$
	$\vee [P =^{\alpha\to o} (\lambda X_\alpha.\ (H_{\alpha\to\alpha}\ X) =^\alpha (H'_{\alpha\to\alpha}\ X))]^F$
$Subst(\mathcal{C}_{12})$:	$\mathcal{C}_{13} : [(H\ v) =^\alpha (H'\ v)]^T \vee [u =^\alpha v]^T \vee [(H\ u) =^\alpha (H'\ u)]^F$
$Fac(\mathcal{C}_{13})$:	$\mathcal{C}_{14} : [u =^\alpha v]^T \vee [(H\ u) =^\alpha (H'\ u)]^F$
	$\vee [((H\ v) =^\alpha (H'\ v)) =^o (u =^\alpha v)]^F$
$2 \times Dec(\mathcal{C}_{14}), Triv$:	$\mathcal{C}_{15} : [u =^\alpha v]^T \vee [(H\ u) =^\alpha (H'\ u)]^F \vee [(H\ v) = u]^F$
	$\vee [(H'\ v) = v]^F$
$2 \times FlexRigid(\mathcal{C}_{15})$:	$\mathcal{C}_{16} : [u =^\alpha v]^T \vee [(H\ u) =^\alpha (H'\ u)]^F \vee [(H\ v) = u]^F$
	$\vee [(H'\ v) = v]^F \vee [H = (\lambda X_\alpha.\ u)]^F \vee [H' = (\lambda X_\alpha.\ X)]^F$
$2 \times Subst(\mathcal{C}_{16})$:	$\mathcal{C}_{17} : [u =^\alpha v]^T \vee [u =^\alpha u]^F \vee [u = u]^F \vee [v = v]^F$
$3 \times Triv(\mathcal{C}_{17})$:	$\mathcal{C}_{18} : [u =^\alpha v]^T$

$4 \times Para(\mathcal{C}_{18}, \{\mathcal{C}_8, \mathcal{C}_9, \mathcal{C}_{10}\}), 3 \times Triv$:

$\mathcal{C}_{19} : [q\ v]^T \vee [r\ u]^T \quad \mathcal{C}_{20} : [q\ v]^T \vee [r\ v]^F \quad \mathcal{C}_{21} : [q\ v]^F \vee [r\ v]^T$

Straightforward Resolution on $\mathcal{C}_{19}, \mathcal{C}_{20}, \mathcal{C}_{21}, \mathcal{C}_{11}$: □

Refutation in $\mathcal{ERUE}$

We replay the derivation above and instead of the paramodulation steps between $\mathcal{C}_{18}$ and $\mathcal{C}_8, \mathcal{C}_9, \mathcal{C}_{10}$ we resolve between $\mathcal{C}_{18}$ and the unification constraint $\mathcal{C}_6$.

$\mathbf{E}_2^=$ We prove that $\dot{=}^\alpha \dot{=}^{\alpha\to\alpha\to o} =^\alpha$, i.e., that $\forall Q_{(\alpha\to\alpha\to o)\to o}.\ (Q\ (\lambda X.\ \lambda Y.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y)) \Rightarrow (Q\ =^\alpha)$

Refutation in $\mathcal{EP}$ ($q_{(\alpha\to\alpha\to o)\to o}$ is a Skolem constant):

$$\begin{array}{ll}
\mathcal{CNF}(\mathbf{E}_2^=): & \mathcal{C}_1 : [q\ (\lambda X.\ \lambda Y.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y)]^T \\
 & \mathcal{C}_2 : [q\ =^{\alpha}]^F \\
Res(\mathcal{C}_1,\mathcal{C}_2): & \mathcal{C}_3 : [(q\ (\lambda X.\ \lambda Y.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y)) =^{\alpha\to\alpha\to o} (q\ =^{\alpha})]^F \\
Dec(\mathcal{C}_3), Triv: & \mathcal{C}_4 : [(\lambda X.\ \lambda Y.\ \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y) =^{\alpha\to\alpha\to o} =^{\alpha}]^F
\end{array}$$

$\mathcal{C}_4$ is identical to $\mathcal{C}_1$ in the refutation for $\mathbf{E}_1^=$ above:. . . □

Refutation in $\mathcal{ERUE}$

We employ the previous initial derivation and then an refutation that is analogous to the $\mathcal{ERUE}$-refutation for $\mathbf{E}_1^=$.

Reasoning about Sets with Primitive Equality

$\mathbf{E}_1^{set}$ Let the set of odd numerals be defined as the set of non-even numerals. Then the power set of the set of odd numerals greater than 100 is equal to the power set of the set of even numerals greater that 100:

$$\begin{array}{l}
\{X|\ odd\ X \wedge num\ X\} = \{X|\ \neg\ ev\ X \wedge num\ X\} \Rightarrow \\
\quad \wp\{X|\ odd\ X \wedge num\ X \wedge X > 100\} = \wp\{X|\ \neg\ ev\ X \wedge num\ X \wedge X > 100\}
\end{array}$$

where the power set is defined by $\wp := \lambda M_{\alpha\to o}.\ \lambda N_{\alpha\to o}.\ \forall X_{\alpha}.\ N\ X \Rightarrow M\ X$. Negation and clause normalisation leads to:

$$\begin{array}{l}
\mathcal{C}_1 : [(\lambda X.\ odd\ X \wedge num\ X) = (\lambda X.\ \neg\ ev\ X \wedge num\ X)]^T \\
\mathcal{C}_2 : [(\lambda N.\ \forall X.\ N\ X \Rightarrow ((odd\ X \wedge num\ X) \wedge X > 100)) = \\
\qquad (\lambda N.\ \forall X.\ N\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)))]^F
\end{array}$$

The probably simplest refutation in calculus $\mathcal{EP}$ is:

$$\begin{array}{ll}
Func'(\mathcal{C}_1): & \mathcal{C}_3 : [(odd\ Y \wedge num\ Y) = (\neg\ ev\ Y \wedge num\ Y)]^F \\
Func(\mathcal{C}_2): & \mathcal{C}_4 : [(\forall X.\ n\ X \Rightarrow ((odd\ X \wedge num\ X) \wedge X > 100)) \\
 & \qquad = (\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)))]^F \\
Equiv(\mathcal{C}_4): & \mathcal{C}_5 : [\forall X.\ n\ X \Rightarrow ((odd\ X \wedge num\ X) \wedge X > 100)]^T \vee \\
 & \qquad [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^T \\
 & \mathcal{C}_6 : [\forall X.\ n\ X \Rightarrow ((odd\ X \wedge num\ X) \wedge X > 100)]^F \vee \\
 & \qquad [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^F \\
Para(\mathcal{C}_3,\mathcal{C}_5), Triv: & \mathcal{C}_7 : [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^T \vee \\
 & \qquad [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^T \\
Fac(\mathcal{C}_7), Triv: & \mathcal{C}_8 : [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^T \\
Para(\mathcal{C}_6,\mathcal{C}_3), Triv: & \mathcal{C}_9 : [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^F \vee \\
 & \qquad [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^F \\
Fac(\mathcal{C}_9), Triv: & \mathcal{C}_{10} : [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge num\ X) \wedge X > 100)]^F \\
Res(\mathcal{C}_8,\mathcal{C}_{10}), Triv: & \square
\end{array}$$

The following $\mathcal{ERUE}$-refutation has a more goal directed character and is at least not more complicated than the above paramodulation proof:

$$\begin{array}{ll}
\mathit{Func}'(\mathcal{C}_1): & \mathcal{C}_3 : [(\mathit{odd}\,Y \land \mathit{num}\ Y) = (\neg\ \mathit{ev}\ Y \land \mathit{num}\ Y)]^T \\
\mathit{Func}(\mathcal{C}_2): & \mathcal{C}_4 : [(\forall X.\ n\ X \Rightarrow ((\mathit{odd}\ X \land \mathit{num}\ X) \land X > 100)) \\
 & \qquad = (\forall X.\ n\ X \Rightarrow ((\neg\ \mathit{ev}\ X \land \mathit{num}\ X) \land X > 100)))]^F \\
\mathit{Dec}(\mathcal{C}_4), \mathit{Triv}: & \mathcal{C}_5 : [(\lambda X.\ n\ X \Rightarrow ((\mathit{odd}\ X \land \mathit{num}\ X) \land X > 100)) \\
 & \qquad = (\lambda X.\ n\ X \Rightarrow ((\neg\ \mathit{ev}\ X \land \mathit{num}\ X) \land X > 100)))]^F \\
\mathit{Func}(\mathcal{C}_5): & \mathcal{C}_6 : [(n\ s \Rightarrow ((\mathit{odd}\ s \land \mathit{num}\ s) \land s > 100)) \\
 & \qquad = (n\ s \Rightarrow ((\neg\ \mathit{ev}\ s \land \mathit{num}\ s) \land s > 100)))]^F \\
2 \times \mathit{Dec}(\mathcal{C}_6), 4 \times \mathit{Triv}: & \mathcal{C}_7 : [(\mathit{odd}\ s \land \mathit{num}\ s) = (\neg\ \mathit{ev}\ s \land \mathit{num}\ s)]^F \\
\mathit{Res}(\mathcal{C}_7, \mathcal{C}_3), \mathcal{UNI}: & \Box
\end{array}$$

If we slightly modify, i.e., complicate, the example by switching the first two conjuncts in the definition of odd numerals, such that $\mathcal{C}_1 : [\lambda X.\ (\mathit{num}\ X \land \mathit{odd}\ X) = \lambda X.\ (\neg\ \mathit{ev}\ X \land \mathit{num}\ X)]^T$ and $\mathcal{C}_3 : [(\mathit{odd}\ X \land \mathit{num}\ X) = (\neg\ \mathit{ev}\ X \land \mathit{num}\ X)]^T$, both refutations obviously need to employ additional recursive calls from within unification in order to show that $[(\mathit{num}\ X \land \mathit{odd}\ X) = (\mathit{odd}\ X \land \mathit{num}\ X)]^F$ is a solvable extensional unification problem. More precisely in an analogous paramodulation refutation to above additional recursive calls are necessary to justify the paramodulation steps leading to $\mathcal{C}_8$ and $\mathcal{C}_{10}$ (note that *Triv* is not applicable in both cases to justify the resolution step immediately). These clauses would look like

$$\begin{array}{ll}
\mathcal{C}_8 : & [\forall X.\ n\ X \Rightarrow ((\neg\ \mathit{ev}\ X \land \mathit{num}\ X) \land X > 100)]^T \lor \\
 & [(\mathit{num}\ X \land \mathit{odd}\ X) = (\mathit{odd}\ X \land \mathit{num}\ X)]^F
\end{array}$$

and

$$\begin{array}{ll}
\mathcal{C}_{10} : & [\forall X.\ n\ X \Rightarrow ((\neg\ \mathit{ev}\ X \land \mathit{num}\ X) \land X > 100)]^F \lor \\
 & [(\mathit{num}\ X \land \mathit{odd}\ X) = (\mathit{odd}\ X \land \mathit{num}\ X)]^F
\end{array}$$

In both cases the unification constraints can be eliminated only by recursive calls to overall proof procedure.

A RUE-resolution refutation analogous to the above one would result in $\mathit{Res}(\mathcal{C}_5, \mathcal{C}_3)$: $\mathcal{C}_7 : [(\mathit{odd}\ s \land \mathit{num}\ s) = (\neg\ \mathit{ev}\ s \land \mathit{num}\ s) = (\mathit{num}\ s \land \mathit{odd}\ s) = (\neg\ \mathit{ev}\ s \land \mathit{num}\ s)]^F$ such that decomposition and subsequent elimination of trivial pairs leads to $\mathcal{C}_8 : [(\mathit{num}\ X \land \mathit{odd}\ X) = (\mathit{odd}\ X \land \mathit{num}\ X)]^F$. Then an recursive call to the overall proof procedure leads to the refutation.

This modified example demonstrates that calculus $\mathcal{EP}$ in some cases unavoidably needs to perform recursive calls to the overall refutation process in order to justify single paramodulation steps. These side computations obviously follow the difference reduction idea. Thus, in practice the paramodulation approach requires both: appropriate heuristics in order to guide a proof along the paramodulation, i.e., term rewriting idea, and appropriate heuristics that guide the side computations based on the difference reduction idea.

The following example demonstrates that in some situations the term rewriting idea even seems to be completely inappropriate and useless. Therefore we further modify and complicate our example above. The idea is to restructure the single conjuncts in the assertion, such that rewriting with the assumtion equation at the *right* subterm of the assertion becomes impossible. This final modification additionally illustrates that an efficient paramodulation approach is probably not easy to mechanise in practice, completely independent from the question if suitable reduction orderings are available or not.

$\mathbf{E}_2^{set}$ We complicate problem $\mathbf{E}_1^{set}$ by switching the conjuncts in the assertion:

$$\begin{aligned}&\{X|\ odd\ X \wedge num\ X\} = \{X|\ \neg\ ev\ X \wedge num\ X\} \Rightarrow\\ &\quad \wp\{X|\ (odd\ X \wedge X > 100) \wedge num\ X\} = \wp\{X|\ (\neg\ ev\ X \wedge X > 100) \wedge num\ X\}\end{aligned}$$

Negation and clause normalisation leads to:

$$\begin{aligned}&\mathcal{C}_1 : [\lambda X.\ (odd\ X \wedge num\ X) = \lambda X.\ (\neg\ ev\ X \wedge num\ X)]^T\\ &\mathcal{C}_2 : [(\lambda N.\ \forall X.\ N\ X \Rightarrow ((odd\ X \wedge X > 100) \wedge num\ X))\\ &\quad = (\lambda N.\ \forall X.\ N\ X \Rightarrow ((\neg\ ev\ X \wedge X > 100) \wedge num\ X))]^F\end{aligned}$$

We first try to proceed analogous to example $\mathbf{E}_1^{set}$:

$$\begin{aligned}Func'(\mathcal{C}_1) :&\quad \mathcal{C}_3 : [(odd\ X \wedge num\ X) = (\neg\ ev\ X \wedge num\ X)]^T\\ Func(\mathcal{C}_2) :&\quad \mathcal{C}_4 : [(\forall X.\ n\ X \Rightarrow ((odd\ s \wedge s > 100) \wedge num\ s)) =\\ &\qquad (\forall X.\ n\ X \Rightarrow ((\neg\ ev\ s \wedge s > 100) \wedge num\ s))]^F\end{aligned}$$

Followowing the paramodulation based refutation for $\mathbf{E}_1^{set}$ we perform a recursive call to the refutation process with rule *Equiv* and then try to rewrite the resulting clauses in an appropriate way:

$$\begin{aligned}Equiv(\mathcal{C}_4) :&\quad \mathcal{C}_5 : [\forall X.\ n\ X \Rightarrow ((odd\ X \wedge X > 100) \wedge num\ X)]^T \vee\\ &\qquad [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge X > 100) \wedge num\ X)]^T\\ &\quad \mathcal{C}_6 : [\forall X.\ n\ X \Rightarrow ((odd\ X \wedge X > 100) \wedge num\ X)]^F \vee\\ &\qquad [\forall X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge X > 100) \wedge num\ X)]^F\end{aligned}$$

Surely, there are many different rewrite steps possible with $\mathcal{C}_3$ on these clauses, but as the reader may convince himself, none of these rewrite steps lead to unification constraints that are solvable (even recursive calls to the overall proof procedure cannot help and unfortunately only overwhelm the search space with useless clauses). The problem is that the *right* paramodulation step is simple not possible because of the improperly modified term-structure. The only possible way to proceed that the author explored is to employ a difference reducing refutation as motivated by the respective $\mathcal{ERUE}$ refutation in example $\mathbf{E}_1^{set}$ above:

$$\begin{aligned}Dec(\mathcal{C}_4), Triv :&\quad \mathcal{C}_5 : [(\lambda X.\ n\ X \Rightarrow ((odd\ X \wedge X > 100) \wedge num\ X))\\ &\qquad = (\lambda X.\ n\ X \Rightarrow ((\neg\ ev\ X \wedge X > 100) \wedge num\ X)))]^F\\ Func(\mathcal{C}_5) :&\quad \mathcal{C}_6 : [(n\ s \Rightarrow ((odd\ s \wedge s > 100) \wedge num\ s))\\ &\qquad = (n\ s \Rightarrow ((\neg\ ev\ s \wedge s > 100) \wedge num\ s)))]^F\\ Dec(\mathcal{C}_6), Triv :&\quad \mathcal{C}_7 : [((odd\ s \wedge s > 100) \wedge num\ s) = ((\neg\ ev\ s \wedge s > 100) \wedge num\ s)]^F\end{aligned}$$

Instead of immediately resolving beetween $\mathcal{C}_7$ and $\mathcal{C}_3$ as in the corresponding derivation for $\mathbf{E}_1^{set}$ we employ the extensionality rules to both clauses and proceed with a straightforward resolution proof.

$$
\begin{array}{lll}
Equiv(\mathcal{C}_7)\mathcal{CNF}, & & \\
Fac, \mathcal{UNI}: & \mathcal{C}_8 : [odd\ s]^T \lor [ev\ s]^F & \\
 & \mathcal{C}_9 : [odd\ s]^T \lor [s > 100]^T & (subsumed\ by\ \mathcal{C}_{12}) \\
 & \mathcal{C}_{10} : [odd\ s]^T \lor [num\ s]^T & (subsumed\ by\ \mathcal{C}_{16}) \\
 & \mathcal{C}_{11} : [s > 100]^T \lor [ev\ s]^F & (subsumed\ by\ \mathcal{C}_{12}) \\
 & \mathcal{C}_{12} : [s > 100]^T & \\
 & \mathcal{C}_{13} : [s > 100]^T \lor [num\ s]^T & (subsumed\ by\ \mathcal{C}_{12}) \\
 & \mathcal{C}_{14} : [num\ s]^T \lor [ev\ s]^F & (subsumed\ by\ \mathcal{C}_{16}) \\
 & \mathcal{C}_{15} : [num\ s]^T \lor [s > 100]^T & (subsumed\ by\ \mathcal{C}_{16}) \\
 & \mathcal{C}_{16} : [num\ s]^T & \\
 & \mathcal{C}_{17} : [odd\ s]^F \lor [s > 100]^F \lor [num\ s]^F \lor [ev\ s]^T &
\end{array}
$$

Straightforward resolution refutation with $\mathcal{C}_8, \mathcal{C}_{21}, \mathcal{C}_{22}, \mathcal{C}_{23}$

An alternative Refutation in RUE-resolution approach that illustrates the difference reduction idea even better is the following:

$$
\begin{array}{ll}
Res(\mathcal{C}_3, \mathcal{C}_7): & \mathcal{C}_5 : [((odd\ X \land num\ X) = (\neg\ ev\ X \land num\ X)) = \\
 & \quad (((odd\ s \land s > 100) \land num\ s) = ((\neg\ ev\ s \land s > 100) \land num\ s)]^F
\end{array}
$$

$Equiv(\mathcal{C}_5), Cnf:$

$$
\begin{array}{l}
\mathcal{C}_7 : [(odd\ X \land num\ X) = (\neg\ ev\ X \land num\ X)]^T \lor \\
\qquad [((odd\ s \land s > 100) \land num\ s) = ((\neg\ ev\ s \land s > 100) \land num\ s)]^T \\
\mathcal{C}_8 : [(odd\ X \land num\ X) = (\neg\ ev\ X \land num\ X)]^F \lor \\
\qquad [((odd\ s \land s > 100) \land num\ s) = ((\neg\ ev\ s \land s > 100) \land num\ s)]^F
\end{array}
$$

$2 \times Equiv'(\mathcal{C}_7), Cnf, Fac, Triv:$

$$
\begin{array}{l}
\mathcal{C}_9 : [odd\ s]^T \lor [ev\ s]^T \lor [odd\ X]^T \lor [ev\ X]^T \lor [num\ s]^F \lor [s > 100]^F \lor [num\ X]^F \\
\mathcal{C}_{10} : [odd\ X]^T \lor [ev\ X]^T \lor [ev\ s]^F \lor [num\ s]^F \lor [s > 100]^F \lor [odd\ s]^F \lor [num\ X]^F \\
\mathcal{C}_{11} : [odd\ s]^T \lor [ev\ s]^T \lor [num\ s]^F \lor [s > 100]^F \lor [ev\ X]^F \lor [num\ X]^F \lor [odd\ X]^F \\
\mathcal{C}_{12} : [ev\ s]^F \lor [num\ s]^F \lor [s > 100]^F \lor [odd\ s]^F \lor [ev\ X]^F \lor [num\ X]^F \lor [odd\ X]^F
\end{array}
$$

$2 \times Equiv'(\mathcal{C}_7), Cnf, Fac, Triv:$

$$
\begin{array}{l}
\mathcal{C}_{13} : [num\ s]^T \lor [num\ X]^T \\
\mathcal{C}_{14} : [s > 100]^T \lor [num\ X]^T \\
\mathcal{C}_{15} : [odd\ s]^T \lor [num\ X]^T \lor [ev\ s]^F \\
\mathcal{C}_{16} : [s > 100]^T \lor [odd\ X]^T \lor [ev\ X]^F \\
\mathcal{C}_{17} : [odd\ s]^T \lor [odd\ X]^T \lor [ev\ s]^F \lor [ev\ X]^F \\
\mathcal{C}_{18} : [s > 100]^T \lor [ev\ X]^T \lor [odd\ X]^F \lor [num\ X]^F \\
\mathcal{C}_{19} : [ev\ s]^T \lor [num\ X]^T \lor [odd\ s]^F \lor [num\ s]^F \lor [s > 100]^F \\
\mathcal{C}_{20} : [ev\ s]^T \lor [odd\ X]^T \lor [odd\ s]^F \lor [num\ s]^F \lor [s > 100]^F \lor [ev\ X]^F \\
\mathcal{C}_{21} : [ev\ s]^T \lor [ev\ X]^T \lor [odd\ s]^F \lor [num\ s]^F \lor [s > 100]^F \lor [odd\ X]^F \lor [num\ X]^F
\end{array}
$$

Straightforward resolution proof on $\mathcal{C}_9, \ldots, \mathcal{C}_{21} : \Box$

We could even resolve immediately between $\mathcal{C}_3$ and $\mathcal{C}_4$ and proceed with a recursive call to the overall proof procedure as illustrated above. The resulting set of clauses then again increases a bit but a straightforward resolution proof is still easy to find.

With this example we demonstrated that a pure termrewriting is most likely impossible in extensional higher-order resolution as a combined extensionality treatment that takes functional and Boolean extensionality into account seems to require the application of difference reduction techniques.

The following example illustrates that neither rule *Para* nor rule *Para'* capture full functional extanionality.[1]

$\mathbf{E}_1^{func}$ $(\forall X_\iota .\ f_{\iota\to\iota}X = g_{\iota\to\iota}\ X) \Rightarrow (p_{(\iota\to\iota)\to o}\ f \Rightarrow p_{(\iota\to\iota)\to o}\ g)$

This problem normalises to:

$$\mathcal{C}_1 : [f\ X = g\ X]^T \quad \mathcal{C}_2 : [p\ f]^T \quad \mathcal{C}_3 : [p\ g]^F$$

Note that neither with rule *Para* nor with rule *Para'* a fruitful rewriting step is possible (when applying rule *Para'* in order to rewrite f in $\mathcal{C}_2$ we obtain the unification constraint $[P\ (f\ X) = p\ f]^F$, which is not solvable with a projection binding). Instead we have to employ difference reducing derivations and again the $\mathcal{ERUE}$ derivation is more elegant:

Refutation in $\mathcal{EP}$

$$\begin{array}{lll}
Res(\mathcal{C}_2,\mathcal{C}_3), Dec: & \mathcal{C}_4 : [f = g]^T & \\
Func(\mathcal{C}_4): & \mathcal{C}_5 : [f\ s_\iota = g\ s_\iota]^T & \\
Leib(\mathcal{C}_5): & \mathcal{C}_6 : [p\ (f\ s)]^T & \mathcal{C}_7 : [p\ (g\ s)]^F \\
Para(\mathcal{C}_1,\mathcal{C}_6), \mathcal{UNI}: & \mathcal{C}_8 : [p\ (g\ s)]^T & \\
Res(\mathcal{C}_8,\mathcal{C}_7), Triv: & \Box &
\end{array}$$

Refutation in $\mathcal{ERUE}$

$$\begin{array}{ll}
Res(\mathcal{C}_2,\mathcal{C}_3), Dec: & \mathcal{C}_4 : [f = g]^T \\
Func(\mathcal{C}_4): & \mathcal{C}_5 : [f\ s_\iota = g\ s_\iota]^T \\
Res(\mathcal{C}_5,\mathcal{C}_1), \mathcal{UNI}: & \Box
\end{array}$$

8.7 Raised Questions

The examples examined so far (an excerpt has been presented in this Chapter) raise various questions concerning the calculi $\mathcal{ER}$, $\mathcal{EP}$, and $\mathcal{ERUE}$:

1. Is rule *FlexFlex* indeed admissible in all three approaches, i.e. can this rule be avoided (as already realised in LEO)? As to the best of my knowledge there is no counterexample to this conjecture.

2. Can rule *Leib* be restricted to base types as conjectured in Remark 4.18 and as already employed in LEO's extensionality treatment (cf. step 7 in LEO's main loop as presented in Subsection 7.2.3)? For the examples examined so far, this restriction is indeed possible without affecting refutability.

3. Is decomposition rule *Dec*, as used in this thesis, or rule *Dec'*, as used in [BK98a], better suited for applications in practice? The difference between both rules has been illustrated in Subsection 8.2.

[1] It has been claimed by an unknown referee of [Ben98], that rule *Para'* in contrast to rule *Para* captures full functional extensionality, which is not the case as our examples illustrates.

4. Is the traditional paramodulation rule *Para* (cf. Figure 5.1) or its higher-order counterpart *Para'* (cf. Figure 5.1) better suited in applications of calculus $\mathcal{EP}$ in practice. The connection between both rules has been illustrated in Section 5.1.

5. In 5.2 it has been remarked that paramodulation into unification constraints may shorten some proofs, but it might be quite complicated to guide this in practice. This question needs further investigation.

6. The following example demonstrates that in calculus $\mathcal{ERUE}$ various applications of rule *Leib* can be substituted by alternative derivations applying resolution on unification constraints instead.

   Remark 8.1. In $\mathcal{ERUE}$ one can prove example $\mathbf{E}_5^{ext}$ (cf. Section 8.1), which formulates an instance of the functional extensionality property for Leibniz equality, as follows (the idea of this refutation is due to Frank Pfenning):

$$\begin{array}{ll}
Cnf(\neg\mathbf{E}_5^{ext}): & \mathcal{C}_1 : [P\ (m\ X)]^F \vee [P\ (n\ X)]^T \\
 & \mathcal{C}_2 : [q\ m]^T \\
 & \mathcal{C}_3 : [q\ n]^F \\
Res(\mathcal{C}_2, \mathcal{C}_3), Dec, Triv: & \mathcal{C}_4 : [m = n]^F \\
Func(\mathcal{C}_4): & \mathcal{C}_5 : [m\ s = n\ s]^F \\
Res(\mathcal{C}_1, \mathcal{C}_5): & \mathcal{C}_6 : [P\ (m\ X)]^F \vee [P\ (n\ X) = ((m\ s) = (n\ s))]^F \\
\mathcal{UNI}(\mathcal{C}_6, [\lambda X.\ (m\ s) = X/P]): & \square
\end{array}$$

   Hence the question arises if rule *Leib* can be avoided completely in $\mathcal{ERUE}$. As rule *Leib* is obviously not avoidable in calculus $\mathcal{EP}$ (there is no proof in $\mathcal{EP}$ for the above problem that avoids the application of rule *Leib*) this result would be another strong argument in favour of calculus $\mathcal{ERUE}$ and against calculus $\mathcal{EP}$.

7. The calculi do not provide sufficient restrictions of the extensionality and equality rules that allow for a fruitful guidance of the proof search in difficult examples. Thus, further investigations in order to restrict and specialise the developed approaches is important and can probably lead to much more powerful implementations as the Leo-system.

8. It should be examined whether $\mathcal{EP}$ and $\mathcal{ERUE}$ bring essential advantages at all over a pure defined equality treatment as employed in $\mathcal{ER}$. Clearly, one advantage of $\mathcal{EP}$ and $\mathcal{ERUE}$ is that the primitive substitution rule in these approaches allows to immediately instantiate a flexible literal by a primitive equation, whereas in calculus $\mathcal{ER}$ the instantiation of Leibniz equality requires two subsequent primitive substitution steps (which can be improved by extending the primitive substitution rule in $\mathcal{ER}$, such that it also instantiates the free literal head by most general Leibniz equations, i.e., $\lambda\overline{X^n}.\ \forall P_{\alpha\to o}.\ \neg(P\ (H_1\ \overline{X^n})) \vee (P\ (H_2\ \overline{X^n}))$, where H_1 and H_2 are new free variables of appropriate type). The importance of primitive substitutions of equality is demonstrated by the Tps-example THM15b discussed in [ABI+96].

Chapter 9

Applications and Related Work

In this chapter we discuss related work and sketch some applications. The discussion is structured with respect to the different kinds of contributions.

9.1 Cooperation and Joint Work

The contributions of the Chapters 2.1 and 3 have been developed in cooperation with Michael Kohlhase. Some main aspects of this chapter were also positively influenced by discussions with Frank Pfenning and Peter Andrews during a research stay at Carnegie Mellon University, Pittsburgh, USA.

As already mentioned, the resolution calculus $\mathcal{ER}$ in Chapter 4 is based on, extends, and corrects Michael Kohlhase's approaches presented in [Koh94b] and [Koh95]. And the first completeness proof for $\mathcal{ER}$, which is presented in [BK98a] and which differs from the one presented in this thesis, has been developed in cooperation with Michael Kohlhase. The author gained important insights about $\mathcal{ER}$, the completeness proof for $\mathcal{ER}$ and the ideas for $\mathcal{ERUE}$ in discussions with Frank Pfenning.

9.2 Abstract Consistency and Model Existence

The abstract consistency method, which has been extended in this thesis, was first developed by Hintikka and Smullyan [Hin55, Smu63, Smu68] for first-order logic and extended to higher-order logic by Andrews [And71]. The new results can clarify the relationship between syntax and semantics for a variety of higher-order deduction calculi. Up to now calculus development in higher-order logic has been guided by Andrew's *Unifying Principle for Type Theory* [And71]. This model existence theorem has set the completeness standard for higher-order calculi such as [And71, Hue72, Hue73a, ALCMP84], even though it is weaker (i.e., with respect to this principle less formulae are valid) than the intuitive one given by Henkin Models. The new model existence theorems, e.g., allow to analyse completeness of the traditional higher-order resolution calculi [And71, JP72, Hue72, Hue73a] or the more recent ones [Wol93, Koh94b, Koh95] with respect to Henkin semantics or, if the extensionality axioms in these approaches were avoided, with respect to the Σ-Models. Both was hardly possible before, i.e., only when using direct proofs and without the help of a proof tool.

The semantical notions in Chapter 2.1 stem from earlier attempts of Kohlhase to achieve completeness with respect to Henkin models for higher-order tableaux [Koh95, Koh98a] and especially from the attempts of the author in cooperation with Kohlhase to achieve completeness for higher-order resolution [Koh94a, Ben97, BK97b, BK98a].

A model existence theorem for a logical system $\mathcal{L}$ is a theorem of the form: *If a set of sentences Φ in $\mathcal{L}$ is a member of an abstract consistency class Γ, then there exists an $\mathcal{L}$-model for Φ*. Thus if we want to show the completeness of a particular calculus $\mathcal{L}$, we first prove that the class Γ of sets of sentences Φ that are $\mathcal{L}$-consistent (cannot be refuted in $\mathcal{L}$) is an abstract consistency class, then the model existence theorem tells us that $\mathcal{L}$-consistent sets of sentences are satisfiable in $\mathcal{L}$. Now we assume that a sentence $\mathbf{A}$ is valid in $\mathcal{L}$, so $\neg\mathbf{A}$ does not have a $\mathcal{L}$-model and is therefore $\mathcal{L}$-inconsistent. From this it is easy to verify that $\mathbf{A}$ is a theorem of $\mathcal{L}$. Note that with this argumentation the completeness proof for $\mathcal{L}$ condenses to verifying that Γ is an abstract consistency class, a task that does not refer to $\mathcal{L}$-models. Thus the usefulness of model existence theorems derives from the fact that it replaces the model-theoretic analysis in completeness proofs with the verification of some proof-theoretic conditions (membership in Γ). In this respect a model existence theorem is similar to a Herbrand Theorem, but it is easier to generalise to other logic systems like higher-order logic.

Another application of model existence theorems is that they allow for very simple (but non-constructive) proofs of cut-elimination theorems. In [And71] Andrews applies his *Unifying Principle* to cut-elimination in a non-extensional sequent calculus, by proving the calculus complete (relative to $\mathfrak{T}$) both with and without the cut rule and concludes that cut-elimination is valid for this calculus. In the extensional case, where a cut-elimination theorem can be found in [Tak68, Tak87], we can directly model a cut-elimination proof following Andrews' approach, using the model existence theorem for Henkin models.

Takahashi discusses in [Tak68] a non-constructive cut-elimination theorem for simple type theory with extensionality. This work is based on [Tak67], where a respective result has been shown for simple type theory without extensionality. Cut-elimination addresses the question: "Given a deductive system, a proof for $\mathbf{A} \Rightarrow \mathbf{B}$ and one for $\mathbf{B} \Rightarrow \mathbf{C}$. How can we construct a proof for $\mathbf{A} \Rightarrow \mathbf{B}$?" Takahashi analyses this problem for a Gentzen style sequent calculus [Gen35]. His system is also used by Takeuti in [Tak87]. A constructive proof of cut-elimination for intuitionistic type theory is presented by a student of Gandy in [Unkar].

With respect to the connection of the primitive substitution principle and higher-order model existence the author wants to point to the theorem 2 in [BK98a]. A consequence of this restricted model existence theorem would be that depending on the maximal order τ of an input problem the primitive substitution of universal quantifiers at head positions could be restricted to those types with an order $\leq \tau$. Unfortunately at CADE-15 Peter Andrews presented a counterexample to the argumentation in the proof of theorem 2 given in [BK98a] and [BK97a] so that we had to withdraw the theorem. The author is grateful to Peter Andrews for his advice (and not only for this particular one) but also wants to point out that the lack of theorem 2 does not affect the other results presented in [BK98a] as they do not depend on theorem 2. The overall role of the primitive substitution principle in higher-order theorem proving and the question whether there are ways to restrict this principle is, at least for the author, still not obvious.

In all these applications, the leverage added by the work presented in Chapters 2.1 and 3 is that we can now extend non-extensional results to extensional cases. However, the generalised model classes have a merit of their own, for instance in higher-order logic programming [NM94], where the denotational semantics of programs can induce non-standard meanings for the classical connectives. For instance, given a SLD-like search strategy as in λ-PROLOG [Mil91], conjunction is not commutative any more. Therefore, various authors have proposed model-theoretic semantics, where property $\mathfrak{b}$ (which expresses that the domain D_o contains exactly $\mathtt{T}$ and $\mathtt{F}$) fails. For instance Wolfram uses Andrews v-complexes [Wol94] as a semantics for λ-PROLOG and Nadathur uses "labeled structures" for the same purpose in [NM94]. It is plausible to assume that the results of this thesis will be useful for further developments in this direction as well.

9.3 Extensional Higher-Order Resolution

Higher-order resolution has been first discussed in [And71]. Andrews calculus still avoids higher-order unification and instead provides a *substitution* rule, i.e., a instantiation rule for free variables, that basically allows to enumerate the Herbrand universe. Most important about [And71] is, that it adapts *Smullyans unifying principle* to higher-order logic in order to prove completeness of the developed resolution approach with respect to the semantical notion of v-complexes; see also the discussion in Section 9.2 above.

The first mechanisable higher-order resolution calculi were presented by Jensen/Pietrowski in [JP72] and Huet in [Hue72, Hue73a]. Especially Huet's Constrained Resolution Approach is closely related to the work presented here, as the non-extensional fragment of calculus $\mathcal{ER}$ is essentially a variant of Huet's Constrained Resolution Approach. The differences of extensional higher-order resolution to Huet's constrained resolution approach have been illustrated in this thesis; especially remark 4.17 pointed to the fact, that in contrast to Huet's approach eager unification becomes essential in $\mathcal{ER}$, such that unification can not be delayed until the end of the refutation process.

The resolution calculus $\mathcal{HORES}$ in [Koh94b] influenced the development of calculus $\mathcal{ER}$ to a large extent. The particular connections between $\mathcal{ER}$ and $\mathcal{HORES}$ have been discussed in detail in Section 4.1. Furthermore, extensionality rule *Equiv* is motivated by an analogous rule already given in the higher-order tableaux calculus [Koh95]. Whereas this calculus claims to be Henkin complete it can easily be shown by a counterexample, for instance by examples $\mathbf{E}_4^{ext}$ and $\mathbf{E}_5^{ext}$, that it is not. The problem is that [Koh95] does not realise a sufficient interplay between the functional and the Boolean extensionality principles. Such an interplay is realised in $\mathcal{ER}$ by the modified rule *Func* (which extends the α- and η-unification rules employed in [Koh95]) and the rule *Leib* which connects unification constraints with Leibniz equations in connection with rule *Equiv*. The extensional tableaux calculus $\mathcal{HTE}$ [Koh98a] which extends [Koh95] translates the rules of calculus $\mathcal{ER}$ presented in this thesis in a tableaux setting. In difference to [Koh95] the calculus in [Koh98a] provides an extended unification rule *Func* as discussed in this thesis (the extended unification rule *Func* applies the functional extensionality principles also to non-λ-abstractions within higher-order unification).

Wolfram suggests in his resolution approach [Wol93] to employ general higher-order E-unification, but does not present a concrete higher-order E-unification algorithm for theories E that include full extensionality principles. And it has been illustrated in this thesis, that general calls to a Henkin complete theorem prover are needed to realise extensional higher-order E-unification. In other words, extensional higher-order unification unavoidably needs to employ higher-order theorem proving for side computations, such that the difference between extensional higher-order unification and Henkin complete higher-order theorem proving disappears. Instead, both algorithms have to be closely integrated. In order to reach Henkin completeness the resolution approach in [Wol93] therefore either needs to add the extensionality axioms or has to address the problem for mutual recursive calls from resolution to

E-unification and vice versa.

The difference of higher-order unification as employed in $\mathcal{ER}$ with respect to the higher-order E-unification approach discussed in [Sny90] is, that the latter is restricted to first-order theories only and does not take the Boolean extensionality property into account. Hence, it does not allow to unify terms like $\lambda X.\ red\ X \wedge circle\ X$ and $\lambda X.\ circle\ X \wedge red\ X$, whereas $\mathcal{ER}$ allows to tests for the unifiability of two terms with respect to an arbitrary higher-order theory including both extensionality principles. Restricted transformation based approaches to higher-order E-unification, where as much computation as possible is pushed to a first-order E-unification procedure, are discussed in [QW96, NQ91], and a restricted combinatory logic approach is presented in [DJ92]. Like [Sny90], these approaches do not take the Boolean extensionality into account and do not allow to consider arbitrary higher-order theories. The study of higher-order E-unification was first suggested by Siekmann in [Sie84] and a complete overview on first-order E-unification is provided by [BS94].

The "theorem proving modulo" approach described in [DHK98] is a way to remove computational arguments from proofs by reasoning modulo a congruence on propositions that is handled by via rewrite rules and equations. In their paper the authors present a higher-order logic as a theory modulo.[1]

Pfenning addresses in [Pfe87] the problem of translating machine found proofs in higher-order logic into natural deduction [Pra65] or sequent style calculi [Gen35]. The proposed solution, which is based on previous works by Andrews [And80] and Miller [Mil83], first translates the machine found proof into a so-called expansion tree proof, where it can be cleaned up and brought into a certain standard form, before it is translated into a natural deduction or sequent calculus proof. Pfenning's work especially focuses on the problem of extensionality and primitive equality and it extends Miller's work [Mil83] by introducing the notion of extensional expansion proofs. The proof translation mechanism developed by Pfenning is employed in the TPS-System in order to translate mating proofs [And76] into natural deduction proofs and it is applicable to other machine oriented calculi as well. The only prerequisite is a translation from the deductive system used by the theorem prover into extensional expansion proofs. The relation to the calculi sketched in this thesis is thus quite obvious: By defining a respective translation algorithm from extensional higher-order resolution into extensional expansion proofs, one can exploit Pfenning's translation mechanism to generate natural deduction proofs.

The role and importance of extensionality in intensional type theory [Str93] is examined by Hofmann in [Hof97]. He illustrates that the identity type in intensional type theory is not powerful enough for the formalisation of mathematics or program development as it lacks the principles of extensionality. Hofmann investigates to what extent extensional constructs of interest (i.e., quotient types) can be added to intensional type theory without sacrificing decidability and existence of canonical forms. He also points to the problem of adding full extensionality to intensional type theory (which is then called extensional type theory) by proving the undecidability of typing in the resulting system. A new idea to extensional equality in intensional type theory is presented in [Alt99].

The treatment of the functional extensionality principles in term rewriting in intuitionistic type theory is furthermore addressed in [DCK93, DCK96].

Typical applications of extensional higher-order resolution (as well as extensional higher-

[1] The author does not fully see yet to what extend the "theorem proving modulo" approach can handle full extensionality in the sense of this thesis.

order paramodulation or RUE-resolution) are comparisons of sets, functions or functionals. Both in mathematics as well as in applications like the verification of functional programs these concepts play an important role. Hence, a subsystem that is capable of at least some basic extensionality reasoning is very important for a mathematical assistant system like ΩMEGA [BCF+97] or a system employed in program verification like PVS [ORS92] or HOL [GM93].

A rather new but nevertheless interesting and challenging application of extensional higher-order resolution lies in the semantical construction in natural language processing. In this context the use of higher-order unification and especially coloured higher-order unification [HK99, HK97], for instance in the dissolving of ellipsis, has been illustrated in [DSP91, GK96a, GK96b, GK97]. The gain of full extensional higher-order theorem proving in this context is motivated by the examples discussed in [KK98b].

9.4 Primitive Equality in Higher-Order Theorem Proving

Equality is usually treated as a defined notion in approaches and systems for automated higher-order theorem proving. This is probably the main reason why the problem of mechanising primitive equality in higher-order logic while preserving Henkin completeness has (to the best knowledge of the author) not been addressed in literature so far.

In contrast, the field of higher-order term rewriting and narrowing has become very active in recent years (see [Pre98, NP98, NM98b, Nip95, Pre95, Pre94, vO94, Wol93]).

The probably most challenging tasks in this field addresses the development of confluent and terminating term rewriting orders (see for instance [JR99, JR98, LP95]) For a summary to the most recent approaches and developments we refer to [Pre98]. Higher-order term rewriting and narrowing is obviously of great importance for the mechanisation of primitive equality in automated higher-order theorem proving, too. But one should not confuse higher-order term rewriting — where the Boolean extensionality property does simply not occur — with higher-order equational reasoning. The difference has been illustrated in detail in Chapter 5, where we show that if one is interested in problem domains which require a combined application of functional and Boolean extensionality principles (e.g., when reasoning on sets described by characteristic functions), then traditional higher-order term rewriting techniques alone cannot be sufficient. And even worse, it has been motivated by the example in Subsection 8.6 that it will be very hard, if not impossible, to integrate the Boolean extensionality principle in higher-order term rewriting (and analogously in *syntactical* higher-order narrowing [Pre98] or superposition [Vir95]), thereby safely preserving a pure term rewriting character. Nevertheless, it seems to be promising to integrate term rewriting and simplification techniques into higher-order theorem proving approaches and to apply these techniques in all those problem domains, where the extensionality principles, especially the Boolean extensionality, are not needed and for which suitable rewrite orderings are available. The calculus $\mathcal{EP}$ presented in this thesis provides a primitive equality treatment based on paramodulation and therefore provides a basis to integrate term rewriting techniques into higher-order equational reasoning. In this sense, the developed approaches $\mathcal{EP}$ and $\mathcal{ERUE}$ can probably fruitful influence and support the development of an (full) extensional higher-order term rewriting approach.

The areas of application for the developed approaches to primitive equality $\mathcal{EP}$, and $\mathcal{ERUE}$ are identical to those for extensional higher-order resolution $\mathcal{ER}$: They improve the mechanisation of reasoning on sets, functions or functionals in mathematics or program verification and can support the semantical construction in natural language processing.

Another interesting application of extensional higher-order equality reasoning might be within the field of logic programming [MNPS91, NM98a]. Also in this research field the treatment of (full) extensional equality has to best knowledge of the author not been addressed so far.

9.5 Theorem Provers for Higher-Order Logic

In the last two decades several higher-order theorem proving system have been built, among them are TPS [ABI$^+$96], HOL [GM93], or PVS [ORS92]. Neither one of these systems provides a full extensionality treatment in the sense of this thesis. The TPS-system, which is very likely the most powerful higher-order theorem prover currently available, at least provides a partial solution: it looks for equations between functional terms in input problems and modifies them by initially applying the extensionally principles in an appropriate way.

The ISABELLE-system [Pau94] and the TWELF-system [SP98] (TWELF is the successor of ELF [Pfe91]) provide logical frameworks, i.e., this systems provide logic languages powerful enough to allow for the specification of object logics as well as the specification of respective proof tactics in order to mechanise reasoning at the object logic level. Both system have reached a considerable degree of automation and have been successfully applied in case studies in the area of programming languages and logics, e.g., the formal verification of Java Byte Code Verifier of Java with ISABELLE and the type preservation and value soundness properties of Mini ML with TWELF (see [Pfe96, Pfe]). Whereas the degree of automation in both systems is steadily increasing, both systems do, to best knowledge of the author, currently not address the problem of mechanising full extensionality reasoning.

Quite closely related to LEO is Konrad's extensional higher-order theorem prover HOT [Kon98], which is based on the higher-order tableaux calculus $\mathcal{HTE}$ [Koh98a] that has already been mentioned above. HOT is implemented in OZ [Sb98, SSW94] and can thus exploit concurrency when branching within tableaux extensions. Therefore, and as HOT is based on more efficient data structures, HOT is (at least) on some examples more efficient as LEO (see [Kon98]). It seems to be interesting to reimplement LEO in OZ using the same datastructures and then to compare the tableaux based prover HOT with LEO.

9.6 Examples

The presented examples can support the development of new approaches and systems for extensional higher-order theorem proving, as they point to the weaknesses of current approaches. Especially the examples discussed in Subsection 8.6 emphasize the difference between higher-order term rewriting and higher-order equational reasoning. And it seems to be very interesting to investigate if and how these examples can be tackled in a pure term rewriting approach.

Chapter 10

Conclusion and Outlook

The role of equality and extensionality higher-order automated theorem proving is not well understood, in particular with respect to its potential for a mechanical treatment. To this end we have developed three calculi $\mathcal{ER}$, $\mathcal{EP}$, and $\mathcal{ERUE}$, which improve the current situation. The practicability of calculus $\mathcal{ER}$ has been demonstrated in case studies with the Leo-system.

Future work should be concerned with the calculi $\mathcal{ER}$, $\mathcal{EP}$, and $\mathcal{ERUE}$ as stated in Section 8.7, namely to investigate the role of the *FlexFlex*-rule, to further restrict the application of the extensionality and equality rules, and to compare the three approaches in practical applications in order to find out, whether a primitive treatment of equality (as in $\mathcal{EP}$ and $\mathcal{ERUE}$) yields any advantages over a purely defined equality treatment (as in $\mathcal{ER}$). Furthermore, the link to general higher-order E-unification needs further study.

Although the experiments with the MIZAR set theory were successful, I do not expect that such a classical theorem-proving approach will ever attain the problem solving expertise of human mathematicians. An integration of various heterogeneous reasoners (e.g., an integration of Tps with the extensionality reasoner Leo and first-order provers like Otter as motivated in [BBS99]) is a more promising alley. Clearly, in order to model human problem solving behaviour the crucial task will be to appropriately guide the collaboration of the integrated systems. Two options in this respect are: (i) to guide their collaboration by a proof planner (deliberative approach), and (ii) to realise the integrated systems as autonomous, specialised proof agents and to guide their collaboration on the basis of evaluations and resource allocations in a resource adaptive agent architecture (reactive approach). Also a mix of these two approaches as in the InteRRaP-architecture [Mül96] may be worth while to explore.

We shall concentrate on the latter option in order to realise a cooperation between Tps, Leo, and Otter in Ωmega, thereby exploiting the agent architecture that is already provided by the Ωmega-system (see [FHJ+99, BS98a, BS99]).

Bibliography

[ABI$^+$96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[AH97] Serge Autexier and Dieter Hutter. Equational proof-planning by dynamic abstraction. In Maria P. Bonacina and Ulrich Furbach, editors, *International Workshop on First-Order Theorem Proving - FTP97*, Linz, Austria, 1997. RISC-Linz Report Series No. 97-50.

[AHMS99] Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. Inka 5.0 – a logic voyager. To appear in the Proceedings of the 16th International Conference on Automated Deduction (CADE-16), Trento, Italy, 1999.

[AINP90] Peter B. Andrews, Sunil Issar, Dan Nesmith, and Frank Pfenning. The TPS theorem proving system. In Stickel [Sti90].

[ALCMP84] Peter B. Andrews, Eve Longini-Cohen, Dale Miller, and Frank Pfenning. Automating higher order logics. *Contemp. Math*, 29:169–192, 1984.

[Alt99] Thorsten Altenkirch. Extensional equality in intensional type theory. To appear at the IEEE Symposium on Logic in Computer Science (LICS'99), Trento, July 1999, 1999.

[AMS98] Serge Autexier, Heiko Mantel, and Werner Stephan. Simultaneous quantifier elimination. In Otthein Herzog and Aandreas Günter, editors, *KI-98: Advances in Artificial Intelligence, 22nd Annual German Conference on Artificial Intelligence*, number 1504 in LNAI, Bremen, Germany, 1998. Springer.

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 36(3):414–432, 1971.

[And72a] Peter B. Andrews. General models and extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.

[And72b] Peter B. Andrews. General models descriptions and choice in type theory. *Journal of Symbolic Logic*, 37(2):385–394, 1972.

[And73] Peter B. Andrews, 1973. Letter to James Roger Hindley dated January 22, 1973.

[And76] Peter B. Andrews. Refutations by matings. *IEEE Trans. Comp.*, C-25(8):801–807, 1976.

[And80] Peter B. Andrews. Transforming matings into natural deduction proofs. In Wolfgang Bibel and Robert Kowalski, editors, *Proceedings of the 5th International Conference on Automated Deduction (CADE-5)*, number 87 in LNCS, pages 281–292. Springer, 1980.

[And81] Peter B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, April 1981.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Academic Press, 1986.

[And89] Peter B. Andrews. On Connections and Higher Order Logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[BA98] Matthew Bishop and Peter B. Andrews. Selectively instantiating definitions. In Kirchner and Kirchner [KK98a].

[Bar84] Hendrik P. Barendregt. *The Lambda Calculus – Its Syntax and Semantics.* North Holland, 1984.

[Bar92] Hendrik P. Barendregt. Lambda-calculi with types. In Samson Abramsky, Dov M. Gabbay, and Thomas S. Maibaum, editors, *Handbook of Logic and Computer Science*, volume 2, pages 118–309. Oxford University Press, 1992.

[BBS99] Christoph Benzmüller, Matthew Bishop, and Volker Sorge. Integrating TPS and ΩMEGA. *Journal of Universal Computer Science*, 5(3):188–207, March 1999. Special issue on Integration of Deduction System.

[BCF+97] Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Karsten Konrad, Erica Melis, Andreas Meier, Wolf Schaarschmidt, Jörg Siekmann, and Volker Sorge. ΩMEGA: Towards a mathematical assistant. In McCune [McC97a], pages 252–255.

[Ben97] Christoph Benzmüller. A calculus and a system architecture for extensional higher-order resolution. Research Report 97-198, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh,USA, June 1997.

[Ben98] Christoph Benzmüller. An adaption of paramodulation and RUE-resolution to higher-order logic. SEKI-Report SR-98-07, Fachbereich Informatik, Universität des Saarlandes, 1998.

[Ben99] Christoph Benzmüller. Extensional higher-order paramodulation and RUE-resolution. To appear in the Proceedings of the 16th International Conference on Automated Deduction (CADE-16), Trento, Italy, 1999.

[BF94] Peter Baumgartner and Ulrich Furbach. PROTEIN: A PROver with a Theory Extension INterface. In Bundy [Bun94], pages 769–773.

[BGLS92] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation and superposition. In Kapur [Kap92], pages 66–78.

[BHJB92] Karl Hans Bläsius and editors Hans-Jürgen Bürckert. *Deduktionssysteme, Automatisierung des Logischen Denkens.* R. Oldenbourg Verlag, 2nd edition, 1992.

[Bib83] Wolfgang Bibel. Matings in matrices. *Communications of the ACM*, 26:844–852, 1983.

[BK97a] Christoph Benzmüller and Michael Kohlhase. Model Existence for Higher-Order Logic. SEKI-Report SR-97-09, Fachbereich Informatik, Universität des Saarlandes, 1997. Also submitted to Journal of Symbolic Logic.

[BK97b] Christoph Benzmüller and Michael Kohlhase. Resolution for Henkin Models. SEKI-Report SR-97-10, Fachbereich Informatik, Universität des Saarlandes, 1997.

[BK98a] Christoph Benzmüller and Michael Kohlhase. Extensional higher-order resolution. In Kirchner and Kirchner [KK98a], pages 56–72.

[BK98b] Christoph Benzmüller and Michael Kohlhase. LEO — a higher-order theorem prover. In Kirchner and Kirchner [KK98a], pages 139–144.

[BS94] Franz Baader and Jörg Siekmann. Unification theory. In Dov Gabbay, editor, *Logic in Artificial Intelligence and Logic Programming.* Oxford University Press, 1994.

[BS98a] Christoph Benzmüller and Volker Sorge. A blackboard architecture for guiding interactive proofs. In Giunchiglia [Giu98], pages 102–114.

[BS98b] Christoph Benzmüller and Volker Sorge. Integrating TPS with ΩMEGA. In Jim Grundy and Malcolm Newey, editors, *Theorem Proving in Higher Order Logics: Emerging Trends*, Technical Report 98-08, Department of Computer Science and Computer Science Lab, The Australian National University, pages 1–19, Canberra, Australia, October 1998.

[BS98c] Wolfgang Bibel and Peter Schmitt, editors. *Automated Deduction – A Basis for Applications.* Kluwer, 1998.

[BS99] Christoph Benzmüller and Volker Sorge. Critical agents supporting interactive theorem proving. SEKI-Report SR-99-02, Fachbereich Informatik, Universität des Saarlandes, 1999.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the 9th Conference on Automated Deduction (CADE-9)*, number 310 in LNCS, pages 111–120, Argonne, Illinois, USA, 1988. Springer.

[Bun94] Alan Bundy, editor. *Proceedings of the 12th Conference on Automated Deduction (CADE-12)*, number 814 in LNAI, Nancy, France, 1994. Springer.

[Byl89] Czeslaw Bylinski. Basic properties of sets. *Journal of Formalized Mathematics*, 1, 1989.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[CrS98] Lassaad Cheikhrouhou and Jörg Siekmann. Planning diagonalization proofs. In Giunchiglia [Giu98], pages 167–180.

[Dah97] Ingo Dahn. Integration of automated and interactive theorem proving in ILF. In McCune [McC97a], pages 57–60.

[Dar71] J. L. Darlington. Deductive plan formation in higher-order logic. *Machine Intelligence*, 7:129–137, 1971.

[Dav83] Martin Davis. The prehistory and early history of automated deduction. In Jörg Siekmann and Graham Wrightson, editors, *Automation of Reasoning*, volume 2 Classical Papers on Computational Logic 1967–1970 of *Symbolic Computation*. Springer, 1983.

[DCK93] Roberto Di Cosmo and Delia Kesner. A confluent reduction for the extensional typed λ–calculus with pairs, sums, recursion and terminal object. In Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, editors, *Proceedings of International Conference on Automata, Languages and Programming (ICALP '91)*, volume 700 of *LNCS*, pages 645–656. Springer, 1993.

[DCK96] Roberto Di Cosmo and Delia Kesner. Combining algebraic rewriting, extensional lambda calculi, and fixpoints. *Theoretical Computer Science*, 169(2):201–220, 1996.

[DHK98] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique, April 1998.

[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.

[DJ92] Daniel Dougherty and Patricia Johann. A combinatory logic approach to higher-order E-unification. In Kapur [Kap92], pages 79–93.

[DSP91] Mary Dalrymple, Stuart Shieber, and Fernando Pereira. Ellipsis and higher-order-unification. *Linguistics and Philosophy*, 14:399–452, 1991.

[Ern71] G. W. Ernst. A matching procedure for type theory. Technical report, Case Western Reserve University, 1971.

[FHJ$^+$99] Andreas Franke, Stephan Hess, Christoph Jung, Michael Kohlhase, and Volker Sorge. Agent-Oriented Integration of Distributed Mathematical Services. *Journal of Universal Computer Science*, 5(3):156–187, March 1999. Special issue on Integration of Deduction System.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition, 1996.

[Frä22a] Adolf A. Fränkel. Der Begriff *definit* und die Unabhängigkeit des Auswahlaxioms. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch Mathematische Klasse*, pages 253–257, 1922.

[Frä22b] Adolf A. Fränkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, 86:230–237, 1922.

[Frä28] Adolf A. Fränkel. Zusatz zu vorstehendem Aufsatz Herrn v. Neumanns. *Mathematische Annalen*, 99:392–393, 1928.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen I & II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[Giu98] Fausto Giunchiglia, editor. *Proceedings of the of the 8th International Conference (AIMSA'98)*, number 1480 in LNAI, Sozopol, Bulgaria, 1998. Springer.

[GJ98] Sambin Giovannin and Smith Jan. *Twenty-five Years of Constructive Type Theory.* Oxford University Press, 1998.

[GK96a] Claire Gardent and Michael Kohlhase. Focus and higher–order unification. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, 1996.

[GK96b] Claire Gardent and Michael Kohlhase. Higher–order coloured unification and natural language semantics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, 1996. ACL.

[GK97] Claire Gardent and Michael Kohlhase. Computing parallelism in discourse. In *Proceedings of IJCAI '97*, pages 1016–1021, Tokyo, 1997.

[GLMS94] Christoph Goller, Reinhold Letz, Klaus Mayr, and Johann Schumann. SETHEO v3.2: Recent developments. In Bundy [Bun94], pages 778–782.

[GM93] Michael J. C. Gordon and Tom F. Melham. *Introduction to HOL – A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360, 1930. English translation in [vH67].

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte der Mathematischen Physik*, 38:173–198, 1931. English translation in [vH67].

[Göd40] Kurt Gödel. *The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis with the Axioms of Set Theory*, volume 3 of *Annals of Mathematics Studies.* Princeton University Press, Princeton, New Jersey; eighth printing 1970, 1940.

[Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Gou66] William Eben Gould. A matching procedure for ω-order logic. Technical report, Applied Logic Corporation, One Palmer Square, Princeton, NJ, 1966.

[Gra95] Peter Graf. *Term Indexing.* PhD thesis, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, July 1995.

[HBVL97] Thomas Hillenbrand, Arnim Buch, Roland Vogt, and Bernd Löchner. Waldmeister: High-performance equational deduction. *Journal of Automated Reasoning*, 18(2), 1997.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Hen96] Leon Henkin. The discovery of my completeness proofs. *The Bulletin of Symbolic Logic*, 2(2):127–157, 1996.

[Her30] Jaques Herbrand. *Recherches sur la théorie de la démonstration.* PhD thesis, Université de Paris, 1930. Englisch translation in [vH67].

[Hes99] Stephan Hess. Software-Ergonomie in einer Beweisentwicklungsumgebung. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, 1999. Forthcoming.

[Hil04] David Hilbert. Über die Grundlagen der Logik und der Arithmetik. In *Verhandlungen des Dritten Internationalen Mathematiker-Kongress in Heidelberg*, pages 174–185. Teubner, Leibzig, 1904.

[Hil27] David Hilbert. Die Grundlagen der Mathematik. In *Abhandlungen aus dem mathematischen Seminar der Hamburgischen Universität 6*, pages 65–85, 1927.

[Hin55] K. J. J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.

[HK97] Dieter Hutter and Michael Kohlhase. A coloured version of the λ-calculus. In McCune [McC97a], pages 291–305.

[HK99] Dieter Hutter and Michael Kohlhase. A coloured version of the λ-calculus. *Journal of Automated Reasoning*, 1999. Forthcoming.

[HKK+94] 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.

[Hof97] Martin Hofmann. *Extensional Constructs in Intensional Type Theory.* Springer, London, 1997.

[HS86] James R. Hindley and Jonathan P. Seldin. *Introduction to Combinators and Lambda Calculus.* Cambridge University Press, 1986.

[HS96] Dieter Hutter and Claus Sengler. INKA - the next generation. In McRobbie and Slaney [MS96].

[Hue72] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic.* PhD thesis, Case Western Reserve University, 1972.

[Hue73a] Gérard P. Huet. A mechanization of type theory. In Donald E. Walker and Lewis Norton, editors, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence (IJCAI73)*, pages 139–146, 1973.

[Hue73b] Gérard P. Huet. The undecidability of unification in third order logic. *Information and Control*, 22(3):257–267, 1973.

[Hue75] Gérard P. Huet. An unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Jec77] Thomas J. Jech. About the axiom of choice. In Jon Barwise, editor, *Handbook of Mathematical Logic*, pages 345–371. North Holland, Amsterdam, 1977.

[JP72] D. C. Jensen and T. Pietrzykowski. A complete mechanization of (ω)-order type theory. In *Proceedings of the ACM annual Conference*, volume 1, pages 82–92, 1972.

[JR98] Jean-Pierre Jouannaud and Albert Rubio. Rewrite orderings for higher-order terms in η-long β-normal form and the recursive path ordering. *Theoretical Computer Science*, 1998.

[JR99] Jean-Pierre Jouannaud and Albert Rubio. The higher-order recursive path ordering. To appear at the IEEE Symposium on Logic in Computer Science (LICS'99), Trento, July 1999, 1999.

[Kap92] Deepak Kapur, editor. *Proceedings of the 11th Conference on Automated Deduction (CADE-11)*, number 607 in LNCS, Saratoga Spings, NY, USA, 1992. Springer.

[Kau92] Stefan Kaufmann. *Mathematica als Werkzeug: eine Einführung.* Birkhäuser, 1992.

[KK98a] Claude Kirchner and Helene Kirchner, editors. *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in LNAI, Lindau, Germany, 1998. Springer.

[KK98b] Michael Kohlhase and Karsten Konrad. Higher-order automated theorem proving for natural language semantics. Seki Report SR-98-04, Fachbereich Informatik, Universität Saarbrücken, 1998.

[KKS98] Manfred Kerber, Michael Kohlhase, and Volker Sorge. Integrating Computer Algebra Into Proof Planning. *Journal of Automated Reasoning*, 21(3):327–355, 1998.

[Kle97] Lars Klein. Indexing für Terme höherer Stufe. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, 1997.

[Koh93] Michael Kohlhase. A unifying principle for extensional higher-order logic. Technical Report 93–153, Dept. of Mathematics, Carnegie Mellon University, 1993.

[Koh94a] Michael Kohlhase. Higher-order order-sorted resolution. Seki Report SR-94-1, Fachbereich Informatik, Universität des Saarlandes, 1994.

[Koh94b] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. PhD thesis, Fachbereich Informatik, Universität des Saarlandes, 1994.

[Koh95] Michael Kohlhase. Higher-Order Tableaux. In Peter Baumgartner, Rainer Hähnle, and Joachim Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in LNAI, pages 294–309, 1995.

[Koh98a] Michael Kohlhase. Higher-order automated theorem proving. In Bibel and Schmitt [BS98c].

[Koh98b] Michael Kohlhase. Towards mathematical vernacular in a mathematical knowledge base. Seki Report SR-99-??, Fachbereich Informatik, Universität des Saarlandes, 1998.

[Kon98] Karsten Konrad. HOT: An automated theorem prover based on higher-order tableaux. In Jim Grundy and Malcolm Newey, editors, *Proceedings of the 11th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'98)*, number 1479 in LNCS, Canberra, Australia, 1998. Springer.

[LB98] Maxim Lifantsev and Leo Bachmair. An LPO-based termination ordering for higher-order terms without λ-abstraction. In Jim Grundy and Malcolm Newey, editors, *Proceedings of the 11th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'98)*, number 1479 in LNCS, pages 277–293, Canberra, Australia, 1998. Springer.

[LP95] Olav Lysne and Javier Piris. A termination ordering for higher-order rewrite systems. In Jieh Hsiang, editor, *Proceedings of the 6^{th} International Consference on Rewriting Techniques and Applications*, number 914 in LNCS, pages 26–40. Springer, 1995.

[Luc72] Claudio. L. Lucchesi. The undecidability of the unification problem for third order languages. Report CSRR 2059, University of Waterloo, Waterloo, Canada, 1972.

[McC94] William McCune. Otter 3.0 reference manual and guide. Technical Report ANL-94-6, Argonne National Laboratory, Argonne, Illinois 60439, USA, 1994.

[McC97a] William McCune, editor. *Proceedings of the 14th Conference on Automated Deduction*, number 1249 in LNAI, Townsville, Australia, 1997. Springer.

[McC97b] William McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[Mel94] Erica Melis. How mathematicians prove theorems. In *Proc. of the Annual Conference of the Cognitive Science Society*, Atlanta, Georgia U.S.A., 1994.

[Mel95] Erica Melis. A model of analogy-driven proof-plan construction. In Chris S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI95)*, pages 182–189, Montreal, Canada, 1995. Morgan Kaufmann.

[Mel97] Erica Melis. Progress in proof planning: Planning limit theorems automatically. Seki Report SR-97-08, Fachbereich Informatik, Universität des Saarlandes, 1997.

[Mil83] Dale Miller. *Proofs in Higher-Order Logic.* PhD thesis, Carnegie-Mellon University, 1983.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 4(1):497–536, 1991.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.

[ML94] Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1994.

[MNPS91] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andrew Scedrow. Uniform proofs as a foundation for logic programming. *Annals of Pure and Apllied Logic*, 51:125–157, 1991.

[MS96] Michael A. McRobbie and John K. Slaney, editors. *Proceedings of the 13th Conference on Automated Deduction (CADE-13)*, number 1104 in LNAI, New Brunswick, NJ, USA, 1996. Springer.

[Mül96] Jörg P. Müller. *The Design of Intelligent Agents: A Layered Approach.* Number 1177 in LNAI. Springer, December 1996.

[MW97] William McCune and Larry Wos. Otter CADE-13 competition incarnations. *Journal of Automated Reasoning*, 18(2):211–220, 1997. Speacial Issue on the CADE-13 Automated Theorem Proving System Competition.

[Nad87] Gopalan Nadathur. *A Higher-Order Logic as the Basis for Logic Programming.* PhD thesis, University of Pennsylvania, Philadelphia, 1987.

[Neu25] John von Neumann. Eine Axiomatisierung der Mengenlehre. *Journal für die reine und angewandte Mathematik*, 154:219–240, 1925.

[Nip95] Tobias Nipkow. Higher-order rewrite systems. In Jieh Hsiang, editor, *Proceedings of the 6th International Conference on Rewriting Techniques and Applications (RTA-95)*, number 914 in LNCS, pages 256–256, Kaiserslautern, Germany, 1995. Springer.

[NM94] Gopalan Nadathur and Dale Miller. Higher-order logic programming. Technical Report CS-1994-38, Department of Computer Science, Duke University, 1994.

[NM98a] Gopalan Nadathur and Dale Miller. Higher-Order Logic Programming. To appear in the *Handbook of Logic in Artificial Intelligence and Logic Programming*, Dov M. Gabbay, Christopher J. Hogger, and John A. Robinson (eds.), Oxford University Press, 1998.

[NM98b] Tobias Nipkow and Richard Mayr. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.

[NP98] Tobias Nipkow and Christian Prehofer. Higher-order rewriting and equational reasoning. In Bibel [BS98c].

[NQ91] Tobias Nipkow and Zhenyu Qian. Modular higher-order E-unification. In Ronald V. Book, editor, *Proceedings of the* 4^{th} *International Conference on Rewriting Techniques and Applications*, number 488 in LNCS, pages 200–214. Springer, 1991.

[ORS92] Sam Owre, John Rushby, and Natarajan Shankar. PVS: a prototype verification system. In Kapur [Kap92], pages 748–752.

[Pad95] V. Padovani. On equivalence classes of interpolation equations. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Apllications*, number 902 in LNCS. Springer, 1995.

[Pau94] Lawrence C. Paulson. *Isabelle. A Generic Theorem Prover*, volume 828 of *Lecture Notes in Artificial Intelligence LNAI*. Springer, 1994.

[Pfe] Frank Pfenning. Computation and deduction. Unpublished lecture notes, 312 pp. April 1997.

[Pfe87] Frank Pfenning. *Proof Transformations in Higher-Order Logic*. PhD thesis, Carnegie-Mellon University, Pittsburgh Pa., 1987.

[Pfe91] Frank Pfenning. Logic programming in the LF logical framework. In Gérard P. Huet and Gordon D. Plotkin, editors, *Logical Frameworks*. Cambridge University Press, 1991.

[Pfe96] Frank Pfenning. The practice of logical frameworks. In Helene Kirchner, editor, *Proceedings of the Collquium on Trees in Algebra and Programming*, number 1059 in LNCS, pages 119–134. Springer, 1996.

[Pie73] Thomasz Pietrzykowski. A complete mechanization of second-order type theory. *Journal of the Association for Computing Machinery*, 20:333–364, 1973.

[Pra65] Dag Prawitz. *Natural Deduction*. Almquist & Wiksell, 1965.

[Pre94] Christian Prehofer. Higher-order narrowing. In *Logic in Computer Science (LICS '94)*, pages 507–516. IEEE Computer Society Press, 1994.

[Pre95] Christian Prehofer. Solving higher-order equations: From logic to programming. Technical Report TUM-I9508, Institut für Informatik, Technische Universität München (TUM), München, 1995.

[Pre98] Christian Prehofer. *Solving Higher-Order Equations: From Logic to Programming*. Progress in theoretical computer science. Birkhäuser, 1998.

[Qia93] Zhenyu Qian. Linear unification of higher-order patterns. In J.-P. Jouannaud M.-C. Gaudel, editor, *Proceedings of TAPSOFT(CAAP)'93*, number 668 in LNCS, pages 391–405. Springer, 1993.

[QW96] Zhenyu Qian and Kang Wang. Modular higher-order equational preunification. *Journal of Symbolic Computation*, 22:401–424, 1996.

[Red96] Darren Redfern. *The Maple Handbook: Maple V Release 4*. Springer, 1996.

[Rob65] John A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[Rob68] John A. Robinson. New directions in theorem proving. In *Proceedings of IFIP Congress in Information Processing*, volume 68, pages 63–67. North Holland, 1968.

[Rob69] John A. Robinson. Mechanizing higher order logic. *Machine Intelligence*, 4:151–170, 1969.

[RSG98] Julian Richardson, Alan Smaill, and Ian Green. Proof planning in higher-order logic with λClam. In Kirchner and Kirchner [KK98a], pages 129–133.

[Rud92] Piotr Rudnicki. An overview of the mizar project. In *Proceedings of the 1992 Workshop on Types and Proofs as Programs*, pages 311–332, 1992.

[Rus02] Bertrand Russell. Letter to Frege. Printed in [vH67], 1902.

[Rus03] Bertrand Russell. *The principles of mathematics.* Cambridge University Press, Cambridge, England, 1903.

[Rus08] Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, XXX:222–262, 1908.

[RW69] George A. Robinson and Larry Wos. Paramodulation and TP in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.

[Sb98] Programming Systems Laboratory Saarbrücken, 1998. The Oz Webpage: `http://www.ps.uni-sb.de/oz/`.

[Sch60] Kurt Schütte. Semantical and syntactical properties of simple type theory. *Journal of Symbolic Logic*, 25:305–326, 1960.

[Sco67] Dana S. Scott. Existence and description in formal logic. In Schoenmann, editor, *Bertrand Russell: Philosopher of the Century.* Allen and Unwin, 1967.

[SG89] Wayne Snyder and Jean Gallier. Higher-Order Unification Revisited: Complete Sets of Transformations. *Journal of Symbolic Computation*, 8:101–140, 1989.

[SHB$^+$98] Jörg Siekmann, Stephan Hess, Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Michael Kohlhase, Karsten Konrad, Erica Melis, Andreas Meier, and Volker Sorge. LOUI: A Distributed Graphical User Interface for the ΩMEGA Proof System. Proceedings of the International Workshop on User Interfaces for Theorem Provers, 1998.

[SHB$^+$99] Jörg Siekmann, Stephan Hess, Christoph Benzmüller, Lassaad Cheikhrouhou, Armin Fiedler, Helmut Horacek, Michael Kohlhase, Karsten Konrad, Andreas Meier, Erica Melis, Martin Pollet, and Volker Sorge. LOUI: Lovely ΩMEGA User Interface. submitted, 1999.

[Sie84] Jörg Siekmann. Universal unification. In R. E. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction (CADE-7)*, number 170 in LNCS, pages 1–42. Springer, 1984.

[Sko28] Thoralf Skolem. Über die mathematische Logik. *Norsk matematisk tidsskrift 10*, 1928.

[Smu63] Raymond M. Smullyan. A unifying principle for quantification theory. *Proc. Nat. Acad Sciences*, 49:828–832, 1963.

[Smu68] Raymond M. Smullyan. *First-Order Logic.* Springer, 1968.

[Sny90] Wayne Snyder. Higher order E-unification. In Stickel [Sti90], pages 573–578.

[Sny91] Wayne Snyder. *A Proof Theory for General Unification.* Progress in Computer Science and Applied Logic. Birkhäuser, 1991.

[SP98] Carsten Schürmann and Frank Pfenning. Automated theorem proving in a simple meta-logic for LF. In Kirchner and Kirchner [KK98a], pages 286–300.

[SSW94] Christian Schulte, Gert Smolka, and Jörg Würtz. Encapsulated search and constraint programming in Oz. In Alan H. Borning, editor, *Proceedings of the 2^{nd} PPCP*, volume 874 of *LNCS*, pages 134–150, Orcas Island, Washington, USA, May 1994. Springer.

[Ste90] Guy L. Steele. *Common Lisp: The Language, 2nd edition.* Digital Press, Bedford, Massachusetts, 1990.

[Sti90] Mark Stickel, editor. *Proceedings of the 10th Conference on Automated Deduction (CADE-10)*, number 449 in LNCS, Kaiserslautern, Germany, 1990. Springer.

[Str93] Thomas Streicher. *Investigations into intensional type theory.* Unknown Publisher, 1993. Habilitationsschrift.

[Tak53] Gaisi Takeuti. On a generalized logic calculus. *Japan Journal of Mathematics*, 23:39 f., 1953.

[Tak67] Moto-o Takahashi. A proof of cut-elimination in simple type theory. *Journal of the Mathematical Society of Japan*, 19:399–410, 1967.

[Tak68] Moto-o Takahashi. Cut-elimination in simple type theory with extensionality. *Journal of the Mathematical Society of Japan*, 19, 1968.

[Tak87] Gaisi Takeuti. *Proof Theory.* North Holland, 1987.

[Tam98] Tanel Tammet. Towards efficient subsumtion. In Kirchner and Kirchner [KK98a], pages 427–441.

[Try89] Andrzej Trybulec. Tarski grothendieck set theory. *Journal of Formalized Mathematics*, Axiomatics, 1989.

[TS89] Zinaida Trybulec and Halina Swieczkowska. Boolean properties of sets. *Journal of Formalized Mathematics*, 1, 1989.

[Unkar] Unknown. *Unknown Title*. PhD thesis, Worcester College, Unknown Year. I have a manuscript of the thesis without a title page. (the author is a student advised by Professor Gandy).

[vH67] Jean van Heijenoort. *From Frege to Gödel : a source book in mathematical logic 1879-1931*. Source books in the history of the sciences series. Harvard University Press, Cambridge, MA, USA, 3rd printing, 1997 edition, 1967.

[Vir95] Roberto Virga. Higher-order superposition for dependent types. Carnegie mellon university, Carnegie Mellon Univ., Pittsburgh, PA, 1995.

[vO94] V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1994.

[Wei97] Christoph Weidenbach. SPASS: Version 0.49. *Journal of Automated Reasoning*, 18(2):247–252, 1997. Special Issue on the CADE-13 Automated Theorem Proving System Competition.

[WGR96] Christoph Weidenbach, Bernd Gaede, and Georg Rock. Spass & flotter, version 0.42. In McRobbie and Slaney [MS96].

[WNB92] Toby Walsh, Alex Nunes, and Alan Bundy. The use of proof plans to sum series. In Kapur [Kap92], pages 325–339.

[Wol88] Stephen Wolfram. *Mathematica. A System for Doing Mathematics by Computer*. Addison-Wesley, 1988.

[Wol93] David A. Wolfram. *The Clausal Theory of Types*. Cambridge University Press, 1993.

[Wol94] David A. Wolfram. A semantics for λ-PROLOG. *Theoretical Computer Science*, 136(1), 1994.

[WR10] Alfred N. Whitehead and Bertrand Russell. *Principia Mathematica*, volume I. Cambridge University Press, Cambridge, Great Britain; 2nd edition, 1910.

[Zer04] Ernst Zermelo. Beweis, daß jede Menge wohlgeordnet werden kann. *Mathematische Annalen*, 59:514–516, 1904.

[Zer08] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre. I. *Mathematische Annalen*, 65:261–281, 1908.

Authorindex

Altenkirch, Thorsten 130
Andrews, Peter B. viii, ix, x, xi, xv, xix, 2, 3, 4, 5, 6, 8, 9, 10, 11, 16, 22, 27, 29, 30, 32, 36, 37, 51, 52, 61, 93, 95, 102, 107, 110, 115, 126, 127, 128, 129, 130, 132
Autexier, Serge ix, 54, 107

Baader, Franz 101, 130
Bachmair, Leo xvi
Barendregt, Hendrik P. 3, 4, 17
Baumgartner, Peter viii, xviii, 2
Benzmüller, Christoph xi, xiii, xiv, xv, xvii, xviii, xix, 9, 12, 14, 15, 27, 52, 53, 54, 55, 61, 64, 67, 82, 91, 92, 93, 95, 107, 110, 113, 114, 115, 125, 127, 128, 131, 133
Bibel, Wolfgang 2
Bishop, Matthew viii, xix, 2, 6, 8, 11, 32, 93, 95, 102, 107, 110, 126, 132, 133
Bläsius, Karl Hans 94
Buch, Arnim viii, 12
Bundy, Alan ix, 93
Bylinski, Czeslaw xviii, 15, 64, 116

Cheikhrouhou, Lassaad ix, xviii, 93, 107
Cheikhrouhou, Lassaad xviii, 15, 92, 93, 107, 131
Church, Alonzo ix, 3, 16

Dahn, Ingo 104
Dalrymple, Mary 131
Darlington, J. L. 2
Davis, Martin 1
Di Cosmo, Roberto 130
Digricoli, Vincent J. 13, 33, 83
Dougherty, Daniel 130
Dowek, Gilles 130

Ernst, G. W. 2

Fehrer, Detlef xviii, 15, 92, 93, 107, 131
Fiedler, Armin xviii, 15, 92, 93, 107, 131
Fitting, Melvin 6, 94
Franke, Andreas 133
Fränkel, Adolf A. xii, 3, 5

Furbach, Ulrich viii, xviii, 2

Gaede, Bernd viii, xviii, 2
Gallier, Jean xiv, 4, 16, 18, 49, 50, 55, 56, 93
Ganzinger, Harald xvi
Gardent, Claire 131
Gentzen, Gerhard 128, 130
Giovannin, Sambin 5
Gödel, Kurt viii, ix, 1, 2, 5, 6, 8
Goldfarb, Warren D. 2
Goller, Christoph viii, 2
Gordon, Michael J. C. viii, 11, 131, 132
Gould, William Eben 2
Graf, Peter xvii, 14, 94
Green, Ian 107

Hans-Jürgen Bürckert, editors 94
Hardin, Thérèse 130
Henkin, Leon ix, 2, 6, 8, 27, 44
Herbrand, Jaques viii, 1
Hess, Stephan 107, 133
Hilbert, David 1
Hillenbrand, Thomas viii, 12
Hindley, James R. 3, 112
Hintikka, K. J. J. 127
Hofmann, Martin 5, 130
Horacek, Helmut 107
Huang, Xiaorong xvii, xviii, 14, 15, 92, 93, 107, 131
Huet, Gérard P. ix, x, xi, xiii, 2, 4, 6, 10, 19, 49, 51, 52, 54, 56, 62, 127, 129
Hutter, Dieter ix, 107, 131

Issar, Sunil viii, xix, 2, 6, 8, 11, 32, 93, 95, 102, 107, 126, 132

Jan, Smith 5
Jech, Thomas J. 5
Jensen, D. C. x, xi, 2, 6, 10, 127, 129
Johann, Patricia 130
Jouannaud, Jean-Pierre xvi, 12, 131
Jung, Christoph 133

Kaufmann, Stefan ix
Kerber, Manfred ix, xvii, xviii, 14, 15, 92, 93, 107, 131
Kesner, Delia 130
Kirchner, Claude 130
Klein, Lars xvii, 14, 94
Kohlhase, Michael ix, x, xi, xii, xiii, xiv, xv, xvi, xvii, xviii, 3, 4, 6, 9, 10, 12, 14, 15, 27, 35, 48, 49, 52, 53, 54, 55, 56, 61, 64, 82, 91, 92, 93, 99, 107, 113, 114, 115, 125, 127, 128, 129, 131, 132, 133

Konrad, Karsten xii, xviii, 15, 92, 93, 107, 131, 132
Konrad, Karsten 107

Letz, Reinhold viii, 2
Lifantsev, Maxim xvi
Löchner, Bernd viii, 12
Longini-Cohen, Eve 127
Lucchesi, Claudio. L. 2
Lynch, Christopher xvi
Lysne, Olav 12, 131

Mantel, Heiko 54, 107
Martin-Löf, Per 5
Mayr, Klaus viii, 2
Mayr, Richard 4, 12, 131
McCune, William viii, xviii, 2, 14, 95, 96
Meier, Andreas xviii, 15, 92, 93, 107, 131
Melham, Tom F. viii, 11, 131, 132
Melis, Erica ix, xvii, xviii, 14, 15, 92, 93, 107, 131
Miller, Dale 6, 10, 17, 49, 55, 94, 127, 128, 129, 130, 132
Müller, Jörg P. 133

Nadathur, Gopalan 101, 128, 129, 132
Nesmith, Dan viii, xvii, xix, 2, 6, 8, 11, 14, 32, 93, 95, 102, 107, 126, 132
Neumann, John von 4
Nipkow, Tobias 4, 12, 130, 131
Nunes, Alex ix

Owre, Sam viii, 131, 132

Padovani, V. 94
Paulson, Lawrence C. 101, 132
Pereira, Fernando 131
Pfenning, Frank viii, xix, 2, 6, 8, 11, 32, 93, 95, 102, 107, 126, 127, 130, 132
Pietrzykowski, T. x, xi, 2, 6, 10, 127, 129
Pietrzykowski, Thomasz 56
Piris, Javier 12, 131
Pollet, Martin 107
Prawitz, Dag 130
Prehofer, Christian 4, 12, 131

Qian, Zhenyu 94, 130

Redfern, Darren ix
Richardson, Julian 107
Richts, Jörn xvii, 14, 93
Robinson, George A. xvi, 13
Robinson, John A. 1, 2, 48
Rock, Georg viii, xviii, 2

Rubio, Albert xvi, 12, 131
Rudnicki, Piotr 15
Rushby, John viii, 131, 132
Russell, Bertrand 3, 5

Saarbrücken, Programming Systems Laboratory 93, 132
Scedrow, Andrew 132
Schaarschmidt, Wolf xviii, 15, 92, 93, 107, 131
Schairer, Axel 107
Schulte, Christian 93, 132
Schumann, Johann viii, 2
Schürmann, Carsten 132
Schütte, Kurt 2, 9
Scott, Dana S. 5
Seldin, Jonathan P. 3, 112
Sengler, Claus 107
Shankar, Natarajan viii, 131, 132
Shieber, Stuart 131
Siekmann, Jörg ix, xvii, xviii, 14, 15, 92, 93, 101, 107, 130, 131
Skolem, Thoralf 1
Smaill, Alan 107
Smolka, Gert 93, 132
Smullyan, Raymond M. x, xv, 127
Snyder, Wayne xiv, xvi, 4, 10, 16, 18, 49, 50, 55, 56, 93, 130
Sorge, Volker ix, xviii, xix, 15, 92, 93, 107, 110, 131, 133
Steele, Guy L. 14, 92
Stephan, Werner 54
Streicher, Thomas 130
Swieczkowska, Halina xviii, 15, 64, 103, 104, 116

Takahashi, Moto-o x, 128
Takeuti, Gaisi 2, 128
Tammet, Tanel 94
Trybulec, Andrzej xii, 15, 116
Trybulec, Zinaida xviii, 15, 64, 103, 104, 116

Unknown 128

van Heijenoort, Jean 138, 139, 144
van Oostrom, V. 4, 12, 131
Virga, Roberto 131
Vogt, Roland viii, 12

Walsh, Toby ix
Wang, Kang 130
Weidenbach, Christoph viii, xviii, 2, 12
Whitehead, Alfred N. 5
Wolfram, David A. 3, 4, 10, 12, 101, 127, 129, 131

Wolfram, Stephen ix
Wos, Larry viii, xvi, 2, 13
Würtz, Jörg 93, 132

Xi, Hongwei viii, xix, 2, 6, 8, 11, 32, 93, 95, 102, 107, 126, 132

Zermelo, Ernst xii, 3, 5

Index

Σ-algebra, 25
  full, 25
Σ-model, 25
Σ-valuation, 25
λ-calculus, 3
  simply typed, 3, 16
λ-conversion, 4
$\mathcal{POST}$, 92
KEIM, 14
LEO, 2, 14, 92
LEO clause store
  CONT, 95, 96, 98
  Uni-Cont, 98
  Ext-Mod, 98
  EXT, 95–97, 99
  Factorised, 97, 98
  Lightest, 97
  Paramod, 98
  Prim-Subst, 97, 98
  Processed, 98
  Resolved, 97, 98
  SOS, 95, 96
  Unified, 98, 99
  USABLE, 95–97
  set of support, 95
ΩMEGA, 92
TPS, 2, 6, 8, 10, 11
KEIM-toolbox, 93

abstract consistency classes, 9, 35
  compactness of, 36
  non-atomic, 36
  properties for, 35
abstract consistency method, 6
abstract consistency properties, 35, 41
  with primitive equality, 41
abstract extension lemma, 44
admissibility, 19
  of generalised paramodulation rule, 81
  of generalised paramodulation rule, 86
application operator, 19
approximating binding, 18
assignment, 21
atom, 18
axiom of choice, 5
axioms of choice, 5

base type, 16
binding
  approximating, 18
  general, 18
  imitation, 18
  projection, 18
bound variable, 17

calculus, 19
  $\mathcal{EP}$, 13
  $\mathcal{ER}$, 12, 13
  $\mathcal{ERUE}$, 14
calculus ratiocinator, 1
canonical projection, 20
carrier set, 19
characteristic function, 104
clause, 18
  empty, 19
  normal form, 18
  normalisation, 55
  pre-, 18
  proper, 18
closed under subsets, 34
compactness, 34, 36
completeness, 6
  Henkin, 2
  of $\mathcal{EP}_{fc}$, 81
  of $\mathcal{ER}_{fc}$, 61
  of $\mathcal{ERUE}_{fc}$, 90

comprehension
  principles, 3
comprehension principles, 4
congruence, 20
connective, 16
consistency
  non-atomic, 36
constant, 16
constraint, 18
  flex-flex, 18
  flex-rigid, 18
convention, 63
conversion
  α-, 36
  $\beta\eta$-, 17
  λ, 17
  λ-, 4

denotation, 21, 23
Denotatpflicht, 4
derivability, 19
  of proper clauses, 63
description operator, 5
difference-reduction, 33
domain, 19

eager unification, 62
empty clause, 19
equality
  properties of Leibniz equality, 29
  at defining, 6
  defined, 13, 32
  Leibniz, 2, 6, 8, 13, 28, 32, 42
  meta-, 6
  notions of, 6
  of clauses, 54
  primitive, 6, 8, 9, 13, 30, 32, 42
  semantical relation, 6, 8
evaluation
  function, 7
  in functional structures, 24
evaluation function, 21
expressivness, 104
extension lemma, 34, 44
extensional higher-order paramodulation, 71
extensional higher-order resolution, 56, 83
extensional higher-order RUE-Resolution, 83
extensionality
  additional rules for primitive equality, 13
  Boolean, 3–5, 11, 13
  Boolean, for Leibniz equality, 27, 31
  Boolean, for primitive equality, 31
  for Leibniz equality, 27
  for negative primitive equations, 70
  for positive primitive equations, 70
  for primitive and Leibniz equality, 31
  for primitive equality, 31
  full, for Leibniz equality, 27, 31
  full, for primitive equality, 31
  functional, 3–5, 8, 11, 13
  functional, for Leibniz equality, 27
  functional, for primitive equality, 31
  in models, 28
  in models with primitive equality, 31
  treatment, 8, 97

factorisation
  generalised rules, 74
falsity, 26
fix-point free functions, 69
flexible literal heads, 11
formula, 16
frame, 19
free variable, 17
full pre-Σ-algebra, 25
fully invariant, 20
function application, 19
functional
  congruence relation, 20, 45
  pre-structure, 19
  types, 16
functionality, 20

general binding, 11, 18
generalised factorisation rule, 74
generalised paramodulation rule, 73
generalised primitive substitution rule, 74
generalised resolution rules, 73

Henkin model, 8, 27
Henkin model with primitive equality, 30
higher-order logic, 16

- Hilbert style calculus, 6
- Hilbert's program, 1
- Hintikka lemma
  - for $\mathfrak{Acc}_{\beta\mathfrak{b}}$, 40
  - for $\mathfrak{Acc}_{\beta}$, 38
  - for $\mathfrak{Acc}_{\beta\mathfrak{e}}$, 43
  - for $\mathfrak{Acc}_{\beta\mathfrak{eb}}$, 43
  - for $\mathfrak{Acc}_{\beta\mathfrak{f}}$, 39
  - for $\mathfrak{Acc}_{\beta\mathfrak{q}}$, 39
- Hintikka set, 9, 34, 38
- homomorphic extension, 23
- homomorphism, 20

- imitation bindings, 18
- incompleteness, 92, 100, 102
  - of $\mathcal{EP}_{naive}$, 67
  - of naive RUE-Resolution, 83
- incompleteness theorems, 1
- individuals, 7
- interpretation, 7
  - function, 9
  - of constants, 19

- Lego, Coq, type theory
  - constructive, 132
- lifting lemma
  - for $\mathcal{EP}_{fc}$, 71
  - for $\mathcal{ER}_{fc}$, 58
  - for $\mathcal{ERUE}_{fc}$, 85
  - for clause normalisation, 57
- lingua characteristica, 1
- literal, 18
  - negative, 18
  - positive, 18
  - pre-, 18
  - proper, 18
  - unification constraint, 18
- logical connectives, 16

- maximal, 38
- model, 7
  - Henkin, 27
  - Henkin, with primitive equality, 30
  - quotient, 27
  - standard, 7, 27
  - with primitive equality, 30
- model class
  - $\mathfrak{M}_{\beta\mathfrak{b}}$, 26
  - $\mathfrak{M}_{\beta\mathfrak{f}}$, 26
  - $\mathfrak{M}_{\beta\mathfrak{fb}}$, 26
  - $\mathfrak{M}_{\beta\mathfrak{q}}$, 26
  - $\mathfrak{M}_{\beta\mathfrak{qb}}$, 26
- model existence, 44
- model existence theorem, 45
- models
  - functional, 9
  - non-functional, 9

- normal form
  - $\beta\eta$-, 4, 16
  - $\beta\eta$-head-, 4
  - head-, 18, 60, 72, 85

- order, 16, 17, 128

- paradoxes, 3
  - Russel's, 3
- parameters, 16
- paramodulation, 65
  - first-order, 13
  - higher-order, 13
  - into unification constraints, 66
- partial binding, 11
- polarity, 18
- pre-Σ-algebra, 25
- pre-clause, 18
- pre-literal, 18
- pre-structure, 19
  - singleton, 20
  - total, 19
- primitive equality, 12, 41
- primitive substitution, 11, 95, 101
  - generalised rules, 74
  - restricted, 128
- projection bindings, 18
- proper
  - clause, 18
  - clause in $\mathcal{EP}$ and $\mathcal{EP}_f$, 82
  - clause in $\mathcal{ERUE}$ and $\mathcal{ERUE}_f$, 91
  - derivation, 57, 84
- proper derivations, 71
- property
  - $\mathfrak{b}$, 8, 9, 26
  - $\mathfrak{f}$, 26
  - $\mathfrak{q}$, 8, 9, 26
- property $\mathfrak{e}$, 30

- proposition, 16
- pure, 34

- quotient
  - model, 27
  - pre-structure, 20
  - structure, 21, 22

- reduction
  - $\beta\eta$-, 17
- reflexivity resolution, 66
- resolution
  - extensional higher-order, 56, 83
  - extensional higher-order RUE-, 83
  - generalised rules, 73
  - higher-order, 2, 10
  - higher-order constrained, 2
  - on unification constraints, 83
- rule
  - *GFac*, 74
  - *GPrim*, 74
  - $GRes_1$, 73
  - $GRes_2$, 73
  - *Equiv'*, 70
  - *Func'*, 70
  - *GPara*, 73
  - *Leib'*, 70
  - *Leib*, 62
  - *Para'*, 66
  - *Para*, 66
  - *Prim*, 11
  - *Sym*, 84
  - admissible, 19
  - derivable, 19

- satisfies, 25
- saturated, 36
  - atomically, 36
- saturation, 9
- semantics
  - Henkin, 2, 5, 6
  - standard, 2, 5, 6, 8
- semi-valuation, 9
- sentence, 16
- set
  - of support, 14
  - of variables, 16
  - theory, 3, 103
- set of constants, 16
- set of support, 92
- signature, 16
- singleton pre-Σ-structure, 20
- Skolem term, 17
- Skolemisation, 10, 17
- solved
  - for, 18
  - variable, 18
- SOS-architecture, 92, 95
- soundeness
  - of $\mathcal{CNF}$, 55
- soundness
  - of β-equality, 23
  - of $\mathcal{EP}_{naive}$, 67
  - of extensional higher-order paramodulation, 71
  - of extensional higher-order resolution, 56
  - of extensional higher-order RUE-resolution, 84
- standard model, 27
- structure, 19, 21
- substitution, 17
  - application of, 17
- subsumption, 14, 94
  - extensional higher-order, 103
  - higher-order, 103
- sufficiently pure, 34
- symmetric unification constraints, 84

- term indexing, 14, 15
  - extensional higher-order, 102
  - higher-order, 14, 94
  - syntactical higher-order, 102
- term rewriting, 4, 33
  - higher-order, 12
  - syntactical, 4
- term structure, 22
  - β-, 22
  - $\beta\eta$-, 22
- theorem equivalence
  - of $\mathcal{ER}_{fc}$ and $\mathcal{ER}_f$, 64
  - of $\mathcal{ERUE}_{fc}$ and $\mathcal{ERUE}_f$, 91
  - of $\mathcal{EP}$ and $\mathcal{EP}_{fc}$, 82
  - of $\mathcal{ER}$ and $\mathcal{ER}_{fc}$, 64
  - of $\mathcal{ERUE}$ and $\mathcal{ERUE}_{fc}$ (or $\mathcal{ERUE}_f$), 91

- truth, 26
- truth values, 7
- type, 16
  - function, 19
- type theory
  - classical, 2, 3, 5
  - extensional, 130
  - intensional, 130
- typed
  - collection, 19
  - collection of sets, 19
  - mapping, 19
- typed binary relation, 19
- typed function, 19

- unification, 55
  - continuation, 100
  - eager, 62, 95, 101
  - extensional higher-order, 95, 100
  - higher-order, 2, 55, 56, 93
  - higher-order E-, 12, 56
  - higher-order pre-, 2, 55
- unification constraint, 18
- unifying principle, 6
- universe of type α, 19

- valid, 25
- valuation, 8, 25, 26
  - with equality, 30
- value, 23
- variable, 16
  - assignment, 7
  - conditions, 10
  - solved, 18
- variable assignment, 21

- weak derivability
  - of generalised factorisation rule, 74, 86
  - of generalised paramodulation rule, 78
  - of generalised primitive substitution rule, 74, 86
  - of generalised resolution rules, 75, 86

Symbol Index

- $[\mathbf{A}/X]$ ... substitution, 17
- $[\mathbf{A}/X]\mathbf{T}$ ... application of substitution, 17
- $\mathfrak{Acc}_{\beta\mathfrak{b}}$ ... abstract consistency class, 35, 46
- $\mathfrak{Acc}_{\beta}$ ... abstract consistency class, 46
- $\mathfrak{Acc}_{\beta\mathfrak{e}}$ ... abstract consistency class, 41, 47
- $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ ... abstract consistency class, 41, 47
- $\mathfrak{Acc}_{\beta\mathfrak{f}}$ ... abstract consistency class, 35, 46
- $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ ... abstract consistency class, 35, 46
- $\mathfrak{Acc}_{\beta\mathfrak{q}}$ ... abstract consistency class, 35, 46
- $\mathfrak{Acc}_{\beta\mathfrak{qb}}$ ... abstract consistency class, 35
- $\mathcal{BT}$ ... base types, 16
- $\Box$ ... empty clause, 19
- $\mathcal{C}_\tau$ ... set of constants, 16
- $\mathfrak{q}$ ... semantical equality relation, 6, 17
- $\mathbf{E}_1^{Para}$ ... example for incompleteness of $\mathcal{EP}_{naive}$, 68
- $\mathbf{E}_2^{Para}$ ... example for incompleteness of $\mathcal{EP}_{naive}$, 68
- $\mathbf{E}_3^{Para}$ ... example for incompleteness of $\mathcal{EP}_{naive}$, 68
- $\mathbf{E}_4^{Para}$ ... example for incompleteness of $\mathcal{EP}_{naive}$, 68
- $\mathbf{E}_5^{Para}$ ... example for incompleteness of $\mathcal{EP}_{naive}$, 68
- $\mathcal{EP}_{fc}$ ... extensional higher-order paramodulation with full unification and unfolded clause normalisation, 71
- $\mathcal{ER}_{fc}$ ... extensional higher-order resolution with full unification and unfolded clause normalisation, 56
- $\mathcal{ERUE}_{fc}$ ... extensional higher-order RUE-resolution with full unification and unfolded clause normalisation, 83
- $\mathcal{EP}_f$ ... extensional higher-order paramodulation with full unification, 71
- $\mathcal{ER}_f$ ... extensional higher-order resolution with full unification, 56
- $\mathcal{ERUE}_f$ ... extensional higher-order RUE-resolution with full unification, 83
- $\mathcal{EP}_{naive}$ ... naive higher-order paramodulation, 67
- $\mathcal{EP}$ ... extensional higher-order paramodulation, 13, 71

$\mathcal{ER}$ . . . extensional higher-order resolution, 12, 13, 56
$\mathcal{ERUE}$ . . . extensional higher-order RUE-resolution, 14, 83
$\mathfrak{H}$ . . . model class, 10
$\doteq$ . . . Leibniz equality, 6, 17, 42
$\mathrm{EXT}_{\dot{=}}^{\alpha\to\beta}$ . . . functional extensionality for Leibniz equality, 11, 27, 31
$\mathrm{EXT}_{\dot{=}}^{o}$ . . . Boolean extensionality for Leibniz equality, 11, 27, 31
$\mathfrak{M}_{\beta\mathfrak{b}}$ . . . model class, 9, 10
$\mathfrak{M}_{\beta}$ . . . model class, 7, 9, 10
$\mathfrak{M}_{\beta\mathfrak{f}}$ . . . model class, 8, 10
$\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}$ . . . model class, 8, 10
$\mathfrak{M}_{\beta\mathfrak{q}}$ . . . model class, 8, 10
$\equiv$ . . . meta-equality, 6, 17
$\equiv_{\alpha}$ . . . α-equality, 17
$\equiv_{\beta\eta}$ . . . $\beta\eta$-equality, 17, 20
$\equiv_{\beta}$ . . . β-equality, 17, 20
$\equiv_{\eta}$ . . . η-equality, 17
Π . . . logical connective, 16
$=$ . . . primitive equality, 6, 16, 17, 42
$\mathrm{EXT}^{\alpha\to\beta}$ . . . functional extensionality for primitive equality, 31
EXT^{o} . . . Boolean extensionality for primitive equality, 31
$\mathfrak{S}\mathfrak{T}$ . . . model class, 7
Σ . . . signature, 16
$\Sigma^{=}$ . . . signature with primitive equality, 16
$\mathbf{T}_{[\mathbf{A}/X]}$ . . . application of substitution with subsequent head-normalisation, 17
$\mathbf{T}{\downarrow_{\beta\eta}}$ . . . $\beta\eta$-normal form, 17
$\mathbf{T}{\downarrow_{\beta}}$ . . . β-normal form, 17
$\mathbf{T}{\downarrow_{\eta}}$ . . . η-normal form, 17
$\mathbf{T}{\downarrow_{h}}$ . . . head-normal form, 17
$\mathcal{TS}(\Sigma)$ . . . term structure, 22
$\mathcal{TS}(\Sigma)^{\beta}$ . . . β-term structure, 22
$\mathcal{TS}(\Sigma)^{\beta\eta}$ . . . $\beta\eta$-term structure, 22
$\mathcal{T}$ . . . set of types, 16
$\mathcal{V}_{\mathcal{T}}$ . . . set of variables, 16
$\mathit{wff}_{\alpha}(\Sigma^{=})$ . . . set of $\Sigma^{=}$-terms of type α, 16
$\mathit{wff}(\Sigma^{=})$ . . . set of $\Sigma^{=}$-terms, 16
$\mathit{wff}_{\alpha}(\Sigma)$ . . . set of all Σ-terms of type α, 16
$\mathit{wff}(\Sigma)$ . . . set of Σ-terms, 16
$\mathit{wff}_{o}(\Sigma)$ . . . set of propositions, 16
$\mathit{cwff}_{o}(\Sigma)$ . . . set of sentences, 16
$\nabla_{\wedge}$ . . . abstract consistency property, 35
$\nabla_{\mathfrak{b}}$ . . . abstract consistency property, 35
$\nabla_{\mathfrak{f}}$ . . . abstract consistency property, 35
∇_{β} . . . abstract consistency property, 35
∇_{c} . . . abstract consistency property, 35
$\nabla_{\exists}$ . . . abstract consistency property, 35
$\nabla_{\forall}$ . . . abstract consistency property, 35
$\nabla_{\doteq}^{=}$ . . . abstract consistency property, 41
$\nabla_{\neg}$ . . . abstract consistency property, 35
$\nabla_{\vee}$ . . . abstract consistency property, 35
$\nabla_{=}^{\doteq}$ . . . abstract consistency property, 41
$\nabla_{\mathfrak{q}}$ . . . abstract consistency property, 35
@ . . . application operator, 19
$@^{\sim}$ . . . application operator for equivalence classes modulo $\sim$, 20
$@^{\alpha\beta}$ . . . application operator, 19
$\to$ . . . type constructor, 16
$\setminus$ . . . set operator in classical type theory, 104
$\mathcal{D}^{\sim}$ . . . domain of equivalence classes modulo $\sim$, 20
$\mathcal{D}_{\alpha}$ . . . domain of type α, 19
$\mathcal{D}_{o}$ . . . domain of truth values, 7
$\mathcal{D}_{\iota}$ . . . domain of individuals, 7
$\mathcal{E}$ . . . evaluation function, 21
$\mathcal{E}_{\varphi}$ . . . evaluation function, 21
$\mathcal{G}_{\alpha}^{h}$ . . . set of imitation bindings, 18
$\mathcal{G}_{\alpha}^{j}$ . . . set of projection bindings, 18
$\mathcal{I}$ . . . interpretation function, 7, 9, 19
$\mathcal{I}^{\sim}$ . . . interpretation of constants modulo $\sim$, 20
$\mathcal{I}_{\varphi}$ . . . evaluation function, 7, 23
$\cap$ . . . set operator in classical type theory, 104
$\longrightarrow_{\beta}$ . . . β-reduction, 17
$\longrightarrow_{\eta}$ . . . η-reduction, 17
$\cup$ . . . set operator in classical type theory, 104
$:=$ . . . equality at defining, 6, 17
$\dot{-}$ . . . set operator in classical type theory, 104
$\mathbf{F}_{o}$ . . . falsity, defined concept, 26
free($\mathbf{T}$) . . . free variables of, 17
$\mathcal{AB}_{\alpha}^{h}$ . . . set of partial bindings, 18
$\overline{\nabla}_{\wedge}$ . . . Hintikka set property, 38
$\overline{\nabla}_{\mathfrak{b}}^{-}$ . . . Hintikka set property, 40
$\overline{\nabla}_{\mathfrak{b}}^{+}$ . . . Hintikka set property, 40

$\overline{\nabla}_{\mathfrak{f}}$ ... Hintikka set property, 39
$\overline{\nabla}_{\beta}$ ... Hintikka set property, 38
$\overline{\nabla}_{c}^{1}$ ... Hintikka set property, 38
$\overline{\nabla}_{c}^{2}$ ... Hintikka set property, 38
$\overline{\nabla}_{c}^{3}$ ... Hintikka set property, 38
$\overline{\nabla}_{\doteq}^{\Leftrightarrow}$ ... Hintikka set property, 40
$\overline{\nabla}_{\doteq}^{c}$ ... Hintikka set property, 40
$\overline{\nabla}_{\doteq}^{r}$ ... Hintikka set property, 38
$\overline{\nabla}_{\doteq}^{s}$ ... Hintikka set property, 38
$\overline{\nabla}_{\doteq}^{tr}$ ... Hintikka set property, 38
$\overline{\nabla}_{\doteq}^{sy}$ ... Hintikka set property, 38
$\overline{\nabla}_{\exists}$ ... Hintikka set property, 38
$\overline{\nabla}_{\forall}$ ... Hintikka set property, 38
$\overline{\nabla}_{\neg}$ ... Hintikka set property, 38
$\overline{\nabla}_{\vee}$ ... Hintikka set property, 38
$\overline{\nabla}_{=}^{\doteq-}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{\doteq+}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{q+}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{q-}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{b+}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{\Leftrightarrow}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{b-}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{c}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{r}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{s}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{tr}$ ... Hintikka set property, 43
$\overline{\nabla}_{=}^{sy}$ ... Hintikka set property, 43
$\overline{\nabla}_{\mathfrak{q}}^{-}$ ... Hintikka set property, 39
$\overline{\nabla}_{\mathfrak{q}}^{+}$ ... Hintikka set property, 39
$\overline{\nabla}_{t}$ ... Hintikka set property, 38
$\overline{\nabla}_{\doteq}^{tf-}$ ... Hintikka set property, 40
$\overline{\nabla}_{\doteq}^{tf+}$ ... Hintikka set property, 40
$\in$ ... set operator in classical type theory, 104
$[\![f]\!]_{\sim}$ ... equivalence class of f modulo $\sim$, 20
meets ... set operator in classical type theory, 104
misses ... set operator in classical type theory, 104
$\models$ ... semantical validity, 25
$\models_{\varphi}$ ... semantical satisfiability, 25
$\neg$ ... logical connective, 16
ord ... order of a type, 16
φ ... variable assignment, 7, 21
$\pi_{\sim}$ ... canonical projection, 20
$\mathfrak{b}$ ... property for models, 8, 9, 26
$\mathfrak{e}$ ... property for models, 30
$\mathfrak{f}$ ... property for models, 26
$\mathfrak{q}$ ... property for models, 8, 26
$\mathcal{A}/_{\sim}$ ... quotient structure, 22
$\mathcal{A}/_{\sim}$ ... quotient model, 27
$\mathcal{A}/_{\sim}$ ... quotient pre-structure, 21
υ ... valuation, 8, 26
$\sim$ ... congruence, 20
$\sim_{\mathcal{H}}$ ... functional congruence relation, 45
$\dot{\sim}_{\mathcal{H}}$ ... functional congruence relation, 45
$\subseteq$ ... set operator in classical type theory, 104
$\mathbf{T}_{o}$ ... truth, defined concept, 26
$\vdash^{r_n}$ ... single step derivation with rule r_n, 19
$\vdash_{R}$ ... derivation in calculus R, 19
$\vdash_{R}^{n}$ ... derivation in calculus R of length n, 19
$\vee$ ... logical connective, 16