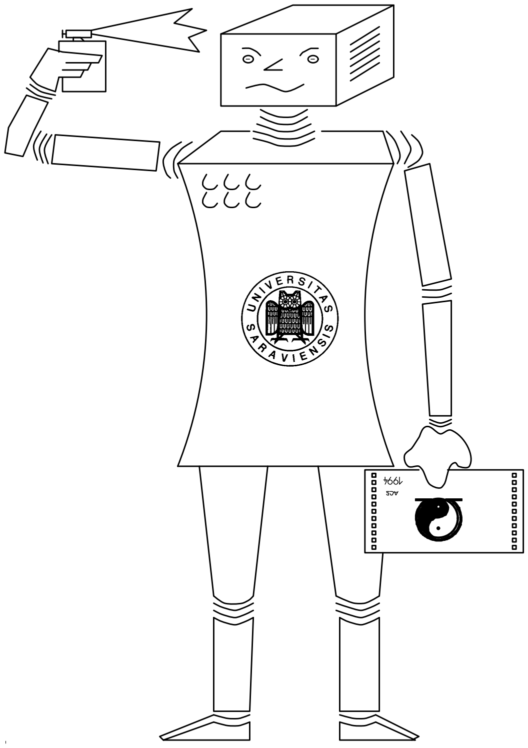
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/



# A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle

Michael Kohlhase

**Published as:** Doctoral Thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994

# A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle

Der Technischen Fakultät
der Universität des Saarlandes
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
vorgelegte Dissertation

von

**Dipl. Math. Michael Kohlhase**

# Abstract

The usage of sorts in first-order automated deduction has brought greater conciseness of representation and a considerable gain in efficiency by reducing the search spaces involved. This suggests that sort information can be employed in higher-order theorem proving with similar results.

This thesis develops a sorted higher-order logic $\Sigma\mathcal{HOL}$ suitable for automatic theorem proving applications. $\Sigma\mathcal{HOL}$ is based on a sorted $\lambda$-calculus $\Sigma\Lambda$ that is obtained by extending Church's simply typed $\lambda$-calculus by a higher-order sort concept including term declarations and functional base sorts. The term declaration mechanism studied here is powerful enough to allow convenient formalization of a large body of mathematics, since it offers natural primitives for domains and codomains of functions, and allows to treat function restriction. Furthermore, it subsumes most other mechanisms for the declaration of sort information known from the literature, and can thus serve as a general framework for the study of sorted higher-order logics. For instance, the term declaration mechanism of $\Sigma\mathcal{HOL}$ subsumes the subsorting mechanism as a derived notion, and hence justifies our special form of subsort inference.

We present sets of transformations for sorted higher-order unification and pre-unification, and prove the nondeterministic completeness of the algorithm induced by these transformations. The main technical difficulty of unification in $\Sigma\Lambda$ is that the analysis of general bindings is much more involved than in the unsorted case, since in the presence of term declarations well-sortedness is not a structural property. This difficulty is overcome by a structure theorem that links the structure of a formula to the structure of its sorting derivation.

We develop two notions of set-theoretic semantics for $\Sigma\mathcal{HOL}$. General $\Sigma$-models are a direct generalization of Henkin's general models to the sorted setting. Since no known machine-oriented calculus can adequately mechanize full extensionality, we generalize general $\Sigma$-models further to $\Sigma$-model structures, which allow full extensionality to fail. The notions of $\Sigma$-model structures and of general $\Sigma$-models allows us to prove unifying principles for these. These model-theoretic variants of Andrews' unifying principle for type theory can be used as a powerful tool in completeness proofs of higher-order calculi.

Finally, we use our pre-unification algorithms as a central inference procedure for a sorted higher-order resolution calculus in the spirit of Huet's Constrained Resolution. This calculus is proven sound and complete with respect to our semantics. It differs from Huet's calculus by allowing early unification strategies and using variable dependencies. For the completeness proof we make use of our unifying principle, and prove a strong lifting lemma.

# Deutsche Zusammenfassung

## Einleitung

Im Bereich der Deduktionssysteme gibt es zwei Grundparadigmen, das der interaktiven und das der vollautomatischen Systeme. Die logischen Grundlagen der *vollautomatischen Beweiser* zeichnen sich durch starke, maschinenorientierte Kalküle aus, die dann als heuristische Suchprozeduren implementiert werden. In diesem Teilbereich haben sich unifikationsbasierte Kalküle, wie Resolution [Rob65], Tableaux [Smu68, Fit90] oder das Matings/Konnektions-Verfahren [And81, Bib83] durchgesetzt. Um diese Kalküle effizient abarbeiten zu können ist es vorteilhaft, wenn die grundlegenden Unterprozesse, wie die Unifikation, entscheidbar und von relativ geringer Komplexität sind. Nicht zuletzt deshalb beschränken sich automatische Beweissysteme meist auf Varianten der Logik erster Stufe. Obwohl diese Beweiser eine beträchtliche Stärke erreicht haben, wachsen die Suchräume so schnell, daß es prinzipielle Grenzen für die entsprechenden Suchprozeduren gibt.

*Interaktive Deduktionsysteme* wie zum Beispiel Nuprl [CAB+86], HOL [Gor85, GM93], PVS [ORS92] oder Isabelle [PN90] verlassen sich nicht so sehr auf die volle Automatisierung, sondern auf den Benutzer, der die Beweissuche im zugrundeliegenden Kalkül direkt durch entsprechende Eingriffe steuert. Da bei dieser Vorgehensweise universelle Beweisprozeduren keine Rolle spielen, wurden sehr ausdrucksmächtige Logiken und Kalküle entwickelt, die sich an typischer, menschlicher Argumentation und mathematischer Praxis orientieren. Allerdings gibt auch hier der Benutzer nicht sämtliche Einzelschritte genau vor, sondern er wird durch einen sogenannten Taktikmechanismus unterstützt.

Diese Arbeit ist durch den Versuch motiviert, Deduktionsysteme zu entwickeln, die die Stärken beider Paradigmen in sich vereinen. Wenn Deduktionssysteme für Mathematiker oder in der Software-Verifikation zukünftig nützlich sein sollen, dann brauchen sie einerseits die ausdrucksstarken Formalismen (wie sie zur Zeit in interaktiven Systemen zu finden sind) und andererseits die starken, universellen Beweismethoden (aus den vollautomatischen Systemen erster Stufe). Sonst wird formale Deduktion zu umständlich und teuer bleiben, um benutzt zu werden. Natürlich kann die Lösung dieses Problems nicht ein vollautomatisches Beweissystem in einer expressiven Logik sein, denn es wäre noch stärker der kombinatorischen Explosion des Suchraums unterworfen, aber ein solches System kann als eine „*Logikmaschine*“ in einem interaktiven System eingesetzt werden, das diese genau dann aufruft, wenn andere Verfahren noch Lücken im Beweis lassen.

Das Ziel dieser Doktorarbeit ist es, eine Mechanisierung einer sehr ausdrucksmächtigen Logik zu entwickeln, die für den Einsatz in der Mathematik geeignet ist. Da sich mathematische Sachverhalte in Logiken erster Stufe nicht adäquat formalisieren lassen und sich im automatischen Beweisen erster Stufe gezeigt hat, daß sortierte Logiken eine effiziente, automatische Beweissuche ermöglichen, haben wir uns für eine *sortierte Logik höherer Stufe* als Formalismus entschieden. Um den konkreten Ansatz besser verstehen zu können, werden wir nun kurz auf die Gebiete des automatischen Beweisens höherer Stufe und des sortierten Beweisens erster Stufe eingehen.

In der Mathematik gibt es einfache Objekte, wie Zahlen oder Punkte der euklidischen Ebene, und es gibt komplexere Objekte, wie Mengen, Funktionen oder Vektorräume. Die Logik erster Stufe beschränkt sich darauf, gerade die einfachen Objekte zu beschreiben, indem sie ausschließlich Variablen für diese zuläßt. Um mathematische Sachverhalte natürlich und adäquat darstellen zu können, benötigt man also Logiken höherer Stufe, in denen sich alle Objekte beschreiben lassen. Zum Beispiel läßt sich in der Logik erster Stufe nach dem ersten Gödelschen Unvollständigkeitssatz [Göd31] die Arithmetik der natürlichen Zahlen nicht darstellen. Deswegen entstand bereits unmittelbar nach der Einführung von unifikationsbasierten Kalkülen [Rob65] für das automatische Beweisen erster Stufe ein großes Interesse daran, solche Methoden auch auf Logiken höherer Stufe zu übertragen [Rob68, Rob69a]. Mitte der Siebziger Jahre wurden dafür dann die theoretischen Grundlagen gelegt [Gou66, And71, Hue72, Hue76, JP72, JP76] und auch die ersten Systeme gebaut [Dar71, Hue72].

Viele Verfahren, die für das Beweisen in Logiken höherer Stufe entwickelt wurden, bauen auf dem einfach getypten $\lambda$-Kalkül von Church auf [Chu40]. Huet entwickelte dafür einen Unifikationsalgorithmus [Hue75] und konnte aufbauend auf einer Arbeit von Andrews [And71] einen Resolutionskalkül für Logik höherer Stufe angeben [Hue72]. Allerdings wurden bald auch eine ganze Reihe von Negativresultaten, wie die Unentscheidbarkeit der Unifikation [Hue72, Luc72, Gol81] oder die Nichtexistenz von allgemeinsten Unifikatoren [Gou66] für die Mechanisierung von Logiken höherer Stufe, gefunden. Darüberhinaus ist die Unifikation höherer Stufe außerordentlich komplex, so daß der Suchbaum im Unifikationsalgorithmus eine sehr hohe Verzweigungsrate hat. Glücklicherweise konnte Huet das Unifikationsproblem höherer Stufe auf die sogenannte *Prä-Unifikation* einschränken, darin werden Termpaare, in denen beide Kopfsymbole freie Variablen sind (sogenannte flexible Paare), als bereits gelöst angesehen. Diese Reduktion erlaubt es einen so großen Teil des Suchraums nach Unifikatoren abzuschneiden, daß das Unifikationsproblem praktisch handhabbar wird. Diese Einschränkung erhält die Lösbarkeit der Unifikationsprobleme (flexible Paare sind immer lösbar, indem man für die Kopfvariablen konstante Funktionen einsetzt, die ihre Argumente absorbieren) und ist daher für Widerlegungskalküle höherer Stufe ausreichend.

Im TPS-Projekt [AINP90] von Andrews an der Carnegie Mellon University konnte zwar die generelle Machbarkeit der Mechanisierung von Logiken höherer Stufe gezeigt werden, aber es wurde gleichzeitig auch deutlich, daß die angewandten Methoden noch zu schwach für den praktischen Einsatz sind. Unserer Meinung nach lassen sich die Schwächen hauptsächlich darauf zurückführen, daß in Systemen höherer Stufe viele der Methoden und Techniken (wie zum Beispiel Spezialbehandlung für Sorten und Gleichheit, raffinierte Suchverfahren oder Implementierungstechniken wie Termindexing), die das automatische Beweisen erster Stufe praktikabel gemacht haben, noch nicht eingesetzt, beziehungsweise entwickelt worden sind. Wir greifen uns für diese Arbeit die Spezialbehandlung von Sorten heraus und gehen nun näher auf die Sortenbehandlung in Logiken erster Stufe ein.

*Sortierte Logiken* [Her30, Sch38, Sch51, Wan52, Obe62] sind logische Systeme, die das Universum in Klassen (Zahlen, Punkte, Flächen, Füchse, Wölfe,...) einteilen und in denen logische Konstanten, Variablen und Terme syntaktisch dadurch gekennzeichnet sind, welchen dieser Klassen sie angehören. Durch Bedingungen an die Wohlgeformtheit (in diesem Fall also Wohlsortiertheit) von Ausdrücken wird sichergestellt, daß nur semantisch sinnvolle Objekte durch logische Formeln dargestellt werden können. So darf eine Funktion der Sorte $\mathbb{A} \to \mathbb{B}$ nur auf einen Term $\mathbf{A}$ angewandt werden, wenn dieser auch die Sorte $\mathbb{A}$ besitzt. Aus der Sicht der Mechanisierung ist es besonders wichtig, daß die Kodierung von Teilen eines mathematischen Sachverhalts als Sorteninformation es erlaubt, diesen Teil effizienter zu verarbeiten [Wal85, SS89, Coh87, Coh89, Fri90]. Die Verwendung von *sortierter*

*Unifikation* ist sinnvoll, denn sie stellt sicher, daß die resultierenden Instanzen wohlsortiert und damit semantisch sinnvolle Ausdrücke sind, die etwas zu einem Beweis beitragen können. Berücksichtigt man nämlich die Sorteninformation, so sind viele Termpaare nicht unifizierbar, die es wären, wenn man diese Information außer acht ließe. Da die Resolutionsmöglichkeiten in einer Klauselmenge von der Unifizierbarkeit abhängen, kann also die Verwendung von sortierten Logiken und Kalkülen wie ein Filter auf dem Suchraum wirken und diesen für die Beweissuche so drastisch einschränken, daß damit Probleme einer automatischen Behandlung zugänglich werden, die mit unsortierten Methoden nicht lösbar sind.

Eine besonders weit entwickelte und einflußreiche Sortenlogik erster Stufe ist das System von Schmidt-Schauß [SS89], in dem verschiedene Formen der Deklaration von Sorteninformation zu dem konzeptionell einfachen, aber extrem ausdrucksmächtigen Mechanismus der *Termdeklarationen* verallgemeinert sind. Termdeklarationen sind Paare $[\mathbf{A}::\mathbb{A}]$ bestehend aus einem Term $\mathbf{A}$ und einer Sorte $\mathbb{A}$, die festlegen, daß alle Instanzen von $\mathbf{A}$ die Sorte $\mathbb{A}$ haben. Hiermit können nicht nur die Sorten von Konstanten und Funktionen ($[0::\mathbb{N}], [+::\mathbb{N} \times \mathbb{N} \to \mathbb{N}]$) dargestellt werden, sondern auch Untersortenbeziehungen $[X_{\mathbb{E}}::\mathbb{N}]$ (da alle geraden Zahlen ($\mathbb{E}$) auch natürliche Zahlen ($\mathbb{N}$) sind) und sogar Informationen wie die, daß die Verdoppelung von natürlichen Zahlen immer gerade Zahlen liefert ($[X_{\mathbb{N}} + X_{\mathbb{N}}::\mathbb{E}]$).

Wir verallgemeinern diese Logik zu einem logischem System $\Sigma\mathcal{HOL}$ höherer Stufe und stellen einen sortierten Resolutionskalkül $\Sigma\mathcal{HR}$ vor, der eine Mechanisierung von $\Sigma\mathcal{HOL}$ mit einem drastisch eingeschränkten Suchraum erlaubt. Wie schon in Schmidt-Schauß System für die Logik erster Stufe, benutzt die im Kalkül verwendete, sortierte Unifikation die Sorteninformation, um viele unsortiert mögliche Resolutionsschritte als unnötig zurückzuweisen.

Die Entwicklung der Logik $\Sigma\mathcal{HOL}$ läßt sich im wesentlichen in zwei Teile einteilen, die sich auch im Aufbau der Arbeit widerspiegeln. Im ersten Teil (Kapitel 2 und 3) wird ein sortierter $\lambda$-Kalkül $\Sigma\Lambda$ vorgestellt, in dem sich die algebraischen Eigenschaften der Wohlsortiertheit und der sortierten $\lambda$-Konversion unabhängig vom logischen Gehalt beschreiben lassen. Wir entwickeln auf dieser Basis sortierte Unifikationsalgorithmen höherer Stufe für $\Sigma\Lambda$ (Kapitel 4). Im zweiten Teil wird dieser $\lambda$-Kalkül dann mit einer logischen Interpretation versehen und wir untersuchen sie dann aufbauend auf den Ergebnissen des ersten Teils (Kapitel 5). Schließlich geben wir mit Hilfe dieser Unifikationsalgorithmen eine Mechanisierung von $\Sigma\mathcal{HOL}$ in einem Resolutionskalkül höherer Stufe an und beweisen dessen Korrektheit und Vollständigkeit (Kapitel 6).

## $\Sigma\Lambda$: Ein sortierter $\lambda$-Kalkül (Kapitel 3)

Ausgehend von Churchs einfach getyptem $\lambda$-Kalkül $\Lambda$ [Chu40] (vergleiche auch [HS86] oder [And86]) und Schmidt-Schauß Sortenlogik [SS89] erster Stufe mit Termdeklarationen werden wir im folgenden die Einführung des sortierten $\lambda$-Kalküls $\Sigma\Lambda$ zusammenfassen. $\Sigma\Lambda$ entsteht aus $\Lambda$ dadurch, daß wir ein Sortensystem höherer Stufe hinzufügen und den Begriff der Wohlgetyptheit zur Wohlsortiertheit verfeinern.

Da die Begriffe des *Typs* und der *Sorte* in der Literatur recht uneinheitlich gebraucht werden, wollen wir nun die zugrundeliegenden Phänomene etwas genauer betrachten und die Begriffe für diese Arbeit voneinander abgrenzen. Der Begriff des Typs stammt aus den Logiken höherer Stufe und wurde erstmals von Russell zur Vermeidung der nach ihm benannten Antinomien in Logiken höherer Stufe entwickelt und dann in die $\lambda$-Kalküle übernommen. In $\Lambda$ werden Typen durch den Abschluß unter Funktionstypen $\alpha \to \beta$ aus einfacheren Typen $\alpha$ und $\beta$ gebildet, andere $\lambda$-Kalküle lassen noch weitere Mechanismen zur Typbildung (zum Beispiel Paartypen) zu. Das jeweilige Typsystem ist für moderne $\lambda$-Kalküle ein so wichtiger Bestandteil, daß diese auch den Namen Typtheorie tragen. Der Begriff der Sorte wurde bereits in der ersten Hälfte dieses Jahrhunderts entwickelt [Her30, Sch38, Sch51, Wan52, Obe62] und wird heute als fester Bestandteil der Deduktionssysteme erster Stufe verwendet. Er wird dort als Repräsentationsmechanismus genutzt, der es erlaubt, Teile einer Axiomatisierung in Form von Sortendeklarationen zu fassen und dann im Kalkül besonders effizient zu handhaben. Einfache (trivial geordnete) Sortensysteme können als ein Spezialfall erster Stufe von Churchs Typsystem angesehen werden, denn sie können in getypten Logiken dadurch simuliert werden, daß neue Basistypen eingeführt werden. Modernere Sortensysteme reichern diesen einfachen Mechanismus allerdings durch einen Subsorten- oder sogar Termdeklarationsmechanismus an, um das Sortensystem ausdrucksmächtiger zu machen. Wir werden im folgenden den Teil eines Typen/Sortensystems mit dem Wort „Typ“ belegen, der zur *Konsistenzsicherung* verwandt wird, während wir den Teil, der als *Repräsentationsmechanismus* genutzt wird mit dem Begriff „Sorte“ belegen.

In $\Sigma\Lambda$ behalten wir zur Sicherung der Konsistenz Churchs Typsystem bei und führen zusätzlich ein System von Sorten ein, das zur Repräsentation von taxonomischem Wissen genutzt werden soll. Natürlich muß dieses Sortensystem dem Typsystem untergeordnet sein, damit die Logik in sich konsistent bleibt. In $\Sigma\Lambda$ sind Sorten also selbst getypte Objekte; insbesondere beschreibt eine Sorte $\mathbb{A}$ vom Typ $\alpha \to \beta$ Funktionen und hat deswegen eine Definitionsbereichssorte $\mathfrak{d}(\mathbb{A})$ vom Typ $\alpha$ und eine Wertebereichssorte $\mathfrak{r}(\mathbb{A})$ vom Typ $\beta$. Beschränkungen der Zulässigkeit von Termdeklarationen stellen dann sicher, daß Terme und ihre Sorten vom Typ her zusammenpassen. So läßt sich beispielsweise die Teilmenge $\mathcal{C}^0(\mathbb{R}, \mathbb{R})$ der stetigen Funktionen aller reellen Funktionen $\mathcal{F}(\mathbb{R}, \mathbb{R})$ darstellen durch eine Basissorte $\mathbb{C}$ mit $\mathfrak{d}(\mathbb{C}) = \mathbb{R}$ und $\mathfrak{r}(\mathbb{C}) = \mathbb{R}$, wobei $\mathbb{R}$ die Sorte der reellen Zahlen ist.

Die Semantik von $\Sigma\Lambda$ ist eine Verfeinerung der *verallgemeinerten Semantik* nach Henkin [Hen50] auf partielle Funktionen. Wir verwenden hier eine verallgemeinerte Semantik, da die Standardsemantik nach dem Gödelschen Unvollständigkeitssatz [Göd31] keine vollständigen Kalküle zuläßt und deswegen keinen brauchbaren Vollständigkeitsbegriff liefert. Eine verallgemeinerte Semantik interpretiert die Funktionsuniversen $\mathcal{D}_{\alpha\to\beta}$ nicht als die Menge $\mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$ aller Funktionen von $\mathcal{D}_\alpha$ nach $\mathcal{D}_\beta$, sondern als eine Teilmenge, die allerdings alle durch Terme darstellbaren Funktionen enthält. Diese Klasse von verallgemeinerten Modellen ist so reich an Nichtstandardmodellen, daß alle nicht ableitbaren Sätze Gegenbeispiele haben und deswegen nicht allgemeingültig sein können. Wir bemühen uns in dieser Dissertationsschrift um einen, im Vergleich zur Literatur, algebraischen Aufbau der Semantik höherer Stufe. Dies und die hier entwickelten Begriffe der $\Sigma$-Struktur und $\Sigma$-Algebra erleichtern uns nachher die technischen Grundlagen für die Vollständigkeitsbeweise.

Im Gegensatz zu einfacheren Sortensystemen läßt sich in einem logischen System mit Termdeklarationen die Sorte eines Terms nicht ableiten, indem man ausschließlich rekursiv die Struktur des Terms analysiert. Deswegen ist es für $\Sigma\Lambda$ notwendig, die Wohlsortiertheit eines Terms durch ein Inferenzsystem zu definieren. Diese, aus dem Bereich der $\lambda$-Kalküle stammende Technik erlaubt die im Bereich der Logik erster Stufe selten verwendete Beweistechnik der Induktion über Herleitungen, die wir in dieser Arbeit ausgiebig verwenden. Um in einem Inferenzsystem für einen sortierten $\lambda$-Kalkül Konsistenz sicherzustellen müssen Signaturen (Mengen von Termdeklarationen) bestimmte Bedingungen erfüllen. Insbesondere müssen die Terme in den Termdeklarationen wohlsortiert sein, was allerdings nur relativ zu einer gültigen Signatur definiert ist. Weiterhin muß einerseits die $\beta\eta$-Reduktion gewissen Sortenbedingungen gehorchen, andererseits muß Wohlsortiertheit von Termen invariant unter $\beta\eta$-Konversion sein, damit durch diese die Wohlsortiertheit von Termen nicht verlorengeht. Wir haben auch hier einen Fall von gegenseitiger Abhängigkeit. Um diese Probleme zu lösen werden die rekursiv voneinander abhängigen Urteile für Wohlsortiertheit, Gültigkeit von Signaturen und sortierter $\beta\eta$-Reduktion in einem gemeinsamen Inferenzsystem (3.2.7) definiert. Auf diese Weise kann die Rolle der Terme in Termdeklarationen, die in [SS89] recht undurchsichtig war, in $\Sigma\Lambda$ vollkommen geklärt werden. Weiterhin können wir die Eigenschaft der Subterm-Abgeschlossenheit (Subterme von wohlsortierten Termen sind wieder wohlsortiert), die Schmidt-Schauß als Einschränkung fordern muß, in $\Sigma\Lambda$ ableiten (3.2.20).

Aufbauend auf diesem Inferenzsystem definieren wir Wohlsortiertheit von Substitutionen und zeigen, daß die Anwendung wohlsortierter Substitutionen (sogenannter $\Sigma$-Substitutionen) die Sorten von Termen erhält (3.4.4). Dies ist eine wesentliche Vorbedingung für den Beweis der gleichen Eigenschaft für sortierte $\beta\eta$-Reduktion (3.5.1). Unsere Grundoperationen in $\Sigma\Lambda$ sind also *sortenerhaltend*, das heißt, sie überführen wohlsortierte Objekte (Terme, Substitutionen, ...) wieder in wohlsortierte. Deswegen können wir uns in unserer Mechanisierung ausschließlich auf die Betrachtung von wohlsortierten Objekten beschränken. Damit haben wir also ein in sich abgeschlossenes logisches System definiert und die wichtigsten Operationen in $\Sigma\Lambda$ untersucht.

Ein Beleg für die Stärke der Deklaration von Sorteninformation über Termdeklarationen kann darin gesehen werden, daß in $\Sigma\Lambda$ der Begriff der Untersorte ein abgeleitetes Konzept ist. Semantisch gesehen stellt die Ableitbarkeit eines Urteils $[X{::}\mathbb{A}] \vdash_\Sigma X{::}\mathbb{B}$ (dieses Urteil besagt, daß unter der Voraussetzung, daß $X$ die Sorte $\mathbb{A}$ hat, hat $X$ auch die Sorte $\mathbb{B}$) sicher, daß alle Objekte der Sorte $\mathbb{A}$ auch die Sorte $\mathbb{B}$ haben, da Instantiierung sortenerhaltend ist. In anderen Worten, $\mathbb{A}$ ist eine Untersorte von $\mathbb{B}$. Es stellt sich heraus, daß alle Inferenzregeln für die Untersortenrelation, wie Reflexivität, Transitivität oder $\mathbb{A} \leq \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})$, aus den Inferenzregeln für Wohlsortiertheit ableitbar sind (3.6.9). Als eine Konsequenz daraus spielen Untersortenbeziehungen weder in der Unifikation noch hinterher im sortierten Resolutionskalkül eine Rolle, da sie durch die Behandlung der Termdeklarationen bereits mit abgedeckt werden.

## Unifikation in $\Sigma\Lambda$ (Kapitel 4)

In $\Sigma\Lambda$ stellte sich die Entwicklung von Unifikationsalgorithmen als ein schwieriges Unterfangen heraus, da die Mechanismen der $\lambda$-Konversion, der Sorten und der Termdeklarationen auf nichttriviale Weise interagieren. Herkömmliche Unifikationsmethoden berechnen partielle Lösungen, indem sie rekursiv die Struktur der jeweiligen Termpaare untersuchen. Da aber sortierte Unifikation auch die Sorteninformation eines Termpaares in Betracht ziehen muß, diese aber von der Struktur der Sortenherleitung abhängt, die wiederum von der Struktur des Terms unabhängig ist, müssen hier zusätzliche Hilfsmittel entwickelt werden.

Das Hauptwerkzeug für die Unifikation höherer Stufe ist der Begriff der *approximierenden Bindungen*, also der allgemeinsten Terme in der Klasse aller Terme einer gegebenen Sorte mit einem gegebenen Kopfsymbol. Da aber das Kopfsymbol im Gegensatz zur Sorte ein aus der Struktur eines Terms abgeleiteter Begriff ist, muß für die Berechnung von approximierenden Bindungen ein Zusammenhang zwischen Termstruktur und Sorte (oder der Sortenherleitung) hergestellt werden. Das theoretische Haupthilfsmittel dafür ist das Strukturtheorem (4.1.2), das für jeden Term $\mathbf{A}$ der Sorte $\mathbb{A}$ die Existenz von Herleitungen des Urteils $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ in einer speziellen Form garantiert. Obwohl die Struktur auch dieser Herleitungen, die wir *semi-strukturell* nennen, nicht direkt der Struktur des Terms $\mathbf{A}$ entspricht, reicht der strukturelle Zusammenhang doch aus, um approximierende Bindungen zu berechnen. Der Beweis, daß approximierende Bindungen wirklich die allgemeinsten Vertreter der Klasse der Terme gegebener Sorte und Kopfsymbols (4.2.4) sind, ist dann eine Konsequenz des Strukturtheorems. Für den Beweis des Strukturtheorems verwenden wir die sehr starke Technik der *logischen Relationen*, wie sie zum Beispiel aus Terminierungsbeweisen für $\beta$-Reduktion oder aus Schnitteliminationsbeweisen bekannt ist. Im Gegensatz zu einfach getypten $\lambda$-Kalkülen ist allerdings eine Eindeutigkeit von approximierenden Bindungen einer gegebenen Sorte nicht mehr garantiert, da jede Termdeklaration mit entsprechendem Kopfsymbol einen Beitrag leisten kann. Dieses Phänomen ist dabei keine Besonderheit des $\lambda$-Kalküls, weil es bereits bei Termen erster Ordnung, wie sie von Schmidt-Schauß [SS89] betrachtet wurden, auftritt. Mit dem Hilfsmittel der approximierenden Bindungen lassen sich dann im wesentlichen auf herkömmliche Weise [Hue76, Sny91] Unifikationsalgorithmen höherer Stufe bauen, deren Verzweigungsrate im Vergleich zum getypten Fall beschränkt ist durch die Anzahl gewisser Termdeklarationen. Wir führen diese Konstruktionen für die Fälle der allgemeinen $\Sigma$-Unifikation und der Prä-$\Sigma$-Unifikation aus.

In $\Sigma\Lambda$ tritt im Vergleich zum unsortierten Fall bei der Prä-Unifikation die Schwierigkeit auf, daß für flexible Paare, bei denen die Kopfvariablen funktionale Basissorten haben, im allgemeinen nicht gewährleistet ist, daß die Einsetzung mit konstanten Funktionen wohlsortiert ist. In diesem besonderen Fall können also gewisse flexible Paare nicht als gelöst angesehen werden und müssen auf herkömmliche Weise weiterverarbeitet werden. Wir verwenden später die Prä-$\Sigma$-Unifikation für unseren Resolutionskalkül, da dieser Unifikationsbegriff dafür ausreicht, aber einen wesentlich kleineren Verzweigungsgrad aufweist. Mithilfe der approximierenden Bindungen konnten wir zeigen, daß das Problem der Sortenberechnung in $\Sigma\Lambda$ äquivalent ist zu dem „higher-order matching"-Problem (4.3.5). Da die Entscheidbarkeit dieses Problems nur für beschränkte Formelklassen [Hue76, Dow92, Mil92] gesichert ist, müssen wir diese Frage für den allgemeinen Fall offen lassen.

### $\Sigma\mathcal{HOL}$: Eine sortierte Logik höherer Stufe

Für das logische System $\Sigma\mathcal{HOL}$ wird der sortierte $\lambda$-Kalkül $\Sigma\Lambda$ durch die Festlegung eines Basistyps $o$ und einer zugehörigen Basissorte $\mathbb{O}$ für Wahrheitswerte spezialisiert. Weiterhin werden Termdeklarationen für die logischen Konstanten $[q^{\mathbb{A}}::\mathbb{A} \to \mathbb{A} \to \mathbb{O}]$ für die Gleichheit, die Junktoren $[\neg::\mathbb{O} \to \mathbb{O}]$ und $[\wedge::\mathbb{O} \to \mathbb{O} \to \mathbb{O}]$ und zusätzlich der Quantor $[\Pi^{\mathbb{A}}::(\mathbb{A} \to \mathbb{O}) \to \mathbb{O}]$ als logische Konstanten mit fixierter Bedeutung angenommen.

Wir definieren und untersuchen für $\Sigma\mathcal{HOL}$ zwei verschiedene Semantikbegriffe, den der $\Sigma$-*Modellstrukturen* und den Spezialfall der *verallgemeinerten* $\Sigma$-*Modelle*. Der erste Begriff ist gerade die Spezialisierung der $\Sigma$-Strukturen auf den Kontext mit Wahrheitswerten und logischen Konstanten, der zweite spezialisiert diesen Begriff weiter auf den Fall von $\Sigma$-Algebren und ist deswegen der intuitive Semantikbegriff für Logiken höherer Stufe. Der wesentliche Unterschied der beiden Semantikbegriffe ist die Gültigkeit des Axioms $(\mathbf{A} \Leftrightarrow \mathbf{B}) \Rightarrow (\mathbf{A} = \mathbf{B})$ für Formeln $\mathbf{A}$ und $\mathbf{B}$ der Sorte $\mathbb{O}$, das eine Aussage über die Anzahl der Wahrheitswerte macht. Bedauerlicherweise kann dieses Axiom in keinem der bisher bekannten maschinenorientierten Kalküle zielorientiert behandelt werden. Weil es aber in allen verallgemeinerten $\Sigma$-Modellen gültig ist, haben wir diese zu $\Sigma$-Modellstrukturen verallgemeinert, um so einen Vollständigkeitsbegriff zu erhalten, der von der Anzahl der Wahrheitswerte unabhängig ist.

Durch diesen Semantikbegriff wird es uns möglich, modelltheoretische Vollständigkeitsaussagen für maschinenorientierte Kalküle höherer Stufe zu machen. In der Literatur sind bisher nur beweistheoretische Vollständigkeitsaussagen bekannt. Diese Aussagen betreffen die deduktive Stärke eines Kalküls im Vergleich zu einem anderen (Andrews Kalkül[1] $\mathfrak{T}$ aus [And71]) und beruhen im wesentlichen auf Andrews „Unifying Principle for Type Theory" [And71], das den Beweis der Vollständigkeit von Kalkülen auf die Überprüfung einiger natürlicher, beweistheoretischer Eigenschaften zurückführt. Da es für die Logik höherer Stufe kein einfaches Herbrand-Theorem gibt, sind die Unifying Principles zur Standardmethode für Vollständigkeitsbeweise höherer Stufe geworden.

Wir beweisen für unsere beiden Semantikbegriffe Unifying Principles, wobei das für die $\Sigma$-Modellstrukturen (5.4.18) auf Techniken aus [And71] beruht. Allerdings stärken wir den Begriff der abstrakten Konsistenz durch eine zusätzliche Saturiertheitsbedingung und erhalten so im Beweis totale Valuationen. Diese erlauben es uns den Beweis und die dahinterstehenden semantischen Objekte stark zu vereinfachen. Insbesondere lassen diese totalen Valuationen erst die Konstruktion eines Modells für abstrakt konsistente Satzmengen zu und führen daher zu der modelltheoretischen Variante des Unifying Principle. Aus diesem kann man dann mittels der Korrektheit von $\mathfrak{T}$ sofort ein beweistheoretisches Unifying Principle (5.4.19) wie in [And71] als Korollar erhalten.

Das Unifying Principle für verallgemeinerte $\Sigma$-Modelle (5.4.23) benötigt weitere Methoden zur Behandlung der Gleichheit. Diese Theoreme erlauben es uns nun sowohl die Vollständigkeit der Hilbertkalküle $\Sigma\mathfrak{T}\eta$ und $\Sigma\mathfrak{T}\mathfrak{E}$ (siehe 5.5.7 und 5.5.3) als auch Kompaktheitssätze (5.5.8) sehr einfach zu beweisen.

## Ein Resolutionskalkül für $\Sigma\mathcal{HOL}$

Durch die sortierte Unifikation lassen sich die bekannten Widerlegungskalküle für Logiken höherer Stufe [Hue72, Mil83, And89] zu sortierten Kalkülen für $\Sigma\mathcal{HOL}$ verallgemeinern [Koh94]. Wir führen das in dieser Dissertationsschrift für Huets Resolutionskalkül exemplarisch durch und erhalten so den sortierten Resolutionskalkül $\Sigma\mathcal{HR}$ für $\Sigma\mathcal{HOL}$. Dabei übernehmen wir aber Huets Kalkül nicht unverändert, sondern entwickeln ihn auch unabhängig von der Verallgemeinerung sortierte Logik weiter. Die im folgenden zusammengefaßten Methoden und Resultate sind im wesentlichen unabhängig von der Sortierung, da sich die Unterschiede zwischen sortierten und unsortierten Widerlegungskalkülen eigentlich auf die Unifikation beschränken und wir diese bereits gesondert abgehandelt haben.

[1] Dieser Kalkül ist nur im Sinne der $\Sigma$-Modellstrukturen und nicht im Sinne der verallgemeinerten $\Sigma$-Modelle vollständig.

Die *Skolemisierung* in Huets Originalkalkül war nicht korrekt, deswegen verwenden wir in unserem Kalkül $\Sigma\mathcal{HR}$ keine Skolemisierung, sondern verwalten die Abhängigkeiten zwischen existentiellen und universellen Variablen direkt in einer sogenannten *Variablenbedingung*. Unsere Methode beruht auf Arbeiten von Miller [Mil83, Mil92], der die Skolemisierung in Logiken höherer Stufe korrigiert hat, und Bibel, der solche Verfahren [Bib82] im Bereich der Konnektionsmethode erster Stufe untersucht hat.

Da die Unifikation höherer Stufe unentscheidbar ist, kann sie in einem Kalkül natürlich nicht ohne weiteres als Teilprozedur etwa der Resolutionsregel eingesetzt werden. Huets Lösung für dieses Problem besteht darin, in der Resolutionsregel die Literale nicht etwa direkt zu unifizieren, sondern als *Nebenbedingungen* (*Constraints*) zu behalten, und am Ende der Deduktion, wenn eine leere Klausel gefunden wurde, auf Lösbarkeit zu überprüfen. Diese Strategie ist natürlich für praktische Anwendungen so nicht verwendbar, da zwischen je zwei Literalen immer resolviert werden kann und nicht eine fortschreitende Instantiierung den Suchraum beschränkt. Deswegen lassen wir in $\Sigma\mathcal{HR}$ eine Mischung aus Unifikation (Constraint-Vereinfachung) und Resolution zu.

Dies erlaubt auch eine konzeptuell einfachere Behandlung von flexiblen Literalen, also Literalen, deren Kopfsymbol eine Variable ist. In $\Sigma\mathcal{HR}$ können wir nämlich anstelle von Huets Splittingregeln [Hue72] Andrews primitive Substitutionen [And89] verwenden. Diese Inferenzregeln erlauben die explizite Instantiierung von Prädikatsvariablen. Im Gegensatz zur Resolution erster Stufe kann die Unifikation in der Resolution höherer Stufe nicht alle notwendigen Instanzen generieren, da auch Prädikatsvariablen auftreten, für die im allgemeinen auch Ausdrücke eingesetzt werden müssen, die logische Konstanten enthalten.

Allerdings macht die Mischung aus Unifikation und Beweissuche für den Beweis der Vollständigkeit von $\Sigma\mathcal{HR}$ (6.4.4) ein starkes *Lifting-Lemma* (6.3.5) erforderlich, das bei Huets Beweis nicht notwendig war. Lifting-Lemmata sind aus Vollständigkeitsbeweisen für Widerlegungskalküle erster Stufe bekannte Teilschritte, die für jede Widerlegung einer Formel $\theta(\mathbf{A})$ eine solche für $\mathbf{A}$ garantieren. Mit ihrer Hilfe läßt sich die Frage nach der Vollständigkeit des Kalküls auf die Vollständigkeit von variablenfreien Formelmengen zurückspielen. Im Gegensatz zur Logik erster Stufe, wo sich durch Instantiierung nur die Terme in Klauseln ändern, kann sich in der Logik höherer Stufe durch die flexiblen Literale auch die Struktur der Klauseln ändern. Wird nämlich in die Kopfvariable eines flexiblen Literals ein Ausdruck mit logischen Konstanten eingesetzt, ist das Resultat kein Literal mehr und muß erst in Klauselnormalform überführt werden. Diese Strukturveränderung stellt sich als eine Schwierigkeit beim Beweis des Lifting-Lemmas heraus, weil es für das Anheben der Beweisschritte aus der Widerlegung von $\theta(\mathbf{A})$ wichtig ist, immer eine genaue Entsprechung der Klauseln und Literale in der Widerlegung von $\mathbf{A}$ zu behalten. An jeder Stelle, wo sich durch Instantiierung die Struktur der Klauseln in der Widerlegung von $\theta(\mathbf{A})$ ändert, muß die Entsprechung durch Anwendung von geeigneten primitiven Substitutionen in der Widerlegung von $\mathbf{A}$ wiederhergestellt werden. Die primitiven Substitutionen in $\Sigma\mathcal{HR}$ sind wieder spezielle approximierende Bindungen, deren Kopfsymbole allerdings logische Konstanten sein müssen, um die Aufgabe der Strukturveränderung erfüllen zu können. Durch das Vorhandensein von Termdeklarationen in $\Sigma\mathcal{HOL}$ tritt hier der einzige wirkliche Unterschied zwischen dem sortierten und dem unsortierten Fall auf. Da die approximierenden Bindungen in $\Sigma\mathcal{HOL}$ mehr als eine logische Konstante enthalten können, kann der Beweis des Lifting-Lemmas an dieser Stelle nicht der Struktur von $\theta$ folgen, wie es im unsortierten Fall möglich wäre.

## Rückblick

Um ein besseres Verständnis für die praktischen Vorteile des Sortenmechanismus in $\Sigma\mathcal{HOL}$ zu bekommen, wollen wir uns nun ansehen, wie sich Sorten auf die Möglichkeiten auswirken, mathematische Sachverhalte in Logiken höherer Stufe zu kodieren. Mit der aus Logiken erster Stufe wohlbekannten Technik der Relativierung lassen sich sortierte Formeln $\mathbf{A}$ in unsortierte Formeln $\mathbf{Rel}(\mathbf{A})$ transformieren. So transformiert sich zum Beispiel ein quantifizierter Ausdruck der Form $\forall X_{\mathbb{A}}.\mathbf{A}$ nach $\forall X_{\tau(\mathbb{A})}.(\mathbf{P}_{\mathbb{A}}X) \Rightarrow \mathbf{Rel}(\mathbf{A})$, wobei $\mathbf{P}_{\mathbb{A}}$ ein neues Prädikatensymbol ist, das die Menge $\mathbb{A}$ beschreibt. Diese neuen Prädikate bekommen ihre Bedeutung durch eine Axiomatisierung $\mathbf{Rel}(\Sigma)$ der Sorteninformation aus der Signatur $\Sigma$, die gleichzeitig mit der Relativierung bereitgestellt werden muß. Leider ist die Situation in $\Sigma\mathcal{HOL}$ nicht so einfach wie für Logik erster Stufe, wo Quantifikation das einzige syntaktische Konstrukt ist, das Variablen bindet und deswegen die Implikation als Mittel zur Beschränkung des Gültigkeitsbereichs ausreicht. Für die Relativierung von $\lambda$-Abstraktionen benötigen wir eine Form von Konditionalen, da schon wegen des Typs nicht die Implikation benutzt werden kann.

Für die meisten sortierten Logiken (zum Beispiel [Wal87, SS89, Wei91]) ist eine wohlsortierte Formel $\mathbf{A}$ genau dann gültig in der Klasse der sortierten Modelle, wenn $\mathbf{Rel}(\mathbf{A})$ gültig ist in der Klasse der Modelle, die $\mathbf{Rel}(\Sigma)$ erfüllen. Von einem theoretischen Standpunkt aus gesehen sind also diese sortierten Logiken nicht expressiver als die unsortierte Logik erster Stufe, außerdem lassen sich sortierte Logiken durch das Zusammenwirken von Relativierung und unsortierter Deduktion mechanisieren. Da sich Konditionale aber erst in Logiken höherer Stufe mit Beschreibungsfunktionen definieren lassen und es für diese bisher keine zielorientierten Mechanisierungen gibt, steht uns die Möglichkeit der Mechanisierung von $\Sigma\mathcal{HOL}$ durch Relativierung und Verwendung von unsortierten Beweisern nicht wirklich offen. In diesem Sinne ist die Mechanisierung von $\Sigma\mathcal{HOL}$ also wirklich ausdrucksstärker als der einfach getypte $\lambda$-Kalkül.

Auch schon ein Vergleich der Größe und Komplexität einer $\Sigma\mathcal{HOL}$-Formelmenge mit ihrer Relativierung, wie wir ihn in 7.2 durchgeführt haben, zeigt deutlich, daß die Suchräume für die relativierte Formelmenge so groß werden, daß an eine automatische Behandlung nicht zu denken ist. Mit diesem Vergleich kann die Verwendung von Sortentechniken im automatischen Beweisen höherer Stufe als eine Verfeinerung angesehen werden, die es erlaubt, spezielle Klassen von Axiomen der Form $(\forall \overline{X^k}.(p^1X^1) \Rightarrow \ldots \Rightarrow (p^kX^k) \Rightarrow (q\mathbf{A}))$, wobei die $p^i$ und $q$ einstellige Prädikate und die $X^i$ die freien Variablen von $\mathbf{A}$ sind, als Termdeklarationen $[\forall \overline{[X^k :: \mathbb{P}^k]}.\mathbf{A} :: \mathbb{Q}]$ zu fassen und algorithmisch in der Unifikation zu behandeln.

## Ausblick

Natürlich haben wir in dieser Arbeit nicht alle Fragen, die sortierten Logiken höherer Stufe und ihre Mechanisierung betreffen, lösen können. Im Folgenden sollen daher einige vielversprechende, weitergehende Forschungsthemen skizziert werden.

Es wäre wünschenswert, den Sortenmechanismus in $\Sigma\mathcal{HOL}$ um Sortenkonstruktoren zu erweitern und so eine direkte Beschreibung von Funktionenräumen, wie zum Beispiel $\mathcal{C}^i(\mathbb{R}, \mathbb{R})$ in der Analysis, zu ermöglichen. Weiterhin wäre die Erweiterung des Sortenmechanismus um Schnittsorten (vergleiche [KP93]) interessant, da sie die Probleme mit der Regularität von Signaturen vereinfachen würde.

In der Logik erster Stufe haben sortierte Unifikationsalgorithmen mit Termdeklarationen zur Entwicklung von speziellen Resolutionskalkülen mit dynamischer Sorteninformation [Wei91, Wei93] geführt. Diese Systeme erlauben die Spezifikation von bedingter Sorteninformation, wie sie zum Beispiel in der Definition der Stetigkeit von Funktionen ($[f::\mathbb{C}] \Leftarrow \forall\epsilon.\exists\delta.\ldots$) auftritt. Dadurch kann im Verlauf der Beweissuche neue Sorteninformation berechnet werden, indem diese Bedingungen bewiesen werden. Diese wird dann sofort durch den Unifikationsalgorithmus genutzt und trägt so zur stetigen Verbesserung der deduktiven Stärke des Kalküls bei. Es scheint, daß sowohl der Mechanismus der bedingten Sortendeklarationen als auch die Resolution mit dynamischer Sorteninformation für die Mechanisierung der Mathematik sehr interessant sein können, insbesondere wenn sie auf Logiken höherer Stufe verallgemeinert werden.

Die Resolutionskalküle mit dynamischen Sorten wurden in [KK94] ihrerseits eine Grundlage für die Mechanisierung von partiellen Funktionen basierend auf den bekannten dreiwertigen Kleene-Logiken [Kle52]. Wir hoffen diese Ansätze für $\Sigma\mathcal{HOL}$ verallgemeinern zu können und damit Kalküle zu haben, die mit der Behandlung von Sorten und partiellen Funktionen einen signifikanten Teil der mathematischen Umgangssprache adäquat abdecken. Solch ein logisches System mit Sorten und partiellen Funktionen ist natürlich für die Mathematik noch besser geeignet als $\Sigma\mathcal{HOL}$, denn nur ein Teil der in der Mathematik auftretenden Funktionen haben Definitionsbereiche, die Cartesische Produkte sind und sich daher einfach als Sorten fassen lassen. So ist es zum Beispiel weit natürlicher, die Funktion $f(x,y) := \frac{1}{x-y}$ als partielle Funktion der Sorte $\mathbb{R} \to \mathbb{R} \to \mathbb{R}$ aufzufassen, die nur für $x \neq y$ definiert ist, als für sie eine Wertebereichssorte zu konstruieren.

Da der intuitive Semantikbegriff für Logiken höherer Stufe die verallgemeinerten $\Sigma$-Modelle sind, wäre es wünschenswert, in der Zukunft Resolutionskalküle zu entwickeln, die im Sinne der verallgemeinerten $\Sigma$-Modelle vollständig sind. Wir stellen in 7.3 eine Idee vor, wie man das Axiom der Wahrheitswerte in einem Resolutionskalkül zielorientiert mechanisieren könnte. Wir hoffen mit unserem Unifying Principle für $\Sigma$-Modelle, ein Beweishilfsmittel bereitgestellt zu haben, das den Vollständigkeitsbeweis eines solchen Kalküls erleichtert.

Das explizite Lifting-Lemma für $\Sigma\mathcal{HR}$ erlaubt es, realistische Suchstrategieen zu untersuchen, wie sie für Resolutionskalküle erster Stufe bekannt sind. Diese Strategien sind für die praktische Einsetzbarkeit von Resolutionsbeweisern höherer Stufe unabdingbar, aber bisher weder theoretisch noch praktisch untersucht worden.

Die in dieser Arbeit vorgestellten Methoden, Kalküle und Algorithmen bilden eine theoretische Basis für die Entwicklung von konkreten, automatischen Beweisern höherer Stufe und damit für Deduktionssysteme für die Mathematik. Eine Implementation ist im Moment an der Universität des Saarlandes im Rahmen des Sonderforschungsbereichs 314 „Künstliche Intelligenz – Wissensbasierte Systeme“ in Arbeit. Die Logik $\Sigma\mathcal{HOL}$ ist die Basis der Arbeitssprache $\mathcal{POST}$ (**P**artial **O**rder-**S**orted **T**ype theory) in der sich mathematische Sachverhalte – beispielsweise eines typischen mathematischen Lehrbuchs – für das Omega-System ausdrücken lassen. Bei diesem System handelt es sich um ein interaktives Beweissystem, wie wir es am Anfang dieser Zusammenfassung beschrieben haben. Die umfangreich Erfahrung in der Kodierung mathematischer Probleme für traditionelle automatische Beweiser erster Stufe, und die Kritik der Repräsentationssprache, die als zu schwach empfunden wurde, waren eine wichtige Motivation für diese Arbeit. Erste Experimente mit dem neuen System haben gezeigt, daß sortierte Logiken höherer Stufe tatsächlich ein ausreichend mächtiges System darstellen um einen nichttrivialen Teil der Mathematik adäquat zu formalisieren. In diesem Zusammenhang verwirklichen wir nun auch unseren Resolutionskalkül $\Sigma\mathcal{HR}$ im $\mathcal{LEO}$ (**L**ogic **E**ngine for OMEGA) System.

# Contents

# 1 Introduction

The field of mathematical logic has its roots in the effort of understanding the process of rational human reasoning. Since mathematical reasoning is human reasoning in its purest and most rigorous form, it is the most natural object for the investigation of this process.

If we look at the history of mathematics, we can observe a recurring pattern of change. Consider, for instance, the case of calculating with numbers, a task that has changed from a difficult job for highly paid specialists in Roman times to a task that is feasible for children in our century. What is the cause of this dramatic change? Of course the formalized reasoning procedures for arithmetic that we use nowadays. These so-called *calculi* consist of a set of rules that can be followed purely syntactically, but nevertheless manipulate arithmetic expressions in a correct and fruitful way. An essential prerequisite for syntactic manipulation is that the objects are given in a formal language suitable for the problem. For example, the introduction of the decimal system has been instrumental to the simplification of arithmetic mentioned above. When the arithmetical calculi were sufficiently well understood and in principle a somewhat mechanical procedure, and when the art of clock-making was mature enough, to design and build mechanical devices of an appropriate kind, the invention of calculating machines for arithmetic by Schickard (1623), Pascal (1642), and Leibniz (1671) was only a natural consequence.

Another important step for understanding the human reasoning process was the observation Aristoteles, and other Greek philosophers, and later by Leibniz, Boole, and Frege, that mathematical methods (calculization) can be applied to the reasoning process itself. In particular, mathematical reasoning can be carried out by syntactically applying simple rules to formal expressions just like in the case of arithmetical calculi. This idea, to develop calculi for reasoning in analogy to those for arithmetic, has strongly influenced the development of formal languages, notions of semantics, and modern logical calculi. Just as the discovery of efficient calculi for arithmetic has led to the development of mechanical calculators, the discovery of logical calculi has led to the development of todays *deduction systems*, which are computer programs that perform reasoning tasks by operationalizing these calculi. The systems built so far can be roughly categorized into the paradigms of interactive and fully automatic systems that aim at finding proofs for theorems with or without user interaction.

*Automated theorem provers* are usually based on refutation calculi that try to prove a theorem by deriving a contradiction from its negation and rely on unification [Rob65] as a central inference procedure. Unification algorithms (see [BS94] for a comprehensive survey) compute substitutions that equate given sets of terms. For refutation purposes only such substitutions (called most general) are needed from which all others can be recovered by instantiation. This property allows the refutation procedure to search for schematic proofs that represent all instances of the proof. Such refutation procedures are conceptually and computationally very simple, if the used sub-procedures, such as normal form reductions and unification, are decidable and of relatively low complexity. Therefore almost all automated theorem provers restrict the input language to some variant or subsystem of first-order logic, where unification is decidable and yields unique, most general unifiers.

The primary emphasis of research for fully automated systems lies in finding strong calculi and search strategies that restrict the search spaces associated with proof search.

There are three classes of refutation calculi for automated theorem proving that turned out to be of primary importance, namely, resolution [Rob65, OS89], the mating/connection method [And81, Bib83], and analytic tableaux [Smu68, Fit90]. The resolution method has been further refined for a special treatment of the equality predicate in the Paramodulation [RW69], $E$-resolution [Mor69], and RUE-resolution [Dig79] calculi. Term Rewriting [KB70] systems, which have at first been developed for pure equational logic, have since been generalized to full first-order logic in the superposition calculi [ZK88, BG90, BG92].

Automated theorem proving systems based on any of the calculi above have reached the power to solve non-trivial problems, but they are (like all search procedures) subject to the combinatorial explosion of the search spaces, and therefore have principal limits to the complexity of proofs that can be found. Thus in general they can approach the efficiency of mathematicians only in domains, where humans have very little or no intuition at all.

These limits have led part of the research community to investigate systems that rely on user interaction to find proofs. Since universal proof procedures do not play an important role in this paradigm, the community has developed highly expressive logical formalisms and calculi that are modeled after human reasoning in mathematical practice. This is essential for the concept of *interactive deduction systems*, because an expressive language allows for adequate formalizations of mathematical practice and also for short proofs that can be easily communicated. The AUTOMATH Project [dB80] has pioneered the area of mechanical proof checking by coding the total contents of a mathematical textbook [Lan30] in a formal language and proving all theorems in the accompanying proof system [Jut79].

Unfortunately, typical proofs in this and other systems are still so long and complex that almost all practical interactive systems provide a so-called “tactic mechanism” (giving rise to the name “tactic theorem proving”), which allows the user to write small specialized automatic reasoning procedures to relieve him of some of the routine work. Among the most influential tactic theorem provers are the Nuprl [CAB+86], the HOL [Gor85, GM93], the PVS [ORS92], and the KIV [HRS90, HRS91] systems, which were originally designed for program and hardware verification. Isabelle [PN90] and Elf [Pfe91] are examples for deduction systems that mechanize logical frameworks, i.e. systems where the logic language is powerful enough to allow the specification of object logics.

The motivation for the work reported in this thesis comes from the attempt to develop deduction systems that inherit the merits of both approaches to theorem proving, the interactive and the fully automatic one [HKK+92, HKK+94]. If deduction systems are to be useful as assistants to mathematicians or in software verification, we will need the expressive formalisms (currently found in interactive systems) and the strong universal proof procedures (without the restriction to first-order logic), since otherwise formal deduction will be too tedious and expensive in practice. Clearly the solution cannot be an automated theorem prover for expressive logics, since it is maybe even more subject to the combinatoric explosion. The author believes that a possible solution might be an interactive deduction system that has access to powerful automated theorem proving procedures (logic engines, such as resolution systems) to fill non-trivial gaps in the proofs the user develops interactively with the system. These logic engines must be automatic systems that can routinely solve non-trivial problems given in the expressive language of the interactive system calling them.

The concrete goal of this thesis is to develop a mechanization of a very expressive logical formalism that is suited for the use in mathematics. As mathematical facts cannot be adequately formalized in first-order logic and first-order automated theorem proving has shown that sort techniques allow for an efficient automatic proof search, we have chosen a sorted higher-order logic for the task. The sorted higher-order resolution calculus developed in this thesis is intended to be a basis for logic engines that are suited for the task described above.

In the remainder of this introduction we will have a look at higher-order logic and deduction, and sorted first-order deduction systems to motivate the features of our common generalization. Then we will review related work and finally outline the results and structure of this thesis.

## 1.1 Higher-Order Logic

At the beginning of this century mathematicians applied the newly developed logical methods to mathematics itself (e.g. [WR10]) and thereby tried to provide it with a secure logical foundation. A solution of this *"Grundlagen" problem* requires a formal language to express all mathematical statements and a consistent logical calculus that can formally derive all true statements. This endeavor turned out to be much more difficult than expected, for instance the well-known Russell-antinomies show that special precautions must be taken in order to prevent inconsistencies. Russell already suggested the use of typed logics [Rus08] as a possible remedy and used this approach together with Whitehead in the Principia Mathematica [WR10]. *Typed logics* classify objects by assigning a certain type to each object, and by restricting term formation to well-typed formulae. To be well-typed objects of functional type can only be applied to arguments whose type matches the type of their domain. The hierarchy of types naturally induces the notion of an order on types and objects.

In [Göd30] Gödel proved the completeness of a subsystem of the logic underlying Principia Mathematica (the so-called *first-order logic*), which allows quantification only over first-order variables (individuals). In contrast to this we will call logical systems that allow quantification over variables of arbitrary order *higher-order logics*. Unfortunately, first-order logic is so weak that neither Peano arithmetic nor, for instance, the theory of torsion groups[2] can be finitely axiomatized. Gödels later result [Göd31], stating that the full system of Principia Mathematica (and indeed any logical system that can formalize Peano arithmetic) is incomplete, has led to the dominance of first-order logic that we still experience today, even though only a fragment of mathematics can be adequately expressed. First-order logic is even more appealing, because the technical device of typing can be left implicit by distinguishing "terms" (denoting individuals) and "propositional formulae" (denoting truth values).

Zermelo, Fraenkel, Gödel, and others used an encoding of mathematics into set theory [Zer08, Fra28, Neu28, Göd40, Ber41] that is itself axiomatizable in first-order logic to give an answer to the foundation problem for mathematics that is accepted by most mathematicians today. It is appealing, since it only uses first-order calculi and thus inherits all nice properties of first-order logic. Consequently, these ideas have been the basis for attempts to build automated theorem provers for mathematics [BLM+86, Qua92]. Note

[2]Torsion groups are groups, where for any element $a$, there is a natural number $n$ such that $a^n$ is the neutral element.

that these systems must be incomplete, as Peano arithmetic can be formalized in them via the encoding.

Mathematical vernacular usually uses a mixture of both: set theory and typed higher-order logics. Set theory provides a powerful tool for describing mathematical objects, and proofs are carried out in a logic implicitly typed by the choice of notation ($n$, $m$, $k$ for natural numbers; $f$, $g$, $h$ for functions ...). Furthermore, explicit quantification over variables of higher type is widespread. Naturally, the opportunity to encode all of this into set theory is almost never used, since the encodings of the objects of interest become much too large and quite unwieldy. On the contrary, deduction is in general carried out in the respective technical language, that has been established for the particular mathematical field in question. Thus, if we view axiomatic set theory only as an answer to the foundation problem of mathematics, where the aim consists in demonstrating that the whole body of mathematics can be encoded into a consistent logical system, the problem can be viewed as solved and the technical inconvenience that, for instance, basic mathematical objects like functions have to be encoded as right total, left unique relations are irrelevant. This need for encoding into axiomatic set theory is clearly an obstacle for giving an adequate account of informal mathematical practice. The author believes that the choice of a typed higher-order logic is much more natural for the use in deduction systems, since it takes those objects as primitive that most mathematicians consider as basic. Moreover, we can also use axiomatic set theory in higher-order logic, since it contains first-order logic as a subsystem.

The expressiveness of the logical systems discussed so far (higher-order logics as well as axiomatic set theories) is guaranteed by the so-called *comprehension axioms*, which postulate the existence of all functions that can be expressed by well-formed formulae parameterized by free variables. Unfortunately, this infinite set of axioms makes a direct use of higher-order logic for the mechanization of mathematics impossible, because an automated theorem prover would have to be interactively supplied with the subset of comprehension axioms relevant for the problem at hand. Thus significant guidance would be left to the user of a deduction system, since the choice of the appropriate comprehension axiom (for instance, postulating the existence of a diagonal sequence in the proof of Cantor's theorem) is often a key idea to the proof.

This was one of the reasons for Church to reformulate higher-order predicate logics to the (logically equivalent) system of *simple*[3] *type theory*. In this logical system the comprehension axioms are cast in an equality theory, which can even be directed to a confluent terminating reduction system, and is therefore decidable. We can understand this reformulation by the following argument. Take an instance of the original comprehension axiom $\exists F.\forall X.FX = \mathbf{A}$, where $\mathbf{A}$ is an arbitrary formula, and give the function $F$, which is guaranteed by this axiom, the name $\lambda X.\mathbf{A}$, then we are left with the assertion that $\forall X.(\lambda X.\mathbf{A})X = \mathbf{A}$. This can be instantiated by an arbitrary formula $\mathbf{B}$ to $(\lambda X.\mathbf{A})\mathbf{B} = [\mathbf{B}/\mathbf{X}]\mathbf{A}$ which is just the definition of $\beta$-equality. Thus Church's simply typed $\lambda$-calculus is a very elegant and intuitive formulation of higher-order logic, since the syntax and semantics of typed predicate logics can be reobtained by simple definitions. Therefore we base the work reported in this thesis on simple type theory.

Independently from its logical origins the simply typed $\lambda$-calculus has become one of the

[3] In fact, Church's type system is a simplification of Russell's system of ramified types [Rus08].

most important tools of computer science for describing functions, programming languages, and more generally computability. This has spawned the development of a rich zoo of specialized type systems for various $\lambda$-calculi (see for instance [Tho91]). Although the name "type theory" has originally been used by Church for his logical system (i.e. the simply typed $\lambda$-calculus augmented by logical constants and axioms), it has become customary to refer with this name to $\lambda$-calculi with powerful type systems but without logical constants or axioms. We will stick to this usage and call our $\lambda$-calculi higher-order logics whenever logical constants and axioms are present.

We also want to mention an alternative, equivalent formulation of higher-order logic. *Combinatory logic* was developed by Schönfinkel [Sch24], and then thoroughly investigated by Curry and Feys [CF58] (for a modern treatment see [HS86]). In combinatory logic the role of $\lambda$-abstraction and $\beta$-conversion is taken up by the combinators $K, S, I$, and the axioms of weak combinatory reduction: $I\mathbf{A} \to \mathbf{A}$, $K\mathbf{AB} \to \mathbf{A}$ and $S\mathbf{ABC} \to (\mathbf{AC})(\mathbf{BC})$. The simply typed version of this system of higher-order logic is equivalent to the simply typed $\lambda$-calculus, since each combinatory logic formula can directly be translated into a $\lambda$-calculus formula and vice versa. Moreover, the equality theories are coextensive modulo this translation. Combinatory logic has one great technical advantage: there is no need for $\lambda$-abstractions, and consequently, bound variables are not a problem. But it has the practical disadvantage that formulae are much more difficult to read for humans. While there have been attempts to mechanize mathematics on the basis of combinatory logics [Rob69b, Joh91, DJ92, Dou93, Joh93, Koh93], this is not the subject of this thesis, although it may become important in the future.

## 1.2 Higher-Order Automated Theorem Proving

The history of building automated theorem provers for higher-order logic is almost as old as the field of deduction systems itself. In fact, one of the first attempts to build a semi-automated deduction systems (SAM [BEG+64, Gua64, Gou65, GOBS69]) did not restrict itself to first-order logic but instead used higher-order logic.

When evaluating calculi for higher-order logic the classical notion of completeness becomes problematic, since higher-order logic cannot admit complete calculi according to Gödel's first incompleteness theorem [Göd31] as mentioned above. At closer view, Gödel's theorem only applies to the so-called standard semantics, where a model consists of a given universe $\mathcal{D}_\iota$ of individuals, the set $\mathcal{D}_o$ of truth values, and universes $\mathcal{D}_{\alpha\to\beta}$ for the function types that are just the sets of all functions with domain in $\mathcal{D}_\alpha$ and codomain in $\mathcal{D}_\beta$. While this semantics is indeed the intuitive semantics for mathematics, it does not necessarily yield a reasonable measure for the completeness of a calculus. If we consider a generalized notion of model theory, the so-called *general models*, where the universes of functional type are only required to be subsets of the set of all functions such that there exists a denotation for any well-formed formula[4], then appropriate generalizations of first-order calculi are complete [Hen50]. Clearly each standard model is a general model. Moreover, there are now so many new models, that all propositions that are valid but not provable (in the standard sense) now have a counterexample. Furthermore, by Gödel's second incompleteness theorem formal methods cannot characterize standard models in the class of general models. Thus this so-called generalized (or Henkin)-semantics yields an appropriate

[4] Note that this requirement directly corresponds to the comprehension axioms.

measure of completeness for higher-order calculi. Fortunately, the corresponding notion of soundness entails that of standard soundness, since each standard model is a general model by definition.

A wide range of methods for higher-order automated theorem proving has been proposed. In [Rob68, Rob69a] Robinson presented a proof procedure that is essentially a tableau implementation of the calculi given in [Sch60, Tak53]. A similar procedure was later implemented in [Hib73] and successfully applied to problems from number theory. In [Rob69b] Robinson proposes to translate a problem given in higher-order logic into combinatory logic and to give it to a conventional first-order Resolution/Paramodulation theorem prover that has also been given an axiomatization of combinatory logic. In [Dar71] Darlington presents a resolution procedure that allows limited second-order formulae in order to handle induction schemata. He employs the unification algorithms from [Gou66]. Andrews proposed a resolution calculus for full higher-order logic [And71], and for the completeness proof pioneers the use of a unifying principle for higher-order logic. This technique is probably more important than the particular calculus itself, which lacks unification just like the calculi discussed so far, and is therefore not practically applicable. The unifying principle of [And71] has become the standard method for proving completeness of higher-order calculi, and we will use, extend and simplify it in this thesis.

The first successful attempts to mechanize and implement higher-order logic were those of Huet [Hue72] and Jensen and Pietrzykowski [Pie73, JP73, JP76]. They combine the resolution principle with *higher-order unification*, which we now discuss in more detail. The unification problem in typed $\lambda$-calculi is naturally much more complex than that for first-order terms, since it has to take the theory of $\lambda$-equality into account. This problem was first investigated in depth by Gould in [Gou66], who already identified the problem as nullary[5]. Even though Gould's unification algorithms (and also those of Darlington [Dar68] and Ernst [Ern71]) are not complete, the early work identified the major difficulties, and lead to the solution of the problem by Huet, Jensen, and Pietrzykowski [Hue72, Hue76, JP73, JP76]. Huet proved that third-order unification is undecidable [Hue73], a result that was independently obtained by Lucchesi [Luc72], refined by [Bax78], and finally extended to second-order logic [Gol81, Far91a]. The last result gives a sharp classification of the undecidability of higher-order unification: if the language has one binary function constant, then unification is undecidable; if there are only unary function symbols, then unification is decidable: it can easily be seen to be equivalent to associative unification, which is decidable [Mak77], and has at most infinitely many most general unifiers [Plo72].

The first implementations of higher-order unification already revealed that the search space for unifiers is far too large to be feasible for practical applications. Huet developed a restriction of the higher-order unification problem (*pre-unification*) that is sufficient for the completeness of refutation procedures, where one is only interested in the solvability of unification problems rather than in the unifiers themselves. Although the pre-unification problem is also undecidable, a simple modification of Huet's algorithm enumerates sets of most general solutions to unifiable problems. Moreover the application to almost all practical problems yields small sets of most general unifiers. For a modern presentation of higher-order (pre-)unification we refer the reader to [SG89, Sny91].

[5]A unification problem is called nullary, if complete sets of unifiers need not always have most general elements.

Another research topic in the field of higher-order unification that is motivated by practical applications is the search for subclasses of higher-order formulae that enjoy a tractable unification problem. A subclass that is particularly interesting because its unification problem is decidable, unitary [Mil92] (solvable unification problems always have unique most general unifiers), and linear [Qia93] (that can be computed in linear space and time) is that of *higher-order patterns* introduced by Miller for the higher-order logic programming language $\lambda$-PROLOG [Mil91]. This subclass has since proven its usefulness for higher-order equality reasoning [Nip91, Pre94b, Pre94a] and logical frameworks [Pfe91]. In some cases it is possible to relax the conditions of higher-order patterns and still obtain a decidable unification problem. For instance, unification of pairs consisting of one higher-order pattern and one second-order formula are still decidable [Pre94b, Pre94a].

The question of higher-order unification, where the theory of $\beta\eta$-equality has been augmented by a further equational theory such as associativity or commutativity, has led to algorithms for general- [Sny90] and modular *higher-order $E$-unification* [NQ91, Mül93, MW94, Web93]. The modular algorithms are, in fact, more of a combination method that allows to combine higher-order unification with existing first-order unification algorithms. For an application of the associative commutative higher-order patterns see [QW94]. In [Joh91, DJ92] the methods developed by Dougherty for unification with combinators have been extended to higher-order $E$-unification by employing combination methods for first-order narrowing.

A totally different approach to higher-order unification is taken by Dougherty in [Dou93], where he gives a unification algorithm for combinatory logic that is based on a first-order narrowing method for the theory of weak combinatory equality.

In contrast to the higher-order unification problem, where the issue of decidability is basically well-understood, the question of decidability of *higher-order matching* (for given formulae $\mathbf{A}$ and $\mathbf{B}$ find a substitution $\sigma$ such that $\sigma(\mathbf{A})=_{\beta\eta}\mathbf{B}$) is still largely open. In [Hue76] Huet was able to prove decidability of second-order matching and recently Dowek extended this result to third-order formulae [Dow92]. Even though various authors have studied other special cases [Zai87, Wol93, CQ94, Cur93] the decidability of the general higher-order matching problem is still open.

Granted a good understanding of higher-order unification, higher-order theorem proving is still a complex issue. We now discuss some of these problems using the example of higher-order resolution. The same problems, however, also appear in the context of other higher-order refutation procedures, such as the higher-order matings method [And89]. Since higher-order unification is undecidable, incorporating unification into the resolution inference rule would not result in an effectively computable rule. As a remedy, the unification process can be delayed by capturing the unification problems as constraints and effectively interleaving the search for empty clauses by resolution with the search for unifiers. Finally, in contrast to first-order refutation theorem proving not all instantiations that are necessary in a refutation can be obtained by unification, since the heads of flexible literals (i.e. literals, where the head is a predicate variable) have to be instantiated with formulae that contain the logical constants $\wedge, \neg, \Pi$. Clearly these substitutions cannot be found using unification, since the needed head symbols are not even present in the clause set, as they have been eliminated in the clause normal form transformation. Huet solves this problem by introducing special "splitting" inference rules that provide the instantiations by enumerating all possible substitutions. This approach can hardly be called practical, since these inference rules are infinitely branching. Unfortunately, a better solution for the general problem remains still to be found. It seems probable that Bledsoe's "set variables" method [Ble77, Ble79] from the context of set-theoretic theorem proving might give some heuristics or even provide a direction towards a complete mechanization.

While experiments like the TPS-project [ALCMP84, And89, AINP90] of Andrews at the Carnegie Mellon University have shown the practical feasibility of higher-order automated theorem proving based on these ideas, such systems are rather weak in their deductive power when compared to automated theorem provers for first-order logic. Part of this weakness stems from the fact that higher-order deductive systems have to treat problems (like complex unification problems and the problem of flexible literals) that are intrinsic to higher-order logic and slow down the proof search. The other major cause of inefficiency is the fact, that most technological advances in first-order theorem proving, such as $E$-unification, sorts, sophisticated search strategies, special methods for equality, as well as implementational progress, such as term indexing have not yet found their way into higher-order theorem proving or are only beginning to be investigated recently. The author believes that the obstacles to proof search intrinsic to higher-order logic may well be compensated by the greater expressive power of higher-order logic and by the existence of shorter proofs. Thus higher-order automated theorem proving will be practically as feasible as first-order theorem proving is now as soon as the technological backlog is made up.

The work reported in this thesis is intended to fill this gap at least with respect to the treatment of sorts by generalizing the first-order sort techniques of Schmidt-Schauß [SS89] to higher-order logic.

## 1.3 Sorts in First-Order Deduction

The introduction of *sorted logics* has been one of the most successful contributions to first-order automated deduction. Sort techniques consist in syntactically distinguishing between objects of different semantic classes (foxes, wolves, numbers, points, lines, etc.); the essential idea behind sorted logic is to assign sorts (specifying the membership in some class) to objects and restrict the range of variables to particular sorts. Sorted logics have already been studied very early from a theoretical point of view by Herbrand [Her30], Schmidt [Sch38, Sch51], Wang [Wan52] and Oberschelp [Obe62]. The practical exploitation of sort information in the search for proofs can dramatically reduce the search space associated with theorem proving (see e.g. [Wal85]), and hence the resulting sorted calculi are much more efficient for deduction purposes. In the context of first-order logic sort information has been successfully employed by Walther [Wal83, Wal85, Wal88], Schmidt-Schauß [SS86, SS89], Cohn [Coh87, Coh89, Coh92], Frisch [Fri90, CF92], Weidenbach [Wei89, Wei91, Wei93], and others.

In unsorted logics the only way to express the knowledge that an object is a member of a certain class of objects is through the use of unary predicates, such as the predicate $\mathbb{N}$ in the formulae ($\mathbb{N}2$), i.e. "2 is a natural number" or $\neg(\mathbb{N}\text{Peter})$ with the meaning "Peter is not a natural number". This leads to a multitude of unit clauses ($\mathbb{S}\mathbf{A}$) in the deduction that only carry the sort information for $\mathbf{A}$. Furthermore, in unsorted logics quantification is unrestricted, whereas in practice one often wants quantifications to range only over the objects in a certain class. The latter kind of quantification can always be formulated with formulae like $\forall X.(\mathbb{N}X) \Rightarrow (\geq X 0)$. But the approach is unsatisfactory, because inter alia the derivation of the nonsensical formula $(\mathbb{N}\mathrm{Peter}) \Rightarrow (\geq \mathrm{Peter}\, 0)$ is permitted, even though $(\geq \mathrm{Peter}\, 0)$ can never be derived because of $\neg(\mathbb{N}\mathrm{Peter})$.

Sorted logics remedy this situation by assigning sorts to constants and variables and require that formulae meet certain restrictions to denote meaningful objects: an application $(\mathbf{AB})$ is *well-sorted*, iff there are sorts $\mathbb{A}$ and $\mathbb{B}$ such that $\mathbf{A}$ is of sort $\mathbb{B} \to \mathbb{A}$ and $\mathbf{B}$ is of sort $\mathbb{B}$. In this case the sort of $(\mathbf{AB})$ is $\mathbb{A}$. Furthermore, sorted logics provide mechanisms for restricted quantification, where the truth value of a formula $\forall X_{\mathbb{A}}.\mathbf{A}$ depending on the instances of $\mathbf{A}$ with respect to objects of sort $\mathbb{A}$. Thus in a sorted logic the quantified formula above would read $\forall X_{\mathbb{N}}.(\geq X\, 0)$ where $\geq$ would be declared to be a binary relation on $\mathbb{N}$ and $0$ to be of sort $\mathbb{N}$.

The set of declarations for sort information is traditionally called the *signature* of the sorted logic. Classical sorted logics know three mechanisms for declaring this sort information:

- Variables can be restricted to sorts via declarations of the form $[X{::}\mathbb{A}]$ where $X$ is a variable and $\mathbb{A}$ is a sort. In fact, most sorted logics postulate a total sort function, that associates a unique sort with each variable.
- Constants and functions can be declared to belong to certain sorts by declarations of the form $[c{::}\mathbb{A}]$ and $[f{::}\mathbb{A} \to \mathbb{B}]$.
- Subsort information can be declared by declarations of the form $[\mathbb{A} \leq \mathbb{B}]$. This induces a subsort relation on sorts, which is the smallest partial ordering that contains these subsort declarations. The subsort relation plays such a central role in sorted logics that these are often called "order-sorted". It is a useful notion to employ, since it allows the specification of hierarchies of sorts, which encode the definitional taxonomies of objects that play a great role in mathematics. Examples of such taxonomies are the hierarchies in algebra (semigroups, monoids, rings, fields,...) or numbers (natural numbers, integers, rationals, complex,...).

Naturally, these declarations have an effect on the notion of a model of a logic system. The carrier set $\mathcal{D}$ has subsets $\mathcal{D}_{\mathbb{A}}$ corresponding to the sorts. The correct semantical notion for functions is that of partial functions, which obey the declarations of the signature, i.e. if $[f{::}\mathbb{A} \to \mathbb{B}]$ is declared in the signature, the denotation of $f$ must be a partial function that is total on $\mathcal{D}_{\mathbb{A}}$, and moreover $f(\mathcal{D}_{\mathbb{A}}) \subseteq \mathcal{D}_{\mathbb{B}}$. For each subsort declaration $[\mathbb{A} \leq \mathbb{B}]$ we must have $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}_{\mathbb{B}}$. This semantics reflects the fact that humans use certain classes to structure the universe and that mathematicians naturally use variables and functions restricted to these classes. Thus the sorted models are closer to the intuition of mathematicians than the unsorted ones.

However, it is well-known and, in fact, one of the major results of first-order logic [Sch38, Sch51, Wan52, Obe62], that the use of sorts does not yield logics that have more expressive or deductive power, since with the technique of relativization all sorted first-order formulae, proofs, and models can be coded into unsorted first-order logic, in such a way that entailment and provability are preserved. For instance, the formula $\forall X{::}\mathbb{A}.\mathbf{A}$ is transformed to $\forall X.\mathbb{A}(X) \Rightarrow \mathbf{Rel}(\mathbf{A})$, where $\mathbb{A}$ is a new unary predicate of the unsorted language. Declarations like $[c{::}\mathbb{A}]$ or $[\mathbb{A} \leq \mathbb{B}]$ from the signature are relativized to new "signature axioms" $\mathbb{A}(c)$ and $\forall X.\mathbb{A}(X) \Rightarrow \mathbb{B}(X)$. On the model theoretic side the algebras of partial functions are transformed into algebras of total functions by extending partial functions arbitrarily. So-called sort theorems now verify the coordination of the two notions of relativizations by stating that a sorted sentence $\mathbf{A}$ is satisfiable, iff its relativization $\mathbf{Rel}(\mathbf{A})$ has a model that also satisfies the signature axioms.

In his sorted logic with term declarations [SS89] Schmidt-Schauß relaxes the implicit condition that only the sorts of constants and variables can be declared, and allows declarations of the form $[\mathbf{A}::\mathbb{A}]$, called *term declarations*, where $\mathbf{A}$ can be an arbitrary formula. The idea of term declarations is that there can be sort information within the structure of a formula, if the formula matches a certain schematic formula (a term declaration). Consider, for instance, the addition function, which (semantically) we would like to have the sort $\mathbb{N}\times\mathbb{N} \longrightarrow \mathbb{N}$ where $\mathbb{N}$ is the sorts of natural numbers. If we also have a sort for the even numbers $\mathbb{E}$, then we might want to specify that the expression $[+aa]$ is an even number, even if $a$ is not. This information can be formalized by declaring the formula $[+X_{\mathbb{N}}X_{\mathbb{N}}]$ to be of sort $\mathbb{E}$ using a term declaration. In this expressive system term declarations of the form $[X_{\mathbb{A}}::\mathbb{B}]$ entail that $\mathbb{A}$ is a subsort of $\mathbb{B}$ and induce the intended subsort ordering on the set of sorts.

Research in sorted first-order logics and sorted deduction has primarily centered around the following topics:

*Unification*: Here the impetus has been in developing expressive sorted logics, finding unification algorithms, and proving complexity results for them. For instance, unification in sorted logics is decidable and unitary, if the subsort relation is tree-like, and finitary, in the case where only function declarations are allowed [Wal84, Wal88]. Schmidt-Schauss proved that in the case of general term declarations, sorted unification is undecidable and infinitary. However, there is a restricted class of term declarations (presented by Uribe and Socher in [Uri92, Soc93]) that is much richer than that of function declarations, and where unification is still decidable. For a very abstract account of unification in various sort theories see [CF92].

*Sorted refutation calculi*: The sorted resolution calculi of Walther [Wal83, Wal87], and Schmidt-Schauß [SS86, SS87, SS89] simply substitute sorted unification for unsorted unification to obtain a sorted refutation calculus (the power and generality of this approach has been made explicit by Frisch in [Fri90]). Here the taxonomic theory in the signature is fixed in advance and is completely separated from the object theory. Thus the sort theory can only influence the deduction by restraining the search space. The calculi of Cohn [Coh87, Coh89, Coh92], Beierle et al. [BHP+92], and Weidenbach [Wei89, WO90, Wei91, Wei93] even allow conditional term declarations. This mechanism allows the derivation of more sort information during a proof and thus makes the sort information that constrains the proof search more and more concise during the proof search. The most extreme calculus is the resolution calculus with dynamic sorts [Wei91, Wei93], where the sort theory and the object theory are completely mixed, but the unification procedure used in the resolution inference always takes the current state of the sort theory into account.

*Logic programming*: Since logic programming languages like PROLOG are based on fragments of first-order logics and the operational semantics can be seen as a very restricted form of resolution, there are various order-sorted logic programming languages (such as, for instance TEL [SNGM87, Smo89] or the many-sorted language GÖDEL[HL94]) that take advantage of the sort mechanisms. In this setting the aspect of search control – the user has a clear understanding of the operational semantics (the search behavior) and uses it for programming – is not as important as the greater conciseness of problem formulations. Since sorted programming languages should not be less efficient than unsorted ones on unsorted problems, compilers of such languages often try to precompute sorts of terms at compile-time in order to reduce the complexity of the sorted unification needed at runtime. Practical type systems also provide mechanisms of polymorphism and type reconstruction to ease the burden of typing for the user.

### 1.4 Sorted $\lambda$-Calculi (Related Work)

The question of the behavior of higher-order logic under the constraints of a full sorted type structure is a natural one to ask, in particular, since calculi in this system promise the development of more powerful deduction systems for real mathematics.

In typed $\lambda$-calculi the idea of declaring sort information is very natural, as all objects are already typed, which amounts to a – very coarse – division of the universe into classes. The type system is merely refined by considering the sorts as additional base types. This gives rise not only to new classes of objects (sorts $\mathbb{A}, \mathbb{B}$), but also of functions (sort $\mathbb{A} \to \mathbb{B}$), where domains and codomains are just the sorts. Thus a sorted higher-order logic seems to be the most adequate system, for example, to formalize analysis, since we now have constructs for the domain and codomain, the image and the support of a function, and for function restriction within the system.

Huet was the first to propose the study of a sorted version of higher-order logic in an appendix to [Hue72]. The unification problem in extensions of this system has since been studied by Nipkow and Qian [NQ92] and Pfenning and the author [KP93]. Furthermore, typed $\lambda$-calculi with order-sorted type structures have been of interest in the programming language community as a theoretical basis for object-oriented programming and for more expressive formalisms for higher-order algebraic specifications [Qia91, Car84, BL90, Pie91].

Nipkow and Qian [NQ92] consider a collection of sort systems parameterized by rules for contravariance[6] in the domain sort. This principle states that, if $\mathbb{A} \leq \mathbb{B}$, then $\mathbb{B} \to \mathbb{C} \leq \mathbb{A} \to \mathbb{C}$ and (semantically) corresponds to implicit function restriction. The paper presents a unification algorithm for the resulting sorted calculi. Functional formulae in these calculi in general do not have unique supporting sorts. The consequent difficulties with extensionality are solved by studying unification under sorted equalities that have been restricted to appropriate domain sorts — such restrictions enable specification of a well-defined $\eta$-rule. Constant overloading and functional base sorts are not present in these calculi.

In [KP93] Pfenning and the author consider a calculus $\Lambda^{\to\&}$ with intersection sorts. The intersection operator & on sorts provides sorts $\mathbb{A}\&\mathbb{B}$ denoting the intersection of the sets denoted by $\mathbb{A}$ and $\mathbb{B}$. This calculus also supports contravariance in the domain sort and constant overloading. Permitting intersection sorts makes it possible to define a minimal sort for every formula, so that all signatures are regular. In this setting problems with extensionality are alleviated by allowing only typed abstractions and by defining a formula

[6]The dual covariance principle states that, if $\mathbb{A} \leq \mathbb{B}$, then $(\mathbb{C} \to \mathbb{A}) \leq (\mathbb{C} \to \mathbb{B})$.

$\lambda X.\mathbf{M}$ to have the sort $\mathbb{A} \rightarrow \mathbb{B}$, iff $\mathbf{M}$ has sort $\mathbb{B}$ assuming that $X$ has sort $\mathbb{A}$. $\eta$-equality is then a typed relation which preserves the sorts of formulae. This calculus has been generalized by Pfenning [Pfe92] to a $\lambda$-calculus with dependent types, which will be used as a logical framework extending LF in the Elf programming language [Pfe91].

The calculi mentioned above only allow for sorting the universe of individuals, so they are not directly comparable, in terms of expressive power, with the one presented in this thesis. Indeed, these calculi represent a principally different approach to deduction which appears to call for a semantics where functions are total functions on the types and where the sort information only specifies the behavior of these functions when restricted.

The works of Cardelli [Car88], Bruce and Longo [BL90], Curien and Ghelli [CG91], and Pierce [Pie91] treat variants of the system $F_{\leq}$ which encompass polymorphic intersection types (i.e. intersection types whose variables are explicitly quantified) and the interaction between these types, and various subsort relations. These calculi serve as computational models for functional programming languages and are much more expressive than those studied here, but since they are not intended for deduction purposes, their unification problems have not yet been addressed. In such calculi subsort declarations are not required to respect the functional structure of types, rendering the decidability of sort assignment a very complex issue.

Farmer develops a version of higher-order logic LUTINS [Far91a, Far91b], where all objects of functional type are partial functions and uses it as the working language of the IMPS [FGT93] system, an interactive deduction system that has been used to formalize and prove a large variety of theories in mathematics. In this setting sorts are a derived notion, they are used as a device to characterize certain partial functions as total on the sets represented by the sorts and make computation much more efficient using this information. Experiments with the IMPS system have shown that the greater expressiveness of sorted logics[7] is invaluable for formalizing mathematical statements and finding proofs for them.

## 1.5 $\Sigma\Lambda$: A Sorted $\lambda$-Calculus

The logical system $\Sigma\mathcal{HOL}$ (sorted type theory) developed in this thesis is a sorted higher-order logic that is an instance of a sorted $\lambda$-calculus $\Sigma\Lambda$ with term declarations and functional base sorts. We now proceed to discuss the primary features of $\Sigma\Lambda$ and our results.

Since the terms "type" and "sort" are not used uniformly in the literature, let us now take a look at the underlying principles and fix the usage for this thesis. Both terms refer to the idea of annotating syntactic objects with semantic information about class membership. The notion of "type" has first been introduced by Russell to avoid paradoxes and antinomies in higher-order logics. As we have seen in 1.3 the term "sort" comes from first-order deduction systems, where the mechanism is used for representing part of an axiomatization into sort information that can be efficiently manipulated by the calculus. In typed $\lambda$-calculi the type mechanism is used in both ways without properly distinguishing them. In $\Sigma\Lambda$ we want to make the type mechanism of simply typed $\lambda$-calculus more expressive without losing consistency of the language. Thus we separate both uses into a simple type system (for the safety aspect) and a sort system (for the additional expressiveness). In particular, we will use the term "type" to refer to a mechanism that is used for the safety aspect and

[7] Experience with LUTINS shows that the vast majority of examples can be handled with sort mechanisms, a fact that may also corroborated by our experience in the Ω-MKRP system [HKK+92, HKK+94].

the term "sort" for the representation aspect. Clearly the sort system has to conform to the type system in some way in order to ensure that no antinomies can be imported via the sort declarations. In our case, the sort system is a refinement of the underlying type system, and sorted operations will turn out to be refinements of their unsorted counterparts; in particular, well-sorted formulae are still well-typed in $\Sigma\Lambda$.

The calculus $\Sigma\Lambda$ differs from the systems described in 1.4 in the following three principal ways:

*Functional base sorts*: In addition to partitioning the function universes into the classes $\mathbb{A} \to \mathbb{B}$ of functions defined by domains $\mathbb{A}$ and codomains $\mathbb{B}$, the sort system of $\Sigma\Lambda$ allows base sorts of functional type, i.e. base sorts that denote subclasses of the function classes $\mathbb{A} \to \mathbb{B}$. Syntactically, each sort $\mathbb{A}$ comes with a type $\tau(\mathbb{A})$, and — if is of functional type — also with a domain sort $\mathfrak{d}(\mathbb{A})$ and a codomain sort $\mathfrak{r}(\mathbb{A})$. Semantically, the sorts $\mathbb{C}$ denote subsets $\mathcal{D}_{\mathbb{C}}$ of the family of partial functions of type $\alpha$, where $\alpha$ is the type of $\mathbb{C}$.

*Extensionality*: We do not consider function restriction as a "built in" of the system, since we take seriously the mathematical intuition that functions have uniquely specified domains. In $\Sigma\Lambda$, formulae of functional type have unique supporting sorts, i.e. if a formula $\mathbf{A}$ has sorts $\mathbb{A}$ and $\mathbb{B}$, then $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$. Consequently, our subsort relation cannot be contravariant in the domain sort. This property of $\Sigma\Lambda$ allows us to give a meaningful account of the extensionality principle ($\forall X{::}\mathbb{A}.fX = gX \Rightarrow f = g$), which relies on the concept of unique supporting sorts (unique domains of functions), even in a context without typed equality[8].

*Term Declarations*: The term declaration mechanism is much more powerful than the declaration schemata proposed in the $\lambda$-calculi mentioned in 1.4. $\Sigma\Lambda$ can be seen as a unifying framework which subsumes most known declaration mechanisms. For instance, sort inclusion (a concept that is primitive to most sorted logics) is a derivable mechanism in $\Sigma\Lambda$.

The aspects of an extensional sort system and functional base sorts are genuinely higher-order notions, and do not occur in first-order logic. Furthermore, they interact in a very subtle way. We have investigated a subsystem of $\Sigma\Lambda$ that only allows signatures consisting of constant declarations and thus treats the interaction of functional base sorts and extensionality in isolation in [JK93, JK94]. In contrast to this subsystem, the powerful mechanism of term declarations in $\Sigma\Lambda$ allows a straightforward specification of many mathematical concepts (cf. examples 3.4.10, 4.5.17, and 6.4.8).

The idea of using term declarations as a general framework for sorted logics was used in [SS89], so $\Sigma\Lambda$ is a generalization of the system presented there. Achieving a formalization for the term declaration mechanism in the context of $\lambda$-calculi and higher-order accounts for most of the technical difficulties that we deal with in this thesis. The task is so difficult, since term declarations heavily interact with $\beta$-conversion (see the discussion in 3.2.12).

[8] In various logical systems the problem of identifying supporting sorts is circumvented by requiring a typed notion of equality $=^{\mathbb{A}}$. For instance, if $\mathbb{R}$ ($\mathbb{P}$) is the sort of real (positive real numbers) and $i$ ($a$) is the identity (absolute value) function on the real numbers, then we would have $i =^{\mathbb{P}} a$ but not $i =^{\mathbb{R}} a$. In such systems the extensionality principle has the form $\forall X{::}\mathbb{A}.fX =^{\mathbb{B}} gX \Rightarrow f =^{\mathbb{A}\to\mathbb{B}} g$ independent of the intuitive domain of $f$ and $g$.

We have focused on the term declaration mechanism, since it yields a very general basis[9] for understanding sorted $\lambda$-calculi. Moreover, since the term declaration mechanism is in no way tied to extensionality or functional base sorts, we conjecture that it can be added to most type systems in the literature. If we, for instance, add a declaration mechanism (restricted to variable declarations is sufficient) to the system $\Lambda^{\to\&}$ of [KP93], then the special inference rule for well-sorted abstractions gives rise to the contravariance principle in subsorting. This suggests that $\Sigma\Lambda$ and $\Lambda^{\to\&}$ are in some ways more homogenous systems than that of [NQ92], where this is not the case.

The generalization of Schmidt-Schauß' logic to the higher-order setting has exposed methodical difficulties in the first-order system, which have also flawed an earlier attempt [Koh92] to treat the unification problem for $\Sigma\Lambda$. In this thesis we have corrected the relevant definitions of [Koh92] and with these were able to prove all the results claimed there.

One of the difficulties in devising a formal system with term declarations is that the signature needed for defining well-sortedness contains formulae that again have to be well-sorted. Furthermore, the concept of $\beta\eta$-conversion is so basic to $\lambda$-calculi that it should not change[10] the sort of a formula. Therefore it is necessary to combine the inference systems for validity of signatures, well-sortedness, and sorted $\beta\eta$-reduction into one large inference system. This approach has the advantage that the role of the formulae in term declarations, which was somewhat mystical in [SS89], is now absolutely clear. Furthermore, the property of subterm-closedness (subformulae of well-sorted formulae are well-sorted), which Schmidt-Schauß is forced to assume, becomes a theorem of $\Sigma\Lambda$.

Our main results for $\Sigma\Lambda$ are that sorted $\beta\eta$-reduction is terminating and confluent and moreover conserves the sets of sorts of formulae. Additionally, well-sorted formulae always have unique supporting sorts. The proof of the $\beta$-reduction results makes use of the fact that $\Sigma$-substitutions (well-sorted substitutions) also preserve sets of sorts. These results show that even though there is a strong interaction between sorted $\beta\eta$-conversion and term declarations (proofs of well-sortedness are totally independent of the structure of formulae) the system $\Sigma\Lambda$ still satisfies the basic properties required for speaking of a $\lambda$-calculus.

All of the development of $\Sigma\Lambda$ takes place in the general algebraic framework of $\Sigma$-structures, which subsume the structures of well-sorted formulae and the relevant semantical notions of $\Sigma$-algebras of partial functions. This allows us to give a very structured presentation of the theory, and to use the algebraic notions of $\Sigma$-homomorphisms and $\Sigma$-congruences for understanding the syntactic and semantic manipulations needed in the proofs.

Since our study of $\Sigma\Lambda$ was motivated by the quest for efficient calculi for higher-order theorem proving, we study the unification problem of $\Sigma\Lambda$ and present related algorithms for general $\Sigma$-unification and pre-$\Sigma$-unification. Just as in the unsorted case, these algorithms

[9] Note that this is only meant in the context of Church-style logical systems for formalizing mathematics. Clearly intensional $\lambda$-calculi like Martin-Löf type theory [ML94] or the calculi of constructions [CH85] have much more general type systems, but theses systems are usually only used via the propositions as types isomorphism, and thus these sorts do not appear as a mechanism on the object level, which is just what we are interested in.

[10] This assumption is a rather strong one, motivated by the intended semantics of mathematics. In fact, for most logical systems it would suffice to assume that reduction to long $\beta\eta$-normal form does not render formulae ill-sorted by loosing sorts of subformulae. However, in $\Sigma\Lambda$ we need the stronger assumption in order for $\Sigma$-unification to work properly.

build upon the notion of a general binding, i.e. a formula that is most general in the class of all formulae that share a given head and sort. In $\Sigma\Lambda$ we identify these formulae and prove the general binding theorem (4.2.4), stating that for any given formula $\mathbf{A}$ of head $h$ and sort $\mathbb{A}$ there exists a general binding $\mathbf{G}$ of head $h$ and sort $\mathbb{A}$ and a $\Sigma$-substitution $\rho$ such that $\rho(\mathbf{G})$ is equal to $\mathbf{A}$ up to sorted $\beta\eta$-equality. Once this theorem – which is nearly trivial in the unsorted case – is established, the unification algorithms can be obtained with standard methods and their correctness and completeness only requires standard proofs. The only surprising fact is that in the case of pre-$\Sigma$-unification, we have to require regular signatures and even then cannot fully eliminate the so-called "guess rule", since due to the presence of functional base sorts flex-flex pairs are not always trivially solvable.

The proof of the general binding theorem is based on the structure theorem, which is the main technical result of this thesis, since the unification and the completeness results for the resolution calculus heavily depend on it. The structure theorem 4.1.2 establishes a correspondence between the structure of a formula $\mathbf{A}$ and its sort $\mathbb{A}$ by guaranteeing sorting proofs that $\mathbf{A}$ has sort $\mathbb{A}$ in a certain normalized form (cf. 4.1.1). Due to the strong interactions of term declarations and sorted $\beta\eta$-conversion, we have to take advantage of the powerful method of a logical relations proof to be able to prove it. This proof method was developed by Tait for cut-elimination proofs and later adapted for the related task of showing termination of typed $\beta$-reduction. It consists in a subtle combination of inductions over the structure of types and formulae.

### 1.6 $\Sigma\mathcal{HOL}$: A Sorted Higher-Order Logic

In order to obtain the sorted higher-order logic $\Sigma\mathcal{HOL}$ we specialize the types in $\Sigma\Lambda$ by restricting them to a type $\iota$ for individuals and a type $o$ for truth values. We do not need any other types, since we can model all other type distinctions in our sort system. Closely tied to the type $o$ we use a sort $\mathbb{O}$ to be able to speak about truth values in our sort system. To complete the logical system $\Sigma\mathcal{HOL}$, we add compulsory logical constants and term declarations $[q^{\mathbb{A}}::\mathbb{A} \to \mathbb{A} \to \mathbb{O}]$ for equality, $[\neg::\mathbb{O} \to \mathbb{O}]$ and $[\wedge::\mathbb{O} \to \mathbb{O} \to \mathbb{O}]$ for the connectives, and, finally, $[\Pi^{\mathbb{A}}::(\mathbb{A} \to \mathbb{O}) \to \mathbb{O}]$ for quantification.

Since we want to develop a sorted higher-order resolution calculus, we investigate three notions of a model theory. The standard $\Sigma$-model semantics is the intuitive semantics for mathematics, but it has the disadvantage that it does not admit complete calculi, for principal reasons [Göd31], as we have pointed out above. The general $\Sigma$-model semantics is somewhat less intuitive, but admits complete calculi [Hen50], and indeed we present a generalization of Henkin's original calculus from [Hen50] and prove its completeness with respect to this class of models. Unfortunately, current refutation calculi for automated theorem proving have problems with completeness for the general model semantics, since they fail to prove what we have called the axiom of truth values (cf. 5.3.5). Therefore we develop an even weaker semantics – that of $\Sigma$-model structures – which makes these refutation calculi complete. This notion of completeness is equivalent to the notion of relative completeness found in the literature [And71, Mil83, Pfe87]. We compare all of these notions in the sorted setting of $\Sigma\mathcal{HOL}$.

Furthermore, we prove unifying principles for general $\Sigma$-models and $\Sigma$-model structures. These theorems state that sets that have the property of being consistent in some abstract way ($\nabla_{\Sigma}$-consistency) have a $\Sigma$-model (structure). The proof of the unifying principle for

$\Sigma$-model structures is based on techniques from [And71], but we strengthen the notion of abstract consistency class by an additional saturatedness condition. This makes the valuations in the proof total, which in turn simplifies the proof and the semantical objects used in it. Furthermore, it allows to construct a $\Sigma$-term structure for any $\nabla_\Sigma$-consistent set of sentences, which yields the model-theoretic result. Without saturatedness Andrews can only obtain partial valuations, and from this can only conclude $\mathfrak{T}$-consistency of $\nabla_\Sigma$-consistent sets, where $\mathfrak{T}$ is a special Hilbert-Style calculus for higher-order logic. This result is a direct corollary of our's, since $\Sigma\mathfrak{T}$ is sound with respect to $\Sigma$-model structures. The unifying principle for general $\Sigma$-models needs further methods for the manipulation of equality and its connection to propositional equivalence in order to handle the axiom of truth values. These allow us to construct a $\Sigma$-congruence on the $\Sigma$-term structure that collapses the set $\mathcal{D}_{\mathbb{O}}$ to the set $\{\mathtt{T},\mathtt{F}\}$ of truth values, and thus constructs a general $\Sigma$ model for any $\nabla_\Sigma$-consistent set of sentences, provided that $\nabla_\Sigma$ is a saturated, extensional abstract consistency class.

Since an abstract consistency class $\nabla_\Sigma$ can be expressed in purely syntactic terms our unifying principles can be used in a completeness proof for a refutation calculus $\mathcal{C}$ by showing that $\mathcal{C}$-consistency is an abstract consistency property and thus that $\mathcal{C}$-consistent sets of formulae are satisfiable. The contrapositive of this (unsatisfiable sets of formulae are $\mathcal{C}$-refutable) is just the assertion of the completeness for $\mathcal{C}$. We use this argument to give simple and elegant completeness proofs for the sorted variants of the Hilbert-style calculi from [Hen50, And71]. These then result in compactness theorems for $\Sigma\mathcal{HOL}$ with respect to $\Sigma$-model-structures and general $\Sigma$-models.

## 1.7 $\Sigma\mathcal{HR}$: A Mechanization of $\Sigma\mathcal{HOL}$ by Higher-Order Resolution

By using pre-$\Sigma$-unification instead of unsorted, higher-order pre-unification, the refutation calculi for higher-order logics [Hue72, Mil83, And89] can be generalized to sorted calculi for $\Sigma\mathcal{HOL}$. In this thesis we verify this claim by generalizing Huet's calculus of "constrained resolution" [Hue72] to a sorted higher-order resolution calculus $\Sigma\mathcal{HR}$. While the basic concepts of $\Sigma\mathcal{HR}$ come from [Hue72], we further develop the calculus independently of the generalization to the sorted setting. The methods and results that we summarize in the following are mainly independent of the sort system, since the differences are contained in the process of $\Sigma$-unification, which we already have dealt with above.

Naive Skolemization in the resolution calculi in [And71, Hue72] is not sound, in fact, it is possible to prove an instance of an axiom of choice which is known to be independent [And73, Mil83]. Therefore we do not use it in $\Sigma\mathcal{HR}$, but use the well-known technique of explicitly representing the variable dependencies between universally and existentially quantified variables in a relation (called variable condition) that is maintained during the deduction. Our technique is based on the work of Miller [Mil83, Mil91, Mil92] who has corrected Skolemization for higher-order logics and on that of Bibel [Bib82], who has developed such methods in the context of the first-order connection method.

Huet's calculus works on a generalization of first-order clauses that also incorporates unification constraints. As higher-order unification is undecidable the resolution step only cuts literals of complementary polarity from the clauses and adds the appropriate pair to the unification constraint, which is checked for unifiability at the end of the deduction, once an empty clause has been found. Clearly this strategy is not viable for practical applications, since any two literals can be resolved upon, and it is not the case that successive instantiation constrains the search space. Therefore in $\Sigma\mathcal{HR}$ we allow an interleaving of the search for empty clauses and unification (constraint simplification). In particular, the rule $\Sigma\mathcal{HR}(Solv)$ propagates partial solutions from the constraints to the clause part and thus help detect clashes early. Since the substitution may well change the propositional structure of the clause by instantiating a predicate variable, we have to renormalize the clause on the fly. This interleaving also makes it possible to use a variant of Andrews' primitive substitutions [And89] for instantiating flexible literals, which is conceptually much simpler than Huet's splitting rules.

On the other hand the interleaving proof search and unification makes it necessary to prove a a series of lifting lemmata for $\Sigma\mathcal{HR}$, which are then used in the process of showing that $\Sigma\mathcal{HR}$-consistency is an abstract consistency property. In the light of the unifying principle for $\Sigma\mathcal{HOL}$ this fact entails the refutation completeness of $\Sigma\mathcal{HR}$ with respect to $\Sigma$-model structures. Lifting lemmata are theorems well-known from completeness proofs of first-order refutation calculi that guarantee (lifted) refutations of a formula $\mathbf{A}$ whenever there is one for an instance $\theta(\mathbf{A})$. With their help first-order completeness theorems can be reduced to the question of completeness on ground (variable-free) sets of formulae. Huet does not need an explicit lifting lemma for the completeness proof of his calculus, since no instantiation takes place during the deduction. In contrast to first-order logic, where instantiation does not change the propositional structure of a formula or clause, the existence of flexible literals in higher-order logic can result in a change of structure. If the head variable of a flexible literal is instantiated with a formula containing logical constants, then the resulting formula is no longer a clause and has to be renormalized. This change of structure turns out to be the major difficulty in the proof of the lifting lemma, since for lifting steps from the refutation of $\theta(\mathbf{A})$ it is important to maintain a tight correspondence to the clauses and literals in the refutation of $\mathbf{A}$. Thus in any case where instantiation destroys the correspondence it has to be reestablished by suitable primitive substitution inference rules. These rules explicitly instantiate the heads of flexible literals with general bindings. In order to be able to carry out the intended change of structure, their heads have to be logical constants. Due to the existence of term declarations in $\Sigma\mathcal{HOL}$ this is the only place, where there is a difference between the sorted and the unsorted setting. Since general bindings in $\Sigma\mathcal{HOL}$ can have more than one occurrence of logical constants, the proof of the lifting lemma cannot directly follow the structure of $\theta$ as it would be possible in the unsorted case.

## 1.8 Outline of this Thesis

This thesis is organized in three major parts: the first part (sections 2 and 3) is concerned with the introduction of a sorted $\lambda$-calculus $\Sigma\Lambda$, the second part (section 4) deals with $\Sigma$-unification and the third part (sections 5 and 6) is devoted to the development of a higher-order sorted resolution calculus using $\Sigma$-unification. Although we are ultimately interested in the logical system $\Sigma\mathcal{HOL}$, some methods (like unification) do not depend on the interpretation of the $\lambda$-calculus as a logical system. Therefore we will only specialize the system $\Sigma\Lambda$ to the logical system $\Sigma\mathcal{HOL}$ in the third part.

In order to make the exposition in this thesis self-contained and motivate the techniques, we start out by reviewing the classical approach to higher-order deduction before we pass on to our order-sorted version. Thus we use section 2 to give an introduction to Church's simply typed $\lambda$-calculus ($\Lambda$), which serves as a foundation of the following. Here we fix most basic notations and give an algebraic semantics for the simply typed $\lambda$-calculus. However, in contrast to other expositions we already generalize all notions to a partial function setting, since with this precaution we will be able to use the results directly for the sorted version later on. Finally, we review the notion of (type) inference systems, which will be a basic tool for the following sections.

In section 3 we introduce the sorted $\lambda$-calculus $\Sigma\Lambda$. Since in the presence of term declarations sort information cannot simply be derived from the structure of a $\lambda$-formula, we give inference systems for validity of signatures, well-sortedness of formulae, $\beta\eta$-reduction, and $\Sigma$-substitutions. We discuss basic properties like monotonicity or subterm-closedness of signatures, give basic properties of $\Sigma$-substitutions, and discuss the algebraic notions of $\Sigma$-structures and their relations to structures of well-sorted formulae. Furthermore, we show that sorted $\beta\eta$-reduction is terminating, confluent, and sort-preserving and that well-sorted formulae have unique supporting sorts. Finally, we discuss subsorting in $\Sigma\Lambda$ and see that the natural subsorting inference system (and thus the notion of subsorting) is a derivable concept in $\Sigma\Lambda$.

In section 4 we turn our attention to the more algorithmic properties like sort computation or $\Sigma$-unification of $\Sigma\Lambda$. Building on this notion of general bindings we develop three related transformation systems for general $\Sigma$-unification ($\Sigma\mathcal{UT}$) and pre-$\Sigma$-unification ($\Sigma\mathcal{PT}$) and prove them correct and complete. In fact, we need to consider a slightly more general unification problem than found in the literature, since for our resolution calculus $\Sigma\mathcal{HR}$ the unifiers have to respect certain variable conditions.

We start the third part in section 5 by instantiating $\Lambda$ to a logical system $\Sigma\mathcal{HOL}$, which is essentially a sorted version of the Andrews-Henkin version of simple type theory. We give three distinct notions of algebraic semantics:

- standard $\Sigma$-models are the intuitive semantics for sorted higher-order logic,
- general $\Sigma$-models are a generalization of the Andrews/Henkin general model semantics. This is a generalization of the standard semantics that admits complete calculi and is therefore better suited for modeling deduction systems.
- $\Sigma$-model structures are joint generalizations of $\Sigma$-structures, Andrews' $v$-complexes [And71] and Nadathur's labeled structures [Nad92], which allow for extensionality to fail.

We chose this last semantics as relevant for our work, even though it is the weakest notion of the three, since there has not been a reasonable account for extensionality in higher-order refutation calculi. We discuss sorted abstract consistency classes and prove a unifying principle, which will be used in the completeness proofs later on.

Section 6 is devoted to the exposition of a sorted higher-order resolution calculus $\Sigma\mathcal{HR}$, which is a sorted variant of Huet's "Constrained Resolution" calculus. However, since the naive treatment of Skolemization in Huet's calculus is not sound [And73], we develop a variant of Miller's approach [Mil83, Mil92] where the variable dependencies are explicitly represented in a relation (called variable condition) that is maintained during the deduction. Here we make use of the $\Sigma$-unification algorithms that respect variable conditions that we have developed in section 4. Since in contrast to “Constrained Resolution” our calculus allows mixing $\Sigma$-unification and refutation inference rules, we can show a general lifting theorem, which we then use for the completeness proof.

Finally, in sections 7 we discuss some applications of our work and sketch further work left open by this thesis. These concluding sections also give us the chance to situate the work presented in this thesis from the point of view of applicability and the line of research it may lead to.

# 2 Simply Typed λ-Calculus

In this section we review Church's simply typed λ-calculus $\Lambda$. We will use it as an algebraic foundation for higher-order logic. We discuss the algebraic structure of well-formed formulae, and we give notions of algebraic semantics that are independent of the logical view of higher-order logic. For this we introduce the framework of $\Omega$-structures, which provides a more algebraically flavored setting than, for example, the one given in [And86] or [HS86]. In particular, the notions of $\Omega$-homomorphism and $\Omega$-congruence will be useful later on.

We will introduce all concepts of $\Lambda$ in the more general setting of partial functions, as they can be handled with little overhead, and we can also use them as a basis for the sorted λ-calculus $\Sigma\Lambda$ and the sorted higher-order logic $\Sigma\mathcal{HOL}$. In $\Sigma\Lambda$ and $\Sigma\mathcal{HOL}$ the type system is refined by a sort system, in which the domains of functions coincide with their domain sorts (which are subsets of the types), thus functions are total on their domains, but partial on the types.

## 2.1 Preliminaries

We first lay a foundation by fixing the notation for relations and functions, which are the basic objects in all our semantic notions.

**Definition 2.1.1 (Relations)** Let $A$, $B$, and $C$ be sets, then the **Cartesian product $A \times B$ of $A$ and $B$** is the set of pairs $\{(a,b) \mid a \in A, b \in B\}$. A **binary relation $\Phi$** on $A \times B$ is a subset of the Cartesian product $\Phi \subseteq A \times B$, its **domain $\mathbf{Dom}(\Phi)$** is the set $\{a \in A \mid (a,b) \in \Phi\}$, its **image $\mathbf{Im}(\Phi)$** is the set $\{b \in B \mid (a,b) \in \Phi\}$. Let $\Psi \subseteq B \times C$ be another relation, then the **composition of $\Phi$ and $\Psi$** is defined by $\Psi \circ \Phi := \{(a,c) \mid (a,b) \in \Phi, (b,c) \in \Psi\}$, if $\mathbf{Im}(\Phi) \subseteq \mathbf{Dom}(\Psi)$. The relation $\Phi^{-1} := \{(b,a) \mid (a,b) \in \Phi\} \subseteq B \times A$ is called the **inverse relation for $\Phi$**. We call $\Phi$ **left** (**right**) **unique**, if it does not contain two different pairs having the same first (second) components. It is called **total**, if $\mathbf{Dom}(\Phi) = A$, and **surjective**, if $\mathbf{Im}(\Phi) = B$, in this case the inverse relation is total. Let $a \in \mathbf{Dom}(\Phi)$, then the **application $\Phi(a)$ of $\Phi$ to** $a \in A$ is the set $\{b \in B \mid (a,b) \in \Phi\}$. We sometimes write $\Phi(a,b)$, if $(a,b) \in \Phi$.

**Definition 2.1.2** Let $A$ be a set, then we call a relation $\Phi \subseteq A \times A$

- **reflexive**, iff $(a,a) \in \Phi$ holds for all $a \in A$.
- **symmetric**, iff $(a,b) \in \Phi$ implies $(b,a) \in \Phi$.
- **antisymmetric**, iff $(a,b) \in \Phi$ implies $(b,a) \notin \Phi$.
- **transitive**, iff $(a,b), (b,c) \in \Phi$ implies $(a,c) \in \Phi$.
- **an equivalence relation**, iff $\Phi$ is reflexive, symmetric, and transitive.
- **a quasi-ordering**, iff $\Phi$ is transitive and reflexive.

Let $\preceq$ be a quasi-ordering, then we call the relation $\sim \; := \; \preceq \cap \preceq^{-1}$ the **equivalence induced by $\preceq$**, and $\prec \; := \{(a,b) \mid a \preceq b \text{ but } a \not\sim b\}$ the **strict ordering for $\preceq$**. A quasi-ordering $\preceq$ is called a **partial ordering**, iff $\sim$ is trivial, i.e. $x \sim y$, iff $x = y$. It is called

**terminating** or **well-founded**, iff there are no infinite sequences $a_1, a_2, \ldots$ with $a_i \in A$ and $a_{i+1} \prec a_i$. A transitive relation $\Phi$ is called **confluent**, iff for all $a, b, c \in A$ with $\Phi(a,b)$ and $\Phi(a,c)$, there is $d \in A$ such that $\Phi(b,d)$ and $\Phi(c,d)$.

For an equivalence relation $\sim \subseteq A \times A$ we denote the **equivalence class** of $a \in A$ by $[\![a]\!] := [\![a]\!]_\sim := \{b \in A \mid b \sim a\}$.

**Definition 2.1.3 (Partial Function)** We call a left unique relation a **partial function**. It is called **injective**, iff it is also right unique. We denote the family of all partial functions $\Phi \subseteq A \times B$ by $\mathcal{F}_p(A;B)$ and the family of all total relations by $\mathcal{R}^t(A;B)$. The set $\mathcal{F}(A;B) := \mathcal{R}^t(A;B) \cap \mathcal{F}_p(A;B)$ is called the set of **total functions**.

Let $\Phi$ be a partial function and $a \in \mathbf{Dom}(\Phi)$, then the **application** $\Phi(a)$ **of a partial function** $\Phi$ **to** $a \in A$ is the unique $b \in B$ such that $(a,b) \in \Phi$. In order to make the presentation of partial functions simpler, we introduce a special symbol $\perp$ (for the undefined) and extend the definition of function application by $\Phi(c) := \perp$, if $c \in A$, but $c \notin \mathbf{Dom}(\Phi)$, or if $c = \perp$ itself. This often allows to omitting the reasoning about domains. In particular, we have $f \circ g(a) = f(g(a))$ for $g \in \mathcal{F}_p(A;B)$, $f \in \mathcal{F}_p(B;C)$, and $a \in A$ independently of definedness considerations. Note that the symbol $\perp$ is not an object in any of the given sets, but rather a syntactic trick that eases notation. The so-called **function composition** $\circ$ is associative and therefore the sets $\mathcal{F}_p(A;A)$ and $\mathcal{F}(A;A)$ are monoids with this operation.

If $\Phi \in \mathcal{F}_p(A;B)$ and $\Psi \in \mathcal{F}_p(A;B)$ are partial functions such that $\Psi(a) = b$, but $\Psi(c) = \Phi(c)$ for all $c \neq a$, then we denote $\Psi$ by $\Phi, [b/a]$. For partial functions that can be presented by a finite set of pairs (e.g. substitutions and variable contexts), we often use the notation $\Phi := [b^1/a^1], \ldots, [b^n/a^n]$, if $\Phi = \{(a^1, b^1), \ldots, (a^n, b^n)\}$. Furthermore, we denote with $\Phi_{-c}$ the partial function $\{(a,b) \in \Phi \mid a \neq c\}$.

Let $W \subseteq A$ and $\Phi \in \mathcal{F}_p(A;B)$, then the **restriction of** $\Phi$ **to** $W$ is defined to be the function $\Phi|_W := \{(a,b) \in \Phi \mid a \in W\}$. Note that $\Phi|_W \in \mathcal{F}_p(W;B)$ and that with this definition $\Phi$ and $\Phi|_W$ are only equal, iff $W \cap \mathbf{Dom}(\Phi) = \mathbf{Dom}(\Phi)$. If $\Phi$ and $\Psi$ are partial functions such that their restrictions on $\mathbf{Dom}(\Phi) \cap \mathbf{Dom}(\Psi)$ are identical, then we say that they **agree** and write $\Phi || \Psi$. In this case the set-theoretic union $\Phi \cup \Psi$ is again a partial function.

**Remark 2.1.4 ($n$-ary Relations and Functions)** Let $A_1, \ldots, A_n$ be sets, then we can define the $n$**-fold Cartesian product** $A_1 \times \cdots \times A_n$ by $(\cdots(A_1 \times A_2) \times \cdots \times A_n)$, thus it is the set of ordered $n$-tuples $\{(a_1, \ldots, a_n) \mid a_i \in A_i\}$, where $(a_1, \ldots, a_n) := (\cdots(a_1, a_2), \ldots, a_n)$. With this definition we can generalize the previous definitions for binary relations and unary functions to $n$**-ary relations** and $n$**-ary functions**. In particular, the **domain** $\mathbf{Dom}(\Phi)$ of an $n$-ary relation $\Phi$ is the set $\{(a_1, \ldots, a_{n-1}) \mid (a_1, \ldots, a_{n-1}, a_n) \in \Phi\}$, the **image** $\mathbf{Im}(\Phi)$ of $\Phi$ is the set $\{a_n \mid (a_1, \ldots, a_{n-1}, a_n) \in \Phi\}$. We denote the family of $n$-ary total relations by $\mathcal{R}^t(A_1, \ldots, A_n; B)$ and adapt the other notions accordingly.

This construction implies that we can use unary functions instead of general $n$-ary ones. In particular, the well-known process of applying an $n$-ary function $\Phi$ to an $n$-tuple $(a_1, \ldots, a_n)$ can be considered as applying $\Phi$ to the sequence of values $a_1, \ldots, a_n$ one after the other. Thus the application of $\Phi$ to a tuple $(a_1, \ldots, a_k)$ yields an $(n-k)$-ary function that give $a_{n+1}$, if applied to the tuple $(a_{k+1}, \ldots, a_n)$. This process is called **currying**. Therefore $\mathcal{F}(A_1, \ldots, A_n; B)$ becomes $\mathcal{F}(A_1; \mathcal{F}(A_2; \ldots; \mathcal{F}(A_n; B) \ldots)$, and therefore we can restrict ourselves to unary functions.

**Remark 2.1.5 (Extensionality)** Two partial functions $f, g: A \longrightarrow B$ are equal, iff they are equal as binary relations, that is, if $\mathbf{Dom}(f) = \mathbf{Dom}(g)$ and for all $x \in \mathbf{Dom}(f)$ we have $f(x) = g(x)$. This property is called the **extensionality** of equality.

**Definition 2.1.6 (Types)** Let $\mathcal{BT}$ be a set of symbols, then the set $\mathcal{T}$ of **types** is inductively defined to be the set $\mathcal{BT}$ together with all expressions $\alpha \to \beta$, where $\alpha$ and $\beta$ are types. The functional type $\alpha \to \beta$ denotes the type of functions with domain $\alpha$ and codomain $\beta$. The types in $\mathcal{BT} \subseteq \mathcal{T}$ are called **base types**, types of the form $\alpha \to \beta$ are called **functional types**.

We define the **length** of a type $\alpha$ by setting $\mathbf{ln}(\alpha) := 0$, iff $\alpha \in \mathcal{BT}$ and $\mathbf{ln}(\alpha \to \beta) := 1 + \mathbf{ln}(\beta)$. Thus the length intuitively is the number of top level arrows $\to$ in a type. In other words a type $\alpha$ is functional, iff its length is positive.

**Notation 2.1.7** For the following we fix a set $\mathcal{BT}$ of base types and a set $\mathcal{T}$ of types induced by $\mathcal{BT}$. As syntactic variables for types we use lower case Greek letters. We use the convention of association to the right for omitting parentheses in types, thus $\alpha \to \beta \to \gamma$ is an abbreviation for $(\alpha \to (\beta \to \gamma))$. This way the type $\gamma := \beta_1 \to \ldots \to \beta_n \to \alpha$ denotes the type of $n$-ary functions, that take $n$ arguments of the types $\beta_1, \ldots, \beta_n$ and have values of type $\alpha$. To conserve even more space we use a kind of vector notation and abbreviate $\gamma$ by $\overline{\beta_n} \to \alpha$.

We now start with the definition of our basic algebraic structures, which are hierarchies of sets indexed by types. As most objects in $\Lambda$ are such collections, and for well-formedness the type structure has to be respected, we now define the notion of a typed collection, which formalizes this concept.

**Definition 2.1.8 (Typed Collection)** 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)**. In the following we will always assume that $\mathcal{D}_\alpha \cap \mathcal{D}_\beta = \emptyset$, if $\alpha \neq \beta$. This allows us to define the **type function** $\tau: \bigcup_{\alpha \in \mathcal{T}} \mathcal{D}_\alpha \longrightarrow \mathcal{T}$ by $\tau(g) = \alpha$, iff $g \in \mathcal{D}_\alpha$. Let $\mathcal{D}_{\mathcal{T}}$ and $\mathcal{E}_{\mathcal{T}}$ be typed collections, then a collection $\mathcal{I} := \{\mathcal{I}^\alpha \in \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{E}_\alpha) \mid \alpha \in \mathcal{T}\}$ of partial functions is called a **typed partial function** $\mathcal{I}: \mathcal{D}_{\mathcal{T}} \longrightarrow \mathcal{E}_{\mathcal{T}}$.

It is often convenient to view a typed collection $\mathcal{D}_{\mathcal{T}}$ as the union $\bigcup_{\alpha \in \mathcal{T}} \mathcal{D}_\alpha$ and a typed function $\mathcal{I}: \mathcal{D}_{\mathcal{T}} \longrightarrow \mathcal{E}_{\mathcal{T}}$ as a function $\mathcal{I}: \bigcup_{\alpha \in \mathcal{T}} \mathcal{D}_\alpha \longrightarrow \bigcup_{\alpha \in \mathcal{T}} \mathcal{E}_\alpha$ with $\mathcal{I}^\alpha := \mathcal{I}|_{\mathcal{D}_\alpha}$ and $\mathbf{Im}(\mathcal{I}^\alpha) \subseteq \mathcal{E}_\alpha$. We take the liberty to switch the point of view whenever it is convenient. A collection $\{R_\alpha \subseteq \mathcal{D}_\alpha \times \mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ are called a **typed binary relation**.

Our treatment of $\Lambda$ is parametric in the choice of **constants** that are supplied. As constants are typed objects and the considerations of this section depend on their choice, we fix a typed collection $\Omega_{\mathcal{T}}$ of sets of constants. In particular, the algebraic structure of a pre-$\Omega$-structure, which we are about to define, varies in the way the constants are interpreted. The intuitive meaning for objects of functional type is a function, i.e. an object that can be applied to other objects of the appropriate type. Here we give a very abstract notion of an algebraic structure with function applications, which provides us with the basic vocabulary for the development of $\Lambda$.

**Definition 2.1.9 (Pre-Ω-Structure)** Let $\mathcal{D}_{\mathcal{T}}$ be a typed collection of sets,

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

a typed family of partial functions, and let $\mathcal{I}: \Omega \longrightarrow \mathcal{D}$ be a typed total function, then we call the triple $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ a **partial pre-Ω-structure**. The collection $\mathcal{D}$ is called the **carrier set** or the **frame of** $\mathcal{A}$, the set $\mathcal{D}_\alpha$ the **universe of type** $\alpha$, the function @ the **application operator**, and the function $\mathcal{I}$ the **interpretation of constants**. A pre-Ω-structure is called **total**, iff @ is a collection of total functions. For an object $f \in \mathcal{D}_{\alpha\to\beta}$ we define the **domain of** $f$ as the set $\mathbf{Dom}(f) := \{a \in \mathcal{D}_\alpha \mid (f, a) \in \mathbf{Dom}(@)\}$.

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

**Remark 2.1.10** The application operator @ in a pre-Ω-structure is an abstract version of function application. As in the case with functions before (cf. 2.1.4) 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

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

**Definition 2.1.11 (Ω-Homomorphism)** Let $\mathcal{A} = (\mathcal{D}, @^{\mathcal{A}}, \mathcal{I})$ and $\mathcal{B} = (\mathcal{E}, @^{\mathcal{B}}, \mathcal{J})$ be pre-Ω-structures. A **Ω-homomorphism** is a typed function $\kappa: \mathcal{D} \longrightarrow \mathcal{E}$ such that

1. $\kappa \circ \mathcal{I} = \mathcal{J}$.
2. For all $f \in \mathcal{D}_{\alpha\to\beta}$ and $g \in \mathcal{D}_\alpha$ we have: if $g \in \mathbf{Dom}(f)$, then $\kappa(g) \in \mathbf{Dom}(\kappa(f))$ and $\kappa(f)@^{\mathcal{B}}\kappa(g) = \kappa(f@^{\mathcal{A}}g)$.

As usual we define an **Ω-endomorphism** $\kappa$ **on** $\mathcal{A}$ to be an Ω-homomorphism $\kappa: \mathcal{A} \longrightarrow \mathcal{A}$, an **Ω-epimorphism** and an **Ω-monomorphism** to be surjective and injective Ω-homomorphisms respectively.

## 2.2 Well-Formed Formulae

A prominent example of a pre-Ω-structure is the collection of well-formed formulae. For defining them we need a collection of variables as a category of syntactic objects distinct from the collection Ω of constants. Since variables are much more volatile syntactic objects, which are frequently instantiated and renamed, we need an infinite supply of variables of any type. So we fix a countably infinite set $\mathcal{V}_\alpha$ of **variables of type** $\alpha$ for every type $\alpha \in \mathcal{T}$. Thus we have a typed collection $\mathcal{V}$ of variables, which we use in the following.

**Definition 2.2.1 (Well-Formed Formulae)** For each $\alpha \in \mathcal{T}$ we define the set $\mathit{wff}_\alpha(\Omega)$ of **well-formed formulae of type** $\alpha$ inductively by

1. $\Omega_\alpha \cup \mathcal{V}_\alpha \subseteq \mathit{wff}_\alpha(\Omega)$
2. If $\mathbf{A}_{\beta\to\alpha} \in \mathit{wff}_{\beta\to\alpha}(\Omega)$ and $\mathbf{B}_\beta \in \mathit{wff}_\beta(\Omega)$, then $\mathbf{AB} \in \mathit{wff}_\alpha(\Omega)$.
3. If $\mathbf{A}_\alpha \in \mathit{wff}_\alpha(\Omega)$, then $(\lambda X_\beta.\mathbf{A}_\alpha) \in \mathit{wff}_{\beta\to\alpha}(\Omega)$.

We call formulae of the form $\mathbf{AB}$ **applications**, and formulae of the form $\lambda X_\alpha.\mathbf{A}_\beta$ **$\lambda$-abstractions**.

**Notation 2.2.2** We denote the constants by lower case letters and the variables by upper case letters and use bold upper case letters $\mathbf{A}_\alpha$, $\mathbf{B}_{\alpha\to\beta}$, $\mathbf{C}_\gamma \ldots$ as syntactical variables for well-formed formulae. The type of an object is denoted as a subscript, if it is not irrelevant or clear from the context.

In order to make the notation of well-formed formulae more legible, we use the convention that the group brackets ( and ) associate to the left and that the square dot . denotes a left bracket, whose mate is as far right as consistent with the brackets already present. Additionally, we combine successive $\lambda$-abstractions, so that the well-formed formula $(\lambda X^1.\lambda X^2.\ldots\lambda X^n.\mathbf{AE}^1\ldots\mathbf{E}^m)$, which stands for $(\lambda X^1(\lambda X^2\ldots(\lambda X^n(\mathbf{AE}^1)\mathbf{E}^2\ldots\mathbf{E}^m)\cdots)$, becomes $\lambda X^1\ldots X^n.\mathbf{AE}^1\ldots\mathbf{E}^m$, and in addition, we shorten the expression to $\lambda\overline{X^n}.\mathbf{A}\overline{\mathbf{E}^m}$ by a kind of vector notation.

To avoid confusion with equality in the logic we denote the meta-logical relation of syntactic equality of well-formed formulae by $\doteq$.

**Example 2.2.3** If we define $\mathbf{A}@\mathbf{B} := (\mathbf{AB})$ for $\mathbf{A} \in \mathit{wff}_\alpha(\Omega)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Omega)$, then $@: \mathit{wff}_{\alpha\to\beta}(\Omega) \times \mathit{wff}_\alpha(\Omega) \longrightarrow \mathit{wff}_\beta(\Omega)$ is a total function. Thus $(\mathit{wff}(\Omega), @, \mathrm{Id}_\Omega)$ is a total pre-$\Omega$-structure. The intuition behind this example is that we can think of the formula $\mathbf{A} \in \mathit{wff}_{\alpha\to\beta}(\Omega)$ as a function

$$\mathbf{A}: \mathit{wff}_\alpha(\Omega) \longrightarrow \mathit{wff}_\beta(\Omega)\ ;\ \mathbf{B} \mapsto (\mathbf{AB})\ .$$

**Definition 2.2.4** Let $\mathbf{A}$ be a well-formed formula, then a variable $X$ is called **bound** (**free**) **in $\mathbf{A}$**, iff it is (not) in a well-formed part of the form $(\lambda X.\mathbf{B})$ in $\mathbf{A}$. The respective sets of variables are denoted by $\mathbf{Free}(\mathbf{A})$ and $\mathbf{Bound}(\mathbf{A})$. A well-formed formula is called **closed**, if it does not contain free variables. We denote the set of closed well-formed formulae of type $\alpha$ by $\mathit{cwff}_\alpha(\Omega)$.

With the definition of free variables we can define sets of well-formed formulae that have restricted sets of free variables: let $\Xi \subseteq \mathcal{V}$ be a typed collection of variables, then we denote the set of well-formed formulae with free variables in $\Xi$ by $\mathit{wff}_\alpha(\Omega,\Xi) := \{\mathbf{A} \in \mathit{wff}_\alpha(\Omega) \mid \mathbf{Free}(\mathbf{A}) \subseteq \Xi\}$. Since any formula $\mathbf{A}$ can only have finitely many variables there is always a set $\Xi$ of variables such that $\mathbf{A} \in \mathit{wff}_\alpha(\Omega,\Xi)$.

**Definition 2.2.5 (Assignment)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-$\Omega$-structure. A typed function $\varphi: \mathcal{V} \longrightarrow \mathcal{D}$ is called an **assignment into $\mathcal{A}$**.

In a pre-$\Omega$-structure $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ constants are given a meaning by the interpretation function $\mathcal{I}: \Omega \to \mathcal{D}$, and variables get their meaning by assignments $\varphi: \mathcal{V} \to \mathcal{D}$. Since well-formed formulae are inductively built up from constants and variables we can extend $\varphi$ and $\mathcal{I}$ to an $\Omega$-homomorphism on well-formed formulae.

**Definition 2.2.6 (Homomorphic Extension)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a functional pre-$\Omega$-structure and let $\varphi$ be an assignment into $\mathcal{A}$. Then the **homomorphic extension $\mathcal{I}_\varphi$ of $\varphi$ to $\mathit{wff}(\Omega)$** is inductively defined to be a typed partial function $\mathcal{I}_\varphi: \mathit{wff}(\Omega) \longrightarrow \mathcal{D}$ such that

1. $\mathcal{I}_\varphi(X) = \varphi(X)$, if $X$ is a variable,
2. $\mathcal{I}_\varphi(c) = \mathcal{I}(c)$, if $c$ is a constant,
3. $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B})$,
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 call $\mathcal{I}_\varphi(\mathbf{A}_\alpha) \in \mathcal{D}_\alpha$ the **value** or **denotation of $\mathbf{A}_\alpha$ in $\mathcal{A}$ for $\varphi$**. Note that since $\mathcal{A}$ need not be total, we can have $\mathcal{I}_\varphi = \bot$.

**Lemma 2.2.7** *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a functional pre-$\Omega$-structure and $\varphi: \mathcal{V} \longrightarrow \mathcal{D}$ an assignment into $\mathcal{A}$, then the homomorphic extension $\mathcal{I}_\varphi: \mathit{wff}(\Omega) \longrightarrow \mathcal{D}$ is an $\Omega$-homomorphism.*

**Proof:** The assertion is a direct consequence of the definitions and the fact that $\mathcal{I}_\varphi \circ \mathrm{Id}_\Omega = \mathcal{I} \circ \mathrm{Id}_\Omega = \mathcal{I}$ on $\Omega$. □

**Remark 2.2.8** We have defined valuation only on functional pre-$\Omega$-structures, which is sufficient for our purposes, since we want to formalize mathematical systems. In fact, it seems to be rather difficult to give a general definition of values for abstractions without functionality. Andrews and Nadathur solve this problem for $v$-complexes [And72] and labeled structures [Nad92] by assuming a tight correspondences between objects and labels, but we do not know how to generalize this to the framework of $\Omega$-structures.

**Definition 2.2.9 ($\Omega$-Structure)** A functional pre-$\Omega$-structure $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ is called **comprehension-closed**, iff for each assignment $\varphi$ into $\mathcal{A}$ the homomorphic extension $\mathcal{I}_\varphi$ is total on $\mathit{wff}(\Omega)$. A functional pre-$\Omega$-structure is called **$\Omega$-structure**, iff it is comprehension-closed. These closure conditions for the carrier set $\mathcal{D}$ of $\mathcal{A}$ assure that the universes of functions $\mathcal{D}_{\alpha\to\beta}$ are rich enough to contain a value for all $\mathbf{A}_{\alpha\to\beta} \in \mathit{wff}_{\alpha\to\beta}(\Omega)$. For a detailed discussion in the framework of $\Omega$-algebras we refer the reader to [And72, And73].

**Remark 2.2.10** Note that the pre-$\Omega$-structure $\mathit{wff}(\Omega)$ from 2.2.3 is not comprehension closed, since there is no formula $\mathbf{C} = \mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{B}) \in \mathit{wff}_{\alpha\to\beta}(\Omega)$ such that $\mathbf{C}@\mathbf{A} = \mathbf{CA} = \mathcal{I}_{\varphi,[\mathbf{A}/X]}(\mathbf{B})$. 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 $\mathit{wff}(\Omega)$ were comprehension closed $\beta\eta$-equality would have to be valid in $\mathit{wff}(\Omega)$ (cf. 2.3.22), which it clearly is not.

**Lemma 2.2.11** *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be an $\Omega$-structure, $\mathbf{A} \in \mathit{wff}(\Omega)$ and let $\varphi$ and $\psi$ be assignments into $\mathcal{A}$ that coincide on* $\mathbf{Free}(\mathbf{A})$*, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$.*

**Proof:** We prove the lemma 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 $\mathbf{A} \doteq (\lambda X.\mathbf{B})$, then $\mathcal{I}_\varphi(\mathbf{A})@a = \mathcal{I}_{\varphi,[a/X]}(\mathbf{B}) = \mathcal{I}_{\psi,[a/X]}(\mathbf{B}) = \mathcal{I}_\psi(\mathbf{A})@a$ by inductive hypothesis, since $\varphi, [a/X]$ and $\psi, [a/X]$ coincide on the free variables of $\mathbf{B}$. Thus we obtain the assertion from the definition of $\mathcal{I}_\varphi$. □

### 2.3 $\lambda$-Reduction and Normal Forms

In this section we introduce the notions of $\Omega$-congruences, $\lambda$-conversion, and substitutions, which are closely related to each other. The $\lambda$-conversion relations establish certain well-formed formulae as functions, by giving interpretations to function application and function equality.

**Definition 2.3.1 ($\Omega$-Congruence)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-$\Omega$-structure, then a typed equivalence relation $\sim$ is called an **$\Omega$-congruence on** $\mathcal{A}$, iff $f \sim f' \in \mathcal{D}_{\alpha\to\beta}$ and $g \sim g' \in \mathcal{D}_\alpha$ imply $f@g \sim f'@g'$.

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

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

**Definition 2.3.2 (Quotient Pre-$\Omega$-Structure)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-$\Omega$-structure, $\mathcal{D}^\sim_\alpha := \{[\![f]\!] \mid f \in \mathcal{D}_\alpha\}$, and $\mathcal{I}^\sim(c_\alpha) := [\![\mathcal{I}(c_\alpha)]\!]$ for all constants $c_\alpha \in \Omega_\alpha$. Furthermore let $@^\sim$ be defined by $[\![f]\!] @^\sim [\![a]\!] := [\![f@a]\!]$. To see that this definition only depends only on equivalence classes of $\sim$, consider $f' \in [\![f]\!]$ and $g' \in [\![g]\!]$, then $[\![f@g]\!] = [\![f'@g]\!] = [\![f'@g']\!] = [\![f@g']\!]$. So $@^\sim$ is well-defined and thus $\mathcal{A}/_\sim := (\mathcal{D}^\sim, @^\sim, \mathcal{I}^\sim)$ is also a pre-$\Omega$-structure. We call $\mathcal{A}/_\sim$ the **quotient structure of $\mathcal{A}$ for the relation** $\sim$ and the typed function $\pi_\sim : \mathcal{A} \longrightarrow \mathcal{A}/_\sim; f \mapsto [\![f]\!]_\sim$ its **canonical projection**.

This definition is justified by the following theorem.

**Theorem 2.3.3** *Let $\mathcal{A}$ be a pre-$\Omega$-structure and let $\sim$ be an $\Omega$-congruence on $\mathcal{A}$, then*

1. *the canonical projection $\pi_\sim$ is an $\Omega$-epimorphism.*
2. *$\mathcal{A}/_\sim$ is functional, iff $\sim$ is functional.*
3. *$\mathcal{A}/_\sim$ is comprehension-closed, iff $\mathcal{A}$ is.*
4. *$\mathcal{A}/_\sim$ is total, if $\mathcal{A}$ is.*

**Proof:** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-$\Omega$-structure.

1. To convince ourselves that $\pi_\sim$ is indeed an $\Omega$-epimorphism, we note that by definition $\pi_\sim$ is surjective and $\mathcal{I}^\sim = \pi_\sim \circ \mathcal{I}$. Now let $f \in \mathcal{D}_{\beta\to\alpha}$, and $g \in \mathbf{Dom}(f) \subseteq \mathcal{D}_\beta$, then $g' \in [\![g]\!]$ for all $g' \in \mathbf{Dom}(f)$ and therefore $[\![g]\!] = \pi_\sim(g) \in \mathbf{Dom}([\![f]\!]) = \mathbf{Dom}(\pi_\sim(f))$ and $\pi(f)@^\sim\pi(g) = [\![f]\!] @^\sim [\![g]\!] = [\![f@g]\!] = \pi(f@g)$.
2. Note that $[\![f]\!] = [\![g]\!]$, iff $f \sim g$, iff $f@a \sim g@a$, iff $[\![f@a]\!] = [\![g@a]\!]$, iff $[\![f]\!] @^\sim [\![a]\!] = [\![g]\!] @^\sim [\![a]\!]$ for all $a \in \mathcal{D}_\alpha$ and thus for all $[\![a]\!] \in \mathcal{D}^\sim_\alpha$.
3. Let $\psi$ be an assignment into $\mathcal{A}/_\sim$, then there exists an assignment $\varphi$ into $\mathcal{A}$ such that $\psi = \pi_\sim \circ \varphi$, since $\pi_\sim$ is an $\Omega$-epimorphism. We prove that $\mathcal{I}^\sim_\psi = \pi_\sim \circ \mathcal{I}_\varphi$, (which entails the assertion) by induction over the structure of well-formed formulae. In order to simplify the notation we abbreviate $\pi_\sim$ by $\pi$.

(a) $\mathcal{I}^{\sim}_{\psi}(X) = \psi(X) = \pi \circ \varphi(X) = \pi(\mathcal{I}_{\varphi}(X))$

(b) $\mathcal{I}^{\sim}_{\psi}(c) = \mathcal{I}^{\sim}(c) = \pi \circ \mathcal{I}(c) = \pi(\mathcal{I}_{\varphi}(c))$

(c) $\mathcal{I}^{\sim}_{\psi}(\mathbf{AB}) = \mathcal{I}^{\sim}_{\psi}(\mathbf{A})@\mathcal{I}^{\sim}_{\psi}(\mathbf{B}) = \pi \circ \mathcal{I}_{\varphi}(\mathbf{A})@\pi \circ \mathcal{I}_{\varphi}(\mathbf{B}) = \pi(\mathcal{I}_{\varphi}(\mathbf{AB}))$

(d) $\mathcal{I}^{\sim}_{\psi}((\lambda X \mathbf{A}))@\pi(g) = \mathcal{I}^{\sim}_{(\psi,[\pi(g)/X]}(\mathbf{A}) = \pi(\mathcal{I}_{(\varphi,[g/X]}(\mathbf{A}) = \pi \circ \mathcal{I}_{\varphi}((\lambda X \mathbf{A}))@\pi(g)$

4. $\mathbf{A}/_{\sim}$ is total, since $\pi_{\sim}$ is an epimorphism. □

**Definition 2.3.4 (Substitution)** We call an assignment $\sigma$ into *wff*$(\Omega)$ a **substitution**, iff its **support** $\mathbf{supp}(\sigma) := \{X \in \mathcal{V} \mid \sigma(X) \not\equiv X\}$ is finite. We write a substitution $\sigma$ as $\mathbf{supp}(\sigma) = \{X^1, \ldots, X^n\}$ and $\sigma(X^i) = \mathbf{A}^i$ as $[\mathbf{A}^1/X^1], \ldots, [\mathbf{A}^n/X^n]$ or short $\overline{[\mathbf{A}^n/X^n]}$. If $\sigma = [\mathbf{A}/X]$, then we often write $\sigma(\mathbf{B})$ as $[\mathbf{A}/X]\mathbf{B}$. The set $\mathbf{Intro}(\sigma) := \bigcup_{i \leq n} \mathbf{Free}(\mathbf{A}^i)$ is called the **set of variables introduced by** $\sigma$. The set of substitutions is denoted by $\mathbf{SUB}(\Omega)$. An injective substitution $\sigma := \overline{[\mathbf{A}^n/X^n]}$ is called **renaming substitution**, if the $\mathbf{A}^i$ are all variables.

**Remark 2.3.5** A substitution $\sigma$ can always be extended to a total $\Omega$-homomorphism $\widehat{\sigma}$ by requiring

1. $\widehat{\sigma}|_{\Omega} := \mathrm{Id}_{\Omega}$
2. $\widehat{\sigma}(\mathbf{AB}) := (\widehat{\sigma}(\mathbf{A})\widehat{\sigma}(\mathbf{B}))$
3. $\widehat{\sigma}(\lambda X.\mathbf{A}) := (\lambda X.\widehat{\sigma}_{-X}(\mathbf{A}))$

This gives us a second mechanism for extending an assignment $\sigma$ to an $\Omega$-homomorphism. Note that in general with this definition it is not the case, that $\sigma(\mathbf{A}) = \mathcal{I}_{\varphi}(\mathbf{A})$. In 2.3.14 we can see a case, where they are equal. It depends on the context, whether it is more convenient to view substitutions as functions with finite support or as $\Omega$-homomorphisms, hence we take the liberty to switch our point of view whenever convenient.

It is easy to see, that if there is a any well-formed part $(\lambda Y.\mathbf{C})$ of $\mathbf{B}$, in which a variable $X$ is free, $Y \in \mathbf{Free}(\mathbf{A})$, and $\mathbf{B}'$ is obtained from $\mathbf{B}$ by replacing all free occurrences of $X$ with $\mathbf{A}$, then $\mathbf{B}'$ has bound occurrences of variables $Y$ that were free occurrences in $\mathbf{A}$. We call this situation **variable capture** and need to avoid it for correctness of instantiation.

**Definition 2.3.6 ($\alpha$-Conversion)** If a well-formed formula $\mathbf{B}$ is obtained from a well-formed formula $\mathbf{A}$ by replacing a subformula $(\lambda X_{\alpha}.\mathbf{C})$ of $\mathbf{A}$ such that $Y_{\alpha} \notin \mathbf{Free}(\mathbf{C})$ with $(\lambda Y_{\alpha}.[Y_{\alpha}/X_{\alpha}]\mathbf{C})$, then $\mathbf{B}$ is called an **alphabetical variant** of $\mathbf{A}$.

**General Assumption 2.3.7** It will turn out in 2.3.19, that we have $\mathcal{I}_{\varphi}(\mathbf{A}) = \mathcal{I}_{\varphi}(\mathbf{B})$ for any $\Omega$-structure $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$, any assignment $\varphi$ into $\mathcal{A}$, and any pair of well-formed formulae $\mathbf{A}$ and $\mathbf{B}$ that are alphabetical variants. Thus we can avoid variable capture, if we rename the bound variables in $\mathbf{Im}(\sigma)$ by $\alpha$-conversion, so that the sets of bound and free variables are disjoint. Another, more algebraic way of avoiding variable capture is to assume $\alpha$-equality to be built into the system and regard well-formed formulae as syntactically equal, iff they are alphabetical variants ($\alpha$-equal). Formally we replace the pre-$\Omega$-structure *wff*$(\Omega)$ by its quotient modulo $\alpha$-conversion. We could also have used de Bruijn's indices [dB72], as a concrete implementation of this approach at the syntax level.

**Definition 2.3.8 (Idempotent Substitution)** A substitution $\sigma$ is called **idempotent**, iff $\sigma \circ \sigma = \sigma$. Note that the condition $\mathbf{Intro}(\sigma) \cap \mathbf{supp}(\sigma) = \emptyset$ is a sufficient condition for $\sigma$ to be idempotent [Sny91].

**Definition 2.3.9 ($\lambda$-Reduction)** Let $\lambda \in \{\beta, \beta\eta, \eta\}$. We say that a well-formed formula $\mathbf{B}$ is obtained from a well-formed formula $\mathbf{A}$ by a **one-step $\lambda$-reduction** ($\mathbf{A} \rightarrow_\lambda \mathbf{B}$), if it is obtained by applying one of the following rules to a well-formed part (which we call a **$\lambda$-redex**) of $\mathbf{A}$.

**$\beta$-Reduction** $(\lambda X.\mathbf{C})\mathbf{D} \rightarrow_\beta [\mathbf{D}/X]\mathbf{C}$.

**$\eta$-Reduction** If $X$ is not free in $\mathbf{C}$, then $(\lambda X.\mathbf{C}X) \rightarrow_\eta \mathbf{C}$.

As usual we denote the transitive closure of a reduction relation $\rightarrow -\lambda$ with $\rightarrow^*_\lambda$. Thus $\mathbf{A} \rightarrow^* \lambda\mathbf{B}$, iff there is a sequence of one-step $\lambda$-reductions

$$\mathbf{A} \doteq \mathbf{A}^1 \rightarrow_\lambda \cdots \rightarrow_\lambda \mathbf{A}^n \doteq \mathbf{B}$$

These rules induce equivalence relations $=_\beta, =_\eta$, and $=_{\beta\eta}$ on *wff*$(\Omega)$, which we call the **$\lambda$-equality** relations. A formula that does not contain a $\lambda$-redex, and thus cannot be reduced by $\lambda$-reduction, is called a **$\lambda$-normal form**.

The $\beta$-, $\eta$-, and $\beta\eta$-reduction relations are terminating and confluent, as the reader can convince himself by looking at the proofs in [Bar80] or [HS86]. For any formula $\mathbf{A}$ there is a sequence of $\beta$-reductions $\mathbf{A} \rightarrow^*_\beta \mathbf{B}$ such that $\mathbf{B}$ is a **$\beta$-normal form**. Furthermore, for any derivation $\mathbf{A}_\alpha \rightarrow^*_{\beta\eta} \mathbf{B}_\alpha$ there is a derivation $\mathbf{A}_\alpha \rightarrow^*_\beta \mathbf{C}_\alpha \rightarrow^*_\eta \mathbf{B}_\alpha$. Thus we can compute $\beta\eta$-normal forms by first reducing to $\beta$-normal form and then further $\eta$-reducing to normal form.

**Lemma 2.3.10** *The $\lambda$-equivalence relations are $\Omega$-congruences on wff$(\Omega)$. Moreover the $\eta$- and $\beta\eta$-equivalence relations are functional.*

**Proof:** The fact that the $\lambda$-equivalences are $\Omega$-congruences is an immediate consequence of the definitions. To see that the $\eta$- and $\beta\eta$-equivalences are functional let $\mathbf{A}, \mathbf{B} \in wff_{\alpha\rightarrow\beta}(\Omega)$ and $\mathbf{A}@\mathbf{C} \doteq \mathbf{AC} =_{\beta\eta} \mathbf{BC} \doteq \mathbf{B}@\mathbf{C}$ for all $\mathbf{C} \in wff_\alpha(\Omega)$. In particular, we have $\mathbf{A}X =_{\beta\eta} \mathbf{B}X$ for a variable $X \in \mathcal{V}_\alpha$ that is not free in $\mathbf{A}$ and $\mathbf{B}$. By definition we have $\mathbf{A} =_{\beta\eta} \lambda X_\alpha.\mathbf{A}X =_{\beta\eta} \lambda_\alpha \mathbf{B}X =_{\beta\eta} \mathbf{B}$. □

**Definition 2.3.11** Let $\mathbf{A} \doteq (\lambda\overline{X^n}.h\overline{\mathbf{E}^m})$ be a well-formed formula such that $h$ is a constant or variable, then we say that $\mathbf{A}$ is in **head normal form**. The part $\lambda\overline{X^n}$ is called the **binder of A**, the part $h\overline{\mathbf{E}^m}$ the **matrix**, and the constant or variable $h$ is called the **head of A**. We denote the head of $\mathbf{A}$ with $\mathbf{head}(\mathbf{A})$. $\mathbf{A}$ is called **rigid**, iff $h$ is a constant or a bound variable, otherwise **flexible**. If $h$ is the bound variable $X^k$, then $\mathbf{A}$ is called a **($k$-)projection formula**.

A well-formed formula $\mathbf{A}$ that is not in head normal form must be of the form $\mathbf{A} \doteq (\lambda\overline{X}.(\lambda Y.\mathbf{M})\overline{\mathbf{B}^k})$. We call the redex $(\lambda Y.\mathbf{M})\mathbf{B}^1$ the **head redex of A** and the $\beta$-reduction step $(\lambda\overline{X}.(\lambda Y.\mathbf{M})\overline{\mathbf{B}^k}) \rightarrow_\beta (\lambda\overline{X}.[\mathbf{B}^1/Y]\mathbf{M}\mathbf{B}^2 \ldots \mathbf{B}^k)$, which reduces this head redex, a **head reduction step**. We denote the head reduction relation by $\rightarrow^h$. Since $\beta$-reduction is confluent and terminating, the head reduction strategy (restricting $\beta$-reduction to the unique head redex) is complete for $\beta$-reduction to head normal form. This fact is convenient in some situations, where we want to fix a unique $\beta$-reduction sequence. For any formula $\mathbf{A}$ we call the formula $\mathbf{B}$ obtained with a maximal head reduction sequence from $\mathbf{A}$ the **head normal form of A**. Of course the strategy of reducing a term to head normal form and then recursively head reducing the immediate subterms of the matrix yields a complete strategy for full $\beta$-reduction. Note that $\beta$-normal forms are head normal forms, where the subformulae $\mathbf{E}^i$ are also $\beta$-normal forms.

**Definition 2.3.12 (Long $\beta\eta$-Normal Form)** Let $\mathbf{A} \doteq (\lambda\overline{X^n}.h\overline{\mathbf{E}^m})$ be a well-formed formula in head normal form such that the matrix $(h\overline{\mathbf{E}^m})$ is of type $\overline{\beta_k} \rightarrow \gamma$ and $\gamma \in \mathcal{BT}$, then the **$\eta$-expanded form of A** denoted by $\eta[\mathbf{A}]$ is defined to be

$$\lambda X^1 \ldots X^n Y^1_{\beta_1} \ldots Y^k_{\beta_k}.h\mathbf{E}^1 \ldots \mathbf{E}^m Y^1_{\beta_1} \ldots Y^k_{\beta_k} \;,$$

where $Y^i_{\beta_i}$ are new variables of types $\beta_i$. We define the **long head normal form**, denoted by $\mathbf{A}\!\downarrow^h$, of a well-formed formula $\mathbf{A}$ to be the $\eta$-expanded form of the head normal form of $\mathbf{A}$. Similarly the **long $\beta\eta$-normal form** of $\mathbf{A}$, denoted by $\mathbf{A}\!\downarrow$, is the formula that we obtain by recursively extending this process to the arguments $\mathbf{E}^i$ and $Y^j$, thus

$$\mathbf{A}\!\downarrow = \lambda X^1 \ldots X^n Y^1_{\beta_1} \ldots Y^k_{\beta_k}.h\eta\left[\mathbf{E}^1\right] \ldots \eta[\mathbf{E}^m]\eta\left[Y^1_{\beta_1}\right] \ldots \eta\left[Y^k_{\beta_k}\right] \;,$$

**Definition 2.3.13 (Term Structure for $\Omega$)** Let $\mathcal{D}_\mathcal{T}$ be the collection of well-formed formulae in $\beta\eta$-normal form, let $\mathbf{A}@\mathbf{B}$ be the $\beta\eta$-normal form of $\mathbf{AB}$, and $\mathcal{I} := \mathrm{Id}_\Omega$, then we call $\mathcal{TS}(\Omega) := (\mathcal{D}, @, \mathcal{I})$ the **term structure for $\Omega$**.

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

**Lemma 2.3.14** *$\mathcal{TS}(\Omega)$ is a total $\Omega$-structure.*

**Proof:** Note that constants are $\beta\eta$-normal forms, therefore $\mathcal{TS}(\Omega)$ is the quotient structure of *wff*$(\Omega)$ for the relation $=_{\beta\eta}$. It is total and functional by 2.3.3 and 2.3.10. As we have remarked in 2.2.10, *wff*$(\Omega)$ is not comprehension closed, so we cannot use 2.3.3, but have to convince ourselves directly that $\mathcal{TS}(\Omega)$ is comprehension-closed. So let $\varphi$ be an assignment into $\mathcal{TS}(\Omega)$ and $\mathbf{A}$ a well-formed formula. Note that $\sigma := \varphi|_{\mathbf{Free(A)}}$ is a substitution, since $\mathbf{Free}(\mathbf{A})$ is finite. We can convince ourselves that $\mathcal{I}_\varphi(\mathbf{A}) = \sigma(\mathbf{A})\!\downarrow$ by a simple induction on the structure of formulae using

$$\sigma(\lambda X.\mathbf{A})@\mathbf{B} = (\lambda X.\sigma_{-X}\mathbf{A})@\mathbf{B} = \sigma, [\mathbf{B}/X]\mathbf{A} = \mathcal{I}_{\varphi,[\mathbf{B}/X]}(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{A})@\mathbf{B}$$

□

**Remark 2.3.15** Since the $\eta$-expansion relation is a subrelation of the inverse of $\eta$-reduction, the $\eta$-normal forms of $\mathbf{A} \in$ *wff*$(\Omega)$ and $\eta[\mathbf{A}]$ are equal and therefore $\mathbf{A}$ and $\eta[\mathbf{A}]$ are $\eta$-equivalent.

We often use the $\eta$-expanded form rather than the $\eta$-reduced form of the $\beta$-normal form, because this normal form has better closure properties, e.g. if $\mathbf{A}$ is in $\beta$-reduced form, then

$\eta[\mathbf{A}]$ is in $\beta$-reduced form as well, whereas the $\eta$-reduced form need not be. For a detailed discussion we refer to [Sny91].

In the definition of the term structure for $\Omega$, we could also have used long $\beta\eta$-normal forms, however, the interpretation function would then have to be the $\eta$-expansion function.

**Lemma 2.3.16** *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a functional $\Omega$-structure and $X$ be a variable that is not free in $\mathbf{A}$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}X) = \mathcal{I}_\varphi(\mathbf{A})$ for all assignments $\varphi$ into $\mathcal{A}$.*

**Proof:** With 2.2.11 and the fact that $X$ is not free in $\mathbf{A}$ we have

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

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

**Theorem 2.3.17 (Substitution Value Theorem)** *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be an $\Omega$-structure. If the variable $X$ is not bound in a well-formed formula $\mathbf{B}$, then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})$.*

**Proof:** We prove the assertion 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 $\mathbf{CD}$ is entailed by the fact, that substitution and value function are defined inductively on the structure of applications. Furthermore, we have

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

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

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

by inductive hypothesis. Note that $\psi$ and $\varphi$ coincide on the free variables of $\mathbf{A}$, therefore by 2.2.11 we have $\mathcal{I}_{\psi,[\mathcal{I}_\varphi(\mathbf{A})/X]}(\mathbf{D}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{A})/X]}(\lambda Y.\mathbf{D})@a$, which implies the assertion, since $\mathcal{A}$ is functional. □

**Corollary 2.3.18** *If $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ is an $\Omega$-structure and $Y \notin$ **Free**$(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}) = \mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})$ for all assignments $\varphi$ into $\mathcal{A}$.*

**Proof:** We have $\mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})@a = \mathcal{I}_{\varphi,[a/Y]}([Y/X]\mathbf{A}) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathcal{I}_\varphi(\lambda X.\mathbf{A})@a$ with 2.3.17. □

**Corollary 2.3.19** *$\alpha$-conversion is sound in $\Omega$-structures.*

**Definition 2.3.20** We extend the function **head** by the definition **head**$(\mathbf{A}) := k$, iff $\mathbf{A}$ is a $k$-projection formula. Otherwise the function **head** would be undefined for projection formulae, because the its head is some variable, whose name of no role outside the formula (we take alphabetic variants to be identical).

**Corollary 2.3.21** *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be an $\Omega$-structure and $X$ not bound in $\mathbf{A}$, then $\mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) = \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})$ for all assignments $\varphi$ into $\mathcal{A}$.*

**Proof:** We have $\mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) = \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A}) = \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})$ with 2.3.17. □

We combine lemmata 2.3.16 and 2.3.21 to the following soundness result:

**Theorem 2.3.22** *$\beta\eta$-conversion is sound in $\Omega$-structures, i.e. if $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ is an $\Omega$-structure and $\mathbf{A} =_{\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for any assignment $\varphi$.*

We now specialize the notion of $\Omega$-structures to the standard general model semantics for $\Lambda$.

**Definition 2.3.23 (Pre-$\Omega$-Algebra)** A **pre-$\Omega$-algebra** $\mathcal{A} := (\mathcal{D}, \mathcal{I})$ is a pre-$\Omega$-structure $(\mathcal{D}, @, \mathcal{I})$ such that $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{D}_\beta)$ and $f@a = f(a)$. A pre-$\Omega$-algebra is called **full**, iff $\mathcal{D}_{\alpha\to\beta} = \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{D}_\beta)$. It is easy to check that $\mathcal{A}$ is total, iff $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$.

Note that pre-$\Omega$-algebras are functional, since they are defined as structures of mathematical functions. We call a pre-$\Omega$-algebra an **$\Omega$-algebra**, iff it is an $\Omega$-structure, i.e. iff it is comprehension-closed.

## 2.4 Type Inference

We now recast the definitions of well-formed formulae and $\lambda$-conversion as type inference systems. The notion of type inference system helps the analysis of $\lambda$-calculi such as $\Sigma\mathcal{HOL}$, where the type (or in our case the sort) does not only depend on the structure of the formula. Type inference systems define judgments: in the case of the simply typed $\lambda$-calculus the primary judgment is $\Omega \vdash \mathbf{A}{:}\,\alpha$, which means "the formula $\mathbf{A}$ has type $\alpha$". In contrast to definition 2.2.1 this judgment is defined inductively on the structure of the derivation of $\Omega \vdash \mathbf{A}{:}\,\alpha$ rather than on the structure of $\mathbf{A}$. In the simply typed $\lambda$-calculus these notions of inductions coincide, since the type inference system is structural, but in the case of $\Sigma\Lambda$ they do not and the notion of an inference system is necessary to define well-sortedness.

**Definition 2.4.1 (Inference System)** Each logical system is built up from syntactic objects like types, formulae, variables. These objects are called **raw** types, formulae,.... Let $\mathcal{J}$ be a relation on these entities, then we call $\mathcal{J}$ a **judgment schema** and a tuple $(\mathcal{E}^1, \ldots, \mathcal{E}^n) \in \mathcal{J}$ a **judgment**. An **inference rule** is an effectively computable relation

$$\mathcal{R} = \{(\mathcal{C}^1, \ldots, \mathcal{C}^m, \mathcal{D}) \mid \mathcal{C}^i \in \mathcal{J}^i, \mathcal{D} \in \mathcal{K}\}$$

on judgments such that $\mathcal{J}^i$ and $\mathcal{K}$ are judgment schemata. Inference rules are traditionally represented by a set of schemata of the form

$$\frac{\mathcal{C}^1, \ldots, \mathcal{C}^m}{\mathcal{D}}\,\mathcal{R}$$

where the part $\mathcal{C}^1, \ldots, \mathcal{C}^m$ is called **antecedent** and $\mathcal{D}$ is called the **succedent**. In order to give a finite presentation of an inference system the schemata may be schematic in the syntactic objects. Inference rules with empty antecedent are called **axioms** and otherwise **proper inference rules**. An **inference system** $\mathcal{I}$ is a finite set of **inference rules**.

**Definition 2.4.2 ($\mathcal{I}$-Derivation)** Let $\mathcal{I}$ be an inference system and $\mathcal{D}$ be a finite tree, where each node $\mathcal{N}$ in $\mathcal{D}$ is labeled with a triple $(\mathcal{J}, R, \{\mathcal{J}^1, \ldots, \mathcal{J}^n\})$ such that $R \in \mathcal{I}$ and $(\mathcal{J}^1, \ldots, \mathcal{J}^n, \mathcal{J}) \in R$. $\mathcal{J}$ is called the **assertion**, $R$ the **justification**, and the set of $\mathcal{J}^i$ the **support** of $\mathcal{N}$. $\mathcal{D}$ is called an **$\mathcal{I}$-derivation**, iff each node $\mathcal{N}$ with label $(\mathcal{J}, R, \{\mathcal{J}^1, \ldots, \mathcal{J}^n\})$ has $n$ children $\mathcal{N}^i$ with assertions $\mathcal{J}^i$. Note that, since the leaves of these trees have no children, they have to be labeled with axioms. Because of the tree nature, we often call an $\mathcal{I}$-derivation a **proof-tree**.

Let $\mathcal{J}$ be a judgment and $\Phi$ be a set of judgments. We call an $\mathcal{I}$-derivation $\mathcal{D}$ an **$\mathcal{I}$-derivation of $\mathcal{J}$ from the set $\Phi$ of hypotheses**, if $\mathcal{J}$ is the assertion of the root of $\mathcal{D}$ and the supports of the leaves of $\mathcal{D}$ are subsets of $\Phi$. If there exists an $\mathcal{I}$-derivation $\mathcal{D}$ of $\mathcal{J}$ from $\Phi$, then we write $\Phi \vdash_{\mathcal{I}} \mathcal{J}$ or $\mathcal{D}\colon \Phi \vdash_{\mathcal{I}} \mathcal{J}$, if we want to specify the derivation. If $\mathcal{D}$ only consists of a single node labeled with the inference rule $\mathcal{R}$, then we often write $\Phi \vdash_{\mathcal{R}} \mathcal{J}$.

We will frequently prove our theorems by induction on the structure of the derivations involved in the assertion. For this we will use the **structural ordering** on derivations, which we can define by $\mathcal{D} \prec \mathcal{E}$ iff $\mathcal{D}$ is a subtree of $\mathcal{E}$.

**Definition 2.4.3** Let $\mathcal{I}$ be an inference system, then an inference rule $\mathcal{R}$ is called **admissible in** $\mathcal{I}$, iff adding $\mathcal{R}$ to $\mathcal{I}$ does not change the set of judgments derivable from a given set of hypotheses. An inference rule $\mathcal{R}$ is called **derivable in** $\mathcal{I}$, iff for each $\mathcal{J}^1, \ldots, \mathcal{J}^n \vdash_{\mathcal{R}} \mathcal{J} \in \mathcal{R}$ there already exists an $\mathcal{I}$-derivation $\mathcal{J}^1, \ldots, \mathcal{J}^n \vdash_{\mathcal{I}} \mathcal{J}$.

Thus admissible rules can be added to an inference system without changing its theoretical properties. For practical reasoning applications, these added rules can sometimes make life much easier. Clearly, derivable rules of inference are also admissible.

We now give an inference system for the judgments introduced so far. The judgment $\Omega \vdash \mathbf{A}\colon \alpha$ holds, iff $\mathbf{A}$ is a formula of type $\alpha$. Since the type inference system defines the well-formed formulae of $\Lambda$ (those that have a type $\alpha$), we also call it $\Lambda$. We will maintain this practice in the following and identify the names of the logical systems with those of the inference systems defining them.

**Definition 2.4.4 (Type Inference System $\Lambda$)** The syntactic category of **raw formulae** consists of untyped constants, variables, applications, and abstractions, and the inference system $\Lambda$ for the judgment schema of **well-typedness** $\Omega \vdash \mathbf{A}\colon \alpha$ is given by the following schemata:

$$\frac{c \in \Omega_\alpha}{\Omega \vdash c\colon \alpha}\, \mathit{wff{:}const} \qquad \frac{X \in \mathcal{V}_\alpha}{\Omega \vdash X\colon \alpha}\, \mathit{wff{:}var}$$

$$\frac{\Omega \vdash \mathbf{A}\colon \beta \to \alpha \quad \Omega \vdash \mathbf{B}\colon \beta}{\Omega \vdash \mathbf{AB}\colon \alpha}\, \mathit{wff{:}app} \qquad \frac{\Omega \vdash \mathbf{A}\colon \alpha}{\Omega \vdash (\lambda X_\beta.\mathbf{A})\colon \beta \to \alpha}\, \mathit{wff{:}abs}$$

We sometimes call $\Lambda$-derivations **typing proofs**, since they prove the typing judgment of the root node. It is obvious that this inference system for the well-typedness judgment is correct and complete with respect to the definition 2.2.1. Thus we can alternatively use this type inference system as a definition of well-typed formulae by specifying that any raw formula $\mathbf{A}$ is called **well-typed**, iff the judgment $\Omega \vdash \mathbf{A}\colon\alpha$ is derivable in $\Lambda$. Note that just like definition 2.2.1 this is an inductive definition. However, in contrast to the old definition this formally is inductive on the structure of $\Lambda$-derivations rather than on the structure of formulae.

**Remark 2.4.5** By inspection of the type inference system $\Lambda$ above we see that the formulae in the succedent of the inference rules are partitioned by the four possibilities for the structure of formulae (variable, constant, application, and abstraction), which is a disjoint partition of formulae. Thus the root node of any proof of a judgment $\Omega \vdash \mathbf{A}\colon\alpha$ is uniquely determined by the structure of $\mathbf{A}$, and consequently, there is a tree-isomorphism between any typing proof for $\Omega \vdash \mathbf{A}\colon\alpha$ and the formula $\mathbf{A}$ itself (when viewed as a tree). We call inference systems where this is the case **structural**. Note that structural type inference systems can be very conveniently inverted into type inference algorithms that recursively analyze the structure of formulae. For an example of a non-structural type inference system see $\Sigma\Lambda$ in definition 3.2.7.

Now we use inference systems to analyze subsystems of $\lambda$-conversion.

**Definition 2.4.6 (One-Step, Top-Level Reductions)** $\lambda$-reduction can be formalized with a judgment $\Omega \vdash \mathbf{A} \to_\lambda \mathbf{B}$, which is given by the following inference rules for top-level reduction together with the inference rules of 2.4.7 that extend these to the full $\lambda$-reduction judgments.

$$\frac{}{\Omega \vdash (\lambda X_\beta.\mathbf{A})\mathbf{B} \to^t_\beta [\mathbf{B}/X_\beta]\mathbf{A}}\ \mathit{wff{:}\beta{:}top}$$

$$\frac{X_\beta \notin \mathbf{Free}(\mathbf{A}) \quad \Omega \vdash \mathbf{A}\colon\beta\to\alpha}{\Omega \vdash (\lambda X_\beta.\mathbf{A}X) \to^t_\eta \mathbf{A}}\ \mathit{wff{:}\eta{:}top}$$

Here we use the judgment $Y \in \mathbf{Free}(\mathbf{A})$, which we have defined in 2.2.4. We can recast this as a structural inference system

$$\frac{}{Y \in \mathbf{Free}(Y)} \qquad \frac{Y \in \mathbf{Free}(\mathbf{A})}{Y \in \mathbf{Free}(\mathbf{AB})} \qquad \frac{Y \in \mathbf{Free}(\mathbf{B})}{Y \in \mathbf{Free}(\mathbf{AB})}$$

$$\frac{Y \in \mathbf{Free}(\mathbf{A}) \qquad X \neq Y}{Y \in \mathbf{Free}(\lambda X.\mathbf{A})}$$

These top-level $\lambda$-reduction relations can be augmented to a $\lambda$-equality relation in a very general manner, which we present in the following definitions.

**Definition 2.4.7 (Multi-Step Reduction)** Let $\mathcal{R}(\mathbf{A}, \mathbf{B})$ be a relation on well-formed formulae given by an inference system (such as $\mathbf{A} \rightarrow_\lambda \mathbf{B}$), then we obtain the **term relation** by adding the following inference rules for congruence closure

$$\frac{\mathcal{R}(\mathbf{A}, \mathbf{B})}{\mathcal{R}(\mathbf{AC}, \mathbf{BC})}\,\mathit{tr{:}app{:}fn} \qquad \frac{\mathcal{R}(\mathbf{A}, \mathbf{B})}{\mathcal{R}(\mathbf{CA}, \mathbf{CB})}\,\mathit{tr{:}app{:}arg}$$

$$\frac{\mathcal{R}(\mathbf{A}, \mathbf{B})}{\mathcal{R}(\lambda X.\mathbf{A}, \lambda X.\mathbf{B})}\,\mathit{tr{:}abs}$$

and the **multi-step relation** by adding the rules for transitivity and reflexivity

$$\frac{\mathcal{R}(\mathbf{A}, \mathbf{B}) \quad \mathcal{R}(\mathbf{B}, \mathbf{C})}{\mathcal{R}(\mathbf{A}, \mathbf{C})}\,\mathit{ms{:}trans} \qquad \frac{}{\mathcal{R}(\mathbf{A}, \mathbf{A})}\,\mathit{ms{:}ref}$$

and the **congruence relation** with the rule for symmetry

$$\frac{\mathcal{R}(\mathbf{A}, \mathbf{B})}{\mathcal{R}(\mathbf{B}, \mathbf{A})}\,\mathit{eq{:}sym}$$

For any derivation $\mathcal{D}$ in a multi-step inference system, we define the **length of** $\mathcal{D}$ (written as $\mathbf{ln}(\mathcal{D})$) to be the number of *ms:trans* nodes in $\mathcal{D}$.

**Notation 2.4.8** We mostly use the previous definition to extend the $\lambda$-reduction relations $\mathcal{R}(\mathbf{A}, \mathbf{B}) := \mathbf{A} \rightarrow_\lambda^t \mathbf{B}$. We write the corresponding term relations as $\rightarrow_\lambda$, the multi-step relations as $\rightarrow_\lambda^*$, and the congruence relations as $=_\lambda$. Here $\lambda \in \{\beta, \eta, \beta\eta\}$.

# 3 ΣΛ: A Sorted λ-Calculus

In this section we define a sorted λ-calculus ΣΛ with term declarations and functional base sorts. This system is a generalization of the first-order system described in [SS89] and the system Λ presented in section 2. We start out with the system Λ and add more syntactic information to the formulae in order to distinguish certain well-typed formulae as well-sorted. Since well-sortedness is not a structural property, we give a sort inference system for well-sortedness.

As we have mentioned in 1.1 types were developed as a syntactic means to distinguish mathematical objects of fundamentally differing nature and thereby eliminate antinomies and paradoxes from the formal system. We have seen in 1.3 that types (as a mechanism equivalent to flat sort hierarchies) can also serve as a powerful representation mechanism that allows to formalize disjoint sets as differing types. In ΣΛ we want to make the type mechanism more expressive without loosing the safety aspect in terms of antinomies of a type system. Thus we separate both aspects into a simple type system (for the safety aspect) and a sort system (for the additional expressiveness). In fact, we only need one base type symbol $\iota$ for individuals in the presence of a sort system, since all other distinctions can and – as the author believes – should be made within the sort system. We will violate this intuition by introducing a second base type $o$ for truth values, when we instantiate ΣΛ to a logical system in section 5, since in this case it is more convenient to do so.

Clearly the sort system has to conform to the type system in some way, in order to ensure that no antinomies can be imported via the sort declarations. In our case the sort system is a refinement of the underlying type system and sorted operations will turn out to be refinements of their unsorted counterparts. In particular, well-sorted formulae are still well-typed.

## 3.1 Sorts

Sorting the universe of individuals gives rise to new classes of functions, whose domains and codomains are just the sorts. In addition to this essentially first-order way of sorting the function universes, the classes of functions defined by domains and codomains can be further divided into subclasses, since functions are explicit objects of type theory. Sorts of functional type, i.e. base sorts that denote classes of functions, are introduced. Syntactically each sort comes with a type, and – if it is of functional type – also with domain and codomain sorts.

**Definition 3.1.1 (Sort System)** A **sort system** is a quadruple $(\mathcal{S}, \mathcal{BS}, \mathfrak{r}, \mathfrak{d})$, where

1. $\mathcal{BS} := \mathcal{BS}_{\mathcal{T}} := \{\mathcal{BS}_\alpha \mid \alpha \in \mathcal{T}\}$ is a typed collection of sets of symbols, called **base sorts**, which we assume not to contain any types (we always want to be able to distinguish sorts and types).

2. the collection **sorts** $\mathcal{S}$ is the closure of $\mathcal{BS}$ under function construction, i.e. $\mathcal{S}$ is a typed collection that contains $\mathcal{BS}$ such that for any $\mathbb{A} \in \mathcal{S}_\alpha$ and $\mathbb{B} \in \mathcal{S}_\beta$, we have $\mathbb{A} \to \mathbb{B} \in \mathcal{S}_{\alpha\to\beta}$. Note that if all $\mathcal{BS}_\alpha$ are finite, then so is each $\mathcal{S}_\beta$.

3. the **domain sort function** $\mathfrak{d}$ is a function $\mathfrak{d}\colon \mathcal{BS}_{\alpha\to\beta} \longrightarrow \mathcal{S}_\alpha$.

4. the **codomain sort function** $\mathfrak{r}$ is a function $\mathfrak{r}: \mathcal{BS}_{\alpha\to\beta} \longrightarrow \mathcal{S}_\beta$.

If the context is clear we will often denote the sort system $(\mathcal{S}, \mathcal{BS}, \mathfrak{d}, \mathfrak{r})$ only by $\mathcal{S}$.

**Definition 3.1.2** Let $\mathcal{S}$ be a sort system and $\mathbb{A} \in \mathcal{S}$. Remember that $\mathcal{S}$ is a typed collection (cf. 2.1.8) induces a type function. We call the type $\tau(\mathbb{A}) \in \mathcal{T}$ the **type of the sort** $\mathbb{A}$. If $\tau(\mathbb{A}) \notin \mathcal{BT}$, then we call $\mathbb{A}$ a **functional sort** ($\mathbb{A} \in \mathcal{S}^f$) and otherwise **non-functional** ($\mathbb{A} \in \mathcal{S}^{nf}$).

For the structure theorem and the definition of general bindings we will need the notion of the **length** of a sort. We set $\mathbf{ln}(\mathbb{A}) := 0$, iff $\mathbb{A} \in \mathcal{BS}$ and $\mathbf{ln}(\mathbb{A} \to \mathbb{B}) := 1 + \mathbf{ln}(\mathbb{B})$ otherwise. Thus the length intuitively is the number of top level arrows $\to$ in a sort. Although this definition is analogous to the definition of length for types, in general we have $\mathbf{ln}(\mathbb{A}) \neq \mathbf{ln}(\tau(\mathbb{A}))$ due to the existence of functional base sorts.

**Notation 3.1.3** We denote sorts with uppercase symbols like $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, or $\mathbb{D}$. For $\mathbb{C} := (\mathbb{A} \to \mathbb{B}) \in \mathcal{S}$ we define $\mathfrak{r}(\mathbb{C}) := \mathbb{B}$, $\mathfrak{d}(\mathbb{C}) := \mathbb{A}$, thus we can extend the functions $\mathfrak{r}$ and $\mathfrak{d}$ to $\mathcal{S}^f$. Furthermore, we use $\mathfrak{r}$ and $\mathfrak{d}$ on types in the obvious way. For the rest of this thesis we fix a sort system $\mathcal{S} = (\mathcal{S}, \mathcal{BS}, \mathfrak{d}, \mathfrak{r})$.

We often use the shorthands $\mathfrak{d}^k(\mathbb{A})$ and $\mathfrak{r}^k(\mathbb{A})$ for the $k^{\text{th}}$ **domain sort** and the $k$**-fold codomain sort of** $\mathbb{A}$, which we inductively define by

$$\begin{array}{rclcrcl} \mathfrak{r}^0(\mathbb{A}) & = & \mathbb{A} & \quad & \mathfrak{r}^{i+1}(\mathbb{A}) & = & \mathfrak{r}(\mathfrak{r}^i(\mathbb{A})) \\ \mathfrak{d}^0(\mathbb{A}) & = & \mathbb{A} & \quad & \mathfrak{d}^{i+1}(\mathbb{A}) & = & \mathfrak{d}(\mathfrak{r}^i(\mathbb{A})) \end{array}$$

**Definition 3.1.4 (Trivially Sorted)** Since we are ultimately interested in sorted formulae and their typed counterparts, we only consider sort systems where $\tau$ is surjective. Thus we can pick a local inverse, i.e. an injection $\sharp: \mathcal{T} \longrightarrow \mathcal{S}$. We sometimes also denote $\tau$ by $\flat$, when we want to stress the property of being an inverse of $\sharp$.

We call a sort system **trivially sorted**, iff $\tau: \mathcal{BS} \longrightarrow \mathcal{BT}$ is a bijection. In this case there can be no functional base sorts, since all base sorts are of base type. Moreover the functions $\flat$ and $\sharp$ are bijections and inverses, so the sort system $\mathcal{S}$ is isomorphic to $\mathcal{T}$ and the functions $\mathfrak{d}$ and $\mathfrak{r}$ are trivial. If we only have one base type, then we call the sort system **one-sorted**.

It will be important that the signatures, over which our well-sorted formulae are built, "respect function domains", i.e. that for any formula $\mathbf{A}$ and any sorts $\mathbb{A}$ and $\mathbb{B}$ such that $\mathbf{A}$ has sort $\mathbb{A}$ and sort $\mathbb{B}$ at the same time, the domain sorts $\mathfrak{d}(\mathbb{A})$ and $\mathfrak{d}(\mathbb{B})$ are identical. The proof that signatures indeed satisfy this property (see theorem 3.5.4) depends on the fact that term declarations meet the sort condition of *ws:td* in definition 3.2.7 below. This sort condition is given in terms of the equivalence relation **Rdom** , which we now define.

**Definition 3.1.5** We say that sorts $\mathbb{A}$ and $\mathbb{B}$ have **equal domains** ($\mathbb{A}$ **Rdom** $\mathbb{B}$), if either $\mathbb{A}, \mathbb{B} \in \mathcal{S}^{nf}$ and $\tau(\mathbb{A}) = \tau(\mathbb{B})$, or $\mathfrak{r}(\mathbb{A})$ **Rdom** $\mathfrak{r}(\mathbb{B})$ and $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$. Note that $\mathbb{A}$ **Rdom** $\mathbb{B}$, iff $\mathfrak{d}^i(\mathbb{A}) = \mathfrak{d}^i(\mathbb{B})$ for all $1 \leq i \leq k$ such that $\mathfrak{r}^k(\mathbb{A})$ and $\mathfrak{r}^k(\mathbb{B})$ are of the same base type. Thus **Rdom** is an equivalence relation that respects types, i.e. $\mathbb{A}$ **Rdom** $\mathbb{B}$ can only hold, if $\tau(\mathbb{A}) = \tau(\mathbb{B})$. For trivially sorted sort systems **Rdom** is just equality.

**Remark 3.1.6** We have kept the sort system of $\Sigma\Lambda$ as simple as possible for this theoretical exposition. For a practical system we would like to have features like sort constructors, such as $\mathcal{C}: \mathcal{S} \times \mathcal{S} \to \mathcal{S}$, which allows the construction of new base sorts such as $\mathcal{C}(\mathbb{R}, \mathbb{R})$ that stands for the continuous functions with domain $\mathbf{IR} = \mathcal{D}_{\mathbb{R}}$ and codomain $\mathbf{IR}$ from the sort $\mathbb{R}$ of real numbers. Another practical improvement would be the introduction of intersection sorts [KP93, Wei93], which would make our signatures regular (cf. 3.6.16). We believe that, while this would add considerably to the complexity and practical descriptive power of the system, the theory can be dealt with by simple extensions of the methods described here.

## 3.2 Well-Sorted Formulae

Next we introduce the concept of well-sortedness for well-formed formulae. It is defined with respect to a variable context, which gives local sort information for the variables, and a signature, which contains sort information for formula schemata (term declarations). A formula $\mathbf{A}$ be called well-sorted with respect to a signature $\Sigma$ and a context $\Gamma$, iff the judgment $\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ is derivable in the inference system $\Sigma\Lambda$. The variable context now explicitly assigns sorts to variables that are only implicitly typed in $\Sigma\Lambda$.

One of the difficulties in devising a formal system with term declarations is that the signature, needed for defining well-sortedness, itself contains formulae that have to be well-sorted. We have another case of recursive dependency: on the one hand we need well-sortedness conditions on the inference rules for sorted $\beta\eta$-reduction, and on the other hand we need an inference rule $ws{:}\beta\eta$ that guarantees formulae that are $\beta\eta$-equivalent to have identical sets of sorts. Therefore we need to combine the inference systems for valid signatures, well-sortedness, and sorted $\beta\eta$-reduction into one large system $\Sigma\Lambda$.

**Definition 3.2.1 (Variable Context)** Let $X_\alpha$ be a variable and $\mathbb{A}$ a sort, then we call a pair $[X :: \mathbb{A}]$ a **variable declaration** for $X$, iff $\tau(\mathbb{A}) = \alpha$. We call a typed, partial function from variables to sorts a **(variable) context**. Thus a variable context is just a set of variable declarations.

**Notation 3.2.2** We write the fact that a typed partial function $\Gamma: \mathcal{V} \longrightarrow \mathcal{S}$ is a variable context as $\vdash_{ctx} \Gamma$, and we generally use the symbols $\Gamma$, $\Delta$, and $\Xi$ for variable contexts. With our convention from 2.1.3 we have $\Gamma(X) = \mathbb{A}$ for $\Gamma := \Gamma', [X :: \mathbb{A}]$, even if $\Gamma'(X) = \mathbb{B}$. We sometimes abbreviate contexts $[X^1 :: \mathbb{A}], \ldots, [X^n :: \mathbb{A}]$ by $[X^1, \ldots, X^n :: \mathbb{A}]$ in order to conserve space and increase legibility.

For most of our purposes we will only need finite variable contexts, but in section 5 we will need infinite variable contexts to construct suitable term models.

**Definition 3.2.3 (Raw ΣΛ-formulae)** **Raw ΣΛ-formulae** are well-typed Λ-formulae, where λ-abstractions have the form $(\lambda X_{\mathbb{A}}.\mathbf{A})$. To make this definition formal, we extend the trivial injections $\sharp$ and $\flat$ (cf. 3.1.4) of sorts and types to formulae by

1. $\sharp(c) := \flat(c) := c$ for $c \in \Omega$
2. $\sharp(X) := \flat(X) := X$ for $X \in \mathcal{V}$
3. $\sharp([\mathbf{AB}]) := [\sharp(\mathbf{A})\sharp(\mathbf{B})]$ and $\flat([\mathbf{AB}]) := [\flat(\mathbf{A})\flat(B)]$

4. $\sharp([\lambda X_\alpha.\mathbf{A}]) := [\lambda X_{\sharp(\alpha)}.\sharp(\mathbf{A})]$ and $\flat([\lambda X_{\mathbb{A}}.\mathbf{A}]) := [\lambda X_{\flat(\mathbb{A})}.\flat(\mathbf{A})]$

These homomorphisms are so trivial (they only add or delete sort information in abstractions), that we will often keep them implicit and directly consider $\Sigma\Lambda$-formulae as $\Lambda$-formulae.

**Definition 3.2.4 (Term Declaration)** We call a triple $[\forall\Gamma.\mathbf{A}::\mathbb{A}]$ consisting of a variable context $\Gamma$, a raw formula $\mathbf{A}$, and a sort $\mathbb{A}$ a **term declaration** and a finite set of term declarations a **signature**.

The idea of term declarations consists in the intuition, that there can be additional sort information within the structure of a formula, as the following example shows. Consider, for instance, the addition function, which we (semantically) would like to have the sort $\mathbb{N}\times\mathbb{N}\longrightarrow\mathbb{N}$, where $\mathbb{N}$ is the sort of natural numbers. If we also have a sort for the evens $\mathbb{E}$, then we might want to specify that the expression $[+aa]$ is an even number, even if $a$ is not. This information can be formalized by declaring the formula $[+X_{\mathbb{N}}X_{\mathbb{N}}]$, to be of sort $\mathbb{E}$ (an even number) using a term declaration. We might also want to give the addition function the sort $\mathbb{E}\times\mathbb{E}\rightarrow\mathbb{E}$, however, since it is central to our program that formulae have unique domain sorts, we cannot declare this directly in the signature. Closer inspection of the semantics behind our example reveals, that it is consistent with our program to declare the restriction of the addition function to the even numbers has codomain in $\mathbb{E}$, which we can legally declare with the term declaration $[\forall[X::\mathbb{E}],[Y::\mathbb{E}].+XY::\mathbb{E}]$.

**Definition 3.2.5 (Sorted $\alpha$-Conversion)** In $\Sigma\Lambda$ we cannot simply take typed $\alpha$-conversion, since this would not conserve well-sortedness. Consider, for instance, the formulae $\lambda X_{\mathbb{A}}.\mathbf{A}$ and $\lambda Y_{\mathbb{B}}.[Y/X]\mathbf{A}$, which are typed $\alpha$-variants, if $\tau(\mathbb{A})=\tau(\mathbb{B})$. They do not have the same sorts in $\Sigma\Lambda$, thus a formula containing the first as a subformula would become ill-sorted, if it were to be replaced by the second. For a sorted $\alpha$-conversion relation we define raw formulae $\lambda X_{\mathbb{A}}.\mathbf{A}$ and $\lambda Y_{\mathbb{A}}.[Y/X]\mathbf{A}$ to be alphabetic variants.

**General Assumption 3.2.6 (Implicit $\alpha$-Conversion)** Just like in the system $\Lambda$ in assumption 2.3.7, we consider sorted $\alpha$-conversion as built into the system to avoid variable capture during instantiation. Thus we regard $\Sigma\Lambda$-formulae as syntactically equal, iff they are sorted alphabetical variants. Note that this assumption can be justified with exactly the same argument as the one for assumption 2.3.7.

Since the context in a term declaration is a kind of declaration that locally binds variables, we also assume implicit $\alpha$-conversion (i.e. the term declarations $[\forall\Gamma,[X::\mathbb{A}].\mathbf{A}::\mathbb{A}]$ and $[\forall\Gamma,[Y::\mathbb{A}].[Y/X]\mathbf{A}::\mathbb{A}]$ are alphabetic variants) for term declarations and consider alphabetical variants as identical.

**Definition 3.2.7 (Inference System for $\Sigma\Lambda$)** We define **well-sorted formulae** by an inference system for the judgments

- $\vdash_{sig}\Sigma$ ($\Sigma$ is a valid signature)
- $\Gamma\vdash_\Sigma \mathbf{A}::\mathbb{A}$ (in $\Sigma$ formula $\mathbf{A}$ has sort $\mathbb{A}$ assuming $\Gamma$)
- $\Gamma\vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$ ($\mathbf{B}$ can be obtained from $\mathbf{A}$ by sorted $\beta\eta$-conversion in $\Sigma$ assuming $\Gamma$)

We say that a signature $\Sigma$ is **valid**, iff there is a $\Sigma\Lambda$-derivation of $\vdash_{sig} \Sigma$. For a fixed signature $\Sigma$ and a context $\Gamma$ we say that a formula $\mathbf{A}$ **is of sort** $\mathbb{A}$, iff $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, and that $\mathbf{A}$ is **well-sorted**, iff there is a sort $\mathbb{A}$ such that $\mathbf{A}$ has sort $\mathbb{A}$, otherwise we call $\mathbf{A}$ **ill-sorted** in $\Gamma$ and $\Sigma$. We fix the notation $\mathcal{S}^\Gamma_\Sigma(\mathbf{A}) := \{\mathbb{A} \in \mathcal{S} \mid \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}\}$ for the **set of sorts** of $\mathbf{A}$, and $wsf_{\mathbb{A}}(\Sigma,\Gamma)$ for the set of formulae of sort $\mathbb{A}$.

The inference system $\Sigma\Lambda$ has the rules

$$\frac{}{\vdash_{sig} \emptyset}\ sig{:}empty$$

$$\frac{\vdash_{sig} \Sigma \quad c \notin \Sigma \quad \mathbb{A} \in \mathcal{S} \quad \tau(\mathbb{A}) = \alpha}{\vdash_{sig} \Sigma, [c::\mathbb{A}]}\ sig{:}const$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}}{\vdash_{sig} \Sigma, [\forall\Gamma.\mathbf{A}::\mathbb{B}]}\ sig{:}td$$

for the judgment $\vdash_{sig} \Sigma$. Here the second rule allows the introduction of initial sort declarations for constants that have never appeared before, whereas the rule *sig:td* for proper term declarations allows the declaration of further sort information for well-sorted formulae, if the new sort $\mathbb{B}$ respects function domains. Note that in this rule we do not have to require $\vdash_{sig} \Sigma$, since $\mathbf{A}$ can only be proven to be well-sorted, if $\Sigma$ is valid. The previous definition needs the judgment of well-sorted formulae, which we define with the next set of inference rules:

$$\frac{\vdash_{sig} \Sigma \quad \vdash_{ctx} \Gamma \quad \Gamma(X) = \mathbb{A}}{\Gamma \vdash_\Sigma X::\mathbb{A}}\ ws{:}var \qquad \frac{\vdash_{sig} \Sigma \quad [\forall\Delta.\mathbf{A}::\mathbb{A}] \in \Sigma \quad \Delta \subseteq \Gamma}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\ ws{:}td$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Delta \vdash_\Sigma \mathbf{B}::\mathfrak{d}(\mathbb{A}) \quad \Gamma || \Delta}{\Delta \cup \Gamma \vdash_\Sigma (\mathbf{AB})::\mathfrak{r}(\mathbb{A})}\ ws{:}app$$

$$\frac{\Gamma, [X::\mathbb{B}] \vdash_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{B}}.\mathbf{A})::\mathbb{B} \to \mathbb{A}}\ ws{:}abs$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B} \quad \Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}}\ ws{:}\beta\eta$$

Note that the rules for variables, application, and abstraction are the obvious generalizations of the corresponding rules for simple type theory. In the setting with term declarations we do not need a separate rule for constants, since all constants have to be declared in term declarations.

Now we define the judgment $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$ for sorted $\beta\eta$-equality with the following set of inference rules. The one-step top-level reduction rules

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A} \quad X \notin \mathbf{Free}(\mathbf{A})}{\Gamma \vdash_\Sigma (\lambda X_{\mathfrak{d}(\mathbb{A})} . \mathbf{A} X) \rightarrow^t_\eta \mathbf{A}} \, sort{:}\eta{:}top$$

$$\frac{\Gamma, [X :: \mathbb{B}] \vdash_\Sigma \mathbf{A} :: \mathbb{A} \quad \Delta \vdash_\Sigma \mathbf{B} :: \mathbb{B} \quad \Gamma || \Delta}{\Gamma \cup \Delta \vdash_\Sigma (\lambda X_{\mathbb{B}} . \mathbf{A}) \mathbf{B} \rightarrow^t_\beta [\mathbf{B}/X] \mathbf{A}} \, sort{:}\beta{:}top$$

are turned into a congruence judgment by the following inference rules, which are a specialization of those in 2.4.7 to our setting.

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_\lambda \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A} \quad \Gamma' \vdash_\Sigma \mathbf{C} :: \mathfrak{d}(\mathbb{A}) \quad \Gamma || \Gamma'}{\Gamma \cup \Gamma' \vdash_\Sigma \mathbf{A}\mathbf{C} \rightarrow_\lambda \mathbf{B}\mathbf{C}} \, tr{:}app{:}fn$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_\lambda \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{d}(\mathbb{A}) \quad \Gamma' \vdash_\Sigma \mathbf{C} :: \mathbb{A} \quad \Gamma || \Gamma'}{\Gamma \cup \Gamma' \vdash_\Sigma \mathbf{C}\mathbf{A} \rightarrow_\lambda \mathbf{C}\mathbf{B}} \, tr{:}app{:}arg$$

$$\frac{\Gamma, [X :: \mathbb{A}] \vdash_\Sigma \mathbf{A} \rightarrow_\lambda \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{A}} . \mathbf{A} \rightarrow_\lambda \lambda X_{\mathbb{A}} . \mathbf{B}} \, tr{:}abs$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_\lambda \mathbf{B} \quad \Gamma' \vdash_\Sigma \mathbf{B} \rightarrow^*_\lambda \mathbf{C} \quad \Gamma || \Gamma'}{\Gamma' \cup \Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_\lambda \mathbf{C}} \, ms{:}trans \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}}{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_\lambda \mathbf{A}} \, ms{:}ref$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} =_\lambda \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_\lambda \mathbf{A}} \, eq{:}sym$$

We need these rules, since we view $\beta\eta$-conversion as basic to our system, therefore we do not want $\beta\eta$-conversion to increase the sort of a subformula, and thereby possibly convert a well-sorted formula to an ill-sorted one. In the definition of sorted $\eta$-reduction we have taken care to identify the (unique) supporting sort $\mathfrak{d}(\mathbb{A})$ of $\mathbf{A}$, since the formula $\lambda X_{\mathbb{B}} . \mathbf{A} X$ denotes the restriction of the function $\mathbf{A}$ to sort $\mathbb{B}$, if $\mathbb{B}$ is a subsort of $\mathfrak{d}(\mathbb{A})$.

**Remark 3.2.8** By defining the sets of sorts as typed collections we have enforced that the sorts refine an existing type structure. We could have defined the sort system without reference to types by making the domain sort and codomain sort information part of the signature (cf. 3.2.7), but we prefer to have the type information as a useful intuition in the background and keep the definition of the sorted signatures comparatively simple.

The following lemma is useful for carrying out proofs by induction over the structure of $\Sigma\Lambda$-derivations.

**Lemma 3.2.9** *If $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ ends in a ws:td-node, then either $\mathbf{A}$ is a constant or there is a signature $\Sigma' \subseteq \Sigma$ and a context $\Gamma \subseteq \Gamma$ such that $\Gamma' \vdash_{\Sigma'} \mathbf{A}{::}\mathbb{B}$ for some $\mathbb{B}$ with $\mathbb{B}$ $\mathbf{Rdom}$ $\mathbb{A}$.*

**Proof:** Since $\mathcal{A}$ ends in *ws:td*, we know that there is a term declaration $[\forall\Delta.\mathbf{A}{::}\mathbb{A}] \in \Sigma$ and a $\Sigma\Lambda$-derivation $\mathcal{D}\colon \vdash_{sig} \Sigma$. We show the assertion by induction on the structure of $\mathcal{D}$. $\Sigma$ is nonempty, so *sig:empty* cannot apply. If $\mathcal{A}$ ends in *sig:const*, then $\mathbf{A}$ is a constant. If $\mathcal{D}$ ends in *sig:td*, then we have the following situation

$$\frac{\dfrac{\mathcal{D}'}{\Xi \vdash_\Sigma \mathbf{B}{::}\mathbb{C}} \quad \mathbb{C}\ \mathbf{Rdom}\ \mathbb{B}}{\vdash_{sig} \Sigma', [\forall\Xi.\mathbf{B}{::}\mathbb{C}]}\ \textit{sig:td}$$

Thus we have to consider two cases: if $\Delta = \Xi$, $\mathbf{A} = \mathbf{B}$, and $\mathbb{A} = \mathbb{B}$, then $\mathcal{D}'\colon \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{C}$ and $\mathbb{C}$ $\mathbf{Rdom}$ $\mathbb{B}$ $\mathbf{Rdom}$ $\mathbb{A}$, which gives the assertion, since $\mathbf{Rdom}$ is transitive. If this is not the case, we obtain the assertion by inductive hypothesis for $\mathcal{D}'$. □

The next lemma convinces us that the judgments defined above respect well-formedness, i.e. that the information described by $\Sigma\Lambda$ merely refines the type information.

**Lemma 3.2.10** *Let $\Sigma$ be a valid signature.*

1. *If $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$, then $\flat(\mathbf{A}) =_{\beta\eta} \flat(\mathbf{B})$.*
2. *If $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, then $\tau(\mathbf{A}) = \tau(\mathbb{A})$.*
3. *If $\vdash_{sig} \Sigma$ and $[\forall\Gamma.\mathbf{A}{::}\mathbb{A}] \in \Sigma$, then $\tau(\mathbf{A}) = \tau(\mathbb{A})$.*

**Proof:** We prove the assertions by a simultaneous induction over the structure of the $\Sigma\Lambda$-proofs for the judgments involved. The only interesting cases for the first assertion are *sort:β:top* and *sort:η:top*, where we can read off the assertions from the inference rules.

For the second assertion we consider the cases for the last step in the $\Sigma\Lambda$-derivation $\mathcal{D}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$. If $\mathcal{D}$ ends in *ws:var*, then $\mathbf{A}$ is a variable of some type $\alpha$ and we have $\tau(\mathbf{A}) = \alpha$ by definition. In the cases where $\mathcal{D}$ ends in *ws:td* or *ws:βη* we obtain the assertions by 1. and 3. If $\mathcal{D}$ ends in *ws:app*, then $\mathbf{A} \doteq \mathbf{CD}$, and we have $\Sigma\Lambda$-proofs for $\Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and $\Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})$, where $\mathbb{A} = \mathfrak{r}(\mathbb{C})$. By inductive hypothesis we have $\tau(\mathbf{C}) = \tau(\mathbb{C})$ and $\tau(\mathbf{D}) = \tau(\mathfrak{d}(\mathbb{C}))$, and therefore $\tau(\mathbf{A}) = \tau(\mathfrak{r}(\mathbb{C}))$, since $\mathfrak{r}(\tau(\mathbb{C})) = \tau(\mathfrak{r}(\mathbb{C}))$ and $\tau(\mathfrak{d}(\mathbb{C})) = \mathfrak{d}(\tau(\mathbb{C}))$. Finally, we get the assertion for the remaining case, where $\mathcal{D}$ ends in *ws:abs*, with a similar application of the inductive hypothesis.

The only interesting cases for the third assertion are the inductive rules *sig:const* and *sig:td*. While we obtain the assertion for the former by construction, the well-typedness for the latter relies on the fact that $\mathbf{Rdom}$ is a typed binary relation. □

**General Assumption 3.2.11** Since we have assumed implicit $\alpha$-conversion on term declarations (cf. 3.2.6), it is easy to see that a variant of the $\alpha$-conversion principle holds on judgments. The judgment $\Gamma, [X::\mathbb{B}] \vdash_\Sigma \mathbf{A}::\mathbb{A}$ is provable in $\Sigma\Lambda$, iff the alphabetic variant $\Gamma, [Y::\mathbb{B}] \vdash_\Sigma [Y/X]A::\mathbb{A}$ is. We will use this phenomenon to keep contexts in our $\Sigma\Lambda$-derivations disjoint by consistently renaming all judgments in subderivations, whenever clashes occur. In particular, for *ws:abs* nodes of the form

$$\frac{\Gamma, [X::\mathbb{B}] \vdash_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \vdash_\Sigma (\lambda X_\mathbb{B}.\mathbf{A})::\mathbb{B} \to \mathbb{A}}\ ws{:}abs$$

in a $\Sigma\Lambda$-derivation $\mathcal{A}$ we always assume that $X \notin \mathbf{Dom}(\Gamma)$. Moreover, we use the notation $\Gamma, \Delta$ for $\Gamma \cup \Delta$ with the implicit assumption that $\mathbf{Dom}(\Gamma) \cap \mathbf{Dom}(\Delta) = \emptyset$.

**Remark 3.2.12 (Non-Structural)** Inspection of the inference rules above shows that the inference system $\Sigma\Lambda$ for sort inference is non-structural (cf. 2.4.5), since the succedent of *ws:*$\beta\eta$ is not restricted to any structural category and furthermore, sorted $\beta\eta$-conversion can dramatically change the structure of a formula. Thus it is not obvious how to construct a sort inference algorithm from this inference system. We will later recover some structural properties (cf. 4.1.2) of $\Sigma\Lambda$-derivations and use these for sort computation in theorem (4.3.5).

**Remark 3.2.13** At first sight the restriction of $\mathbb{A}$ **Rdom** $\mathbb{B}$ in *sig:td* appears to be a grave restriction on the expressiveness of term declarations, since it severely restricts the overloading of function constants. In particular, it is impossible to combine declarations like $[+::(\mathbb{N} \to \mathbb{N} \to \mathbb{N})], and[+::(\mathbb{R} \to \mathbb{R} \to \mathbb{R})]$ declaring the addition function to be a function of naturals and of reals in one signature, since one of them would have to be added to the signature with the *sig:td* rule and we cannot have $(\mathbb{N} \to \mathbb{N} \to \mathbb{N})$ **Rdom** $(\mathbb{R} \to \mathbb{R} \to \mathbb{R})$, since $\mathbb{R} \neq \mathbb{N}$. However, on closer inspection it turns out that the declarations should really formalize the fact that the restriction of the addition function to the naturals ranges over the naturals and should therefore be declared as $[+::(\mathbb{R} \to \mathbb{R} \to \mathbb{R})], [\forall [X::\mathbb{N}], [Y::\mathbb{N}]. + XY::\mathbb{N}]$, which is legal in $\Sigma\Lambda$. This way the formula $+$ is of sort $(\mathbb{R} \to \mathbb{R} \to \mathbb{R})$ with domain in the reals whereas the formula $[\lambda X_\mathbb{N} Y_\mathbb{N}. + XY]$ is of sort $(\mathbb{N} \to \mathbb{N} \to \mathbb{N})$ and has domain in the naturals.

The notion of a unique supporting sort corresponds to the intuition in mathematics that functions come with a unique (maximal) domain and have to be distinguished from restrictions to subdomains.

**Notation 3.2.14** To conserve space and increase legibility we abbreviate term declarations of the form $[\forall \Gamma, [X^1::\mathbb{A}^1], \ldots, [X^n::\mathbb{A}^n].\mathbf{A}\overline{X^n}::\mathbb{B}]$ with $[\forall \Gamma.\mathbf{A}|{:}\mathbb{A}^1 \to \cdots \to \mathbb{A}^n \to \mathbb{B}]$, if $X^1, \ldots, X^n \notin \mathbf{Dom}(\Gamma)$. Such a declaration specifies that the denotation of the functional formula $\mathbf{A}$ when restricted to $\mathcal{D}_{\mathbb{A}^1} \times \cdots \times \mathcal{D}_{\mathbb{A}^n}$ has values in $\mathcal{D}_\mathbb{B}$.

With this convention the declaration in remark 3.2.13 would look like $[+::\mathbb{R} \to \mathbb{R} \to \mathbb{R}]$ and $[+|{:}\mathbb{N} \to \mathbb{N} \to \mathbb{N}]$, which is much easier to read.

**Definition 3.2.15** Let $\mathbf{A}$ be a well-sorted formula, then we call a context $\Gamma$ **frugal for A**, iff $\Gamma$ only contains the free variables of $\mathbf{A}$. We also call a judgment $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ or a term declaration $[\forall\Gamma.\mathbf{A}{::}\mathbb{A}]$ **frugal**, iff $\Gamma$ is frugal for $\mathbf{A}$ and we call a signature $\Sigma$ **frugal**, iff all term declarations in $\Sigma$ are frugal.

**Lemma 3.2.16** *If $\Sigma$ is a frugal signature and $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, then there is a $\Sigma\Lambda$-derivation of $\Gamma' \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ such that $\Gamma'$ is frugal for $\mathbf{A}$ and $\Gamma' \subseteq \Gamma$.*

**Proof:** By simple induction over the proof of $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$. □

**Remark 3.2.17** Note that we cannot assume all subderivations to be frugal. While for the empty signature $\Sigma$ the judgment $\emptyset \vdash_\Sigma \lambda X_\mathbb{A} Y_\mathbb{B}.Y{::}\mathbb{A} \to \mathbb{B} \to \mathbb{B}$ has the frugal $\Sigma\Lambda$-proof below, the subderivation for $\lambda Y.Y$ cannot be frugal, since we need the variable declaration $[X{::}\mathbb{A}]$ for the final *ws:abs* step in the following $\Sigma\Lambda$-derivation:

$$\dfrac{\dfrac{\dfrac{}{[X{::}\mathbb{A}],[Y{::}\mathbb{B}] \vdash_\Sigma Y{::}\mathbb{B}}\;ws{:}var}{[X{::}\mathbb{A}] \vdash_\Sigma \lambda Y_\mathbb{B}.Y{::}\mathbb{B} \to \mathbb{B}}\;ws{:}abs}{\emptyset \vdash_\Sigma \lambda X_\mathbb{A} Y_\mathbb{B}.Y{::}\mathbb{A} \to \mathbb{B} \to \mathbb{B}}\;ws{:}abs$$

The previous lemma allows us to drop declarations in variable contexts of judgments in order to make them frugal. Note that it is in general possible to drop declarations from signature, in particular, it is not true, that $\vdash_{sig} \Delta$, whenever $\vdash_{sig} \Sigma$ and $\Delta \subseteq \Sigma$, since, for instance, deleting the only constant declaration for a constant in $c \in \overline{\Sigma}$ prohibits the proofs of well-sortedness needed for the term declarations in which $c$ occurs. The following lemma does just the opposite by allowing to enlarge signatures and variable contexts in judgments. We will often use it in the proofs to follow without explicitly stating it.

**Lemma 3.2.18 (Monotonicity)** *Let $\Delta \subseteq \Sigma$ be valid signatures, and let $\Xi \subseteq \Gamma$ be variable contexts, then we have*

1. *If $\Gamma \vdash_\Delta \mathbf{A}{::}\mathbb{A}$, then $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$.*
2. *If $\Xi \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, then $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$.*

**Proof:** The assertions can be proven by simple inductions on the $\Sigma\Lambda$-derivations involved. □

**Remark 3.2.19** Together with monotonicity our assumption (3.2.11) on disjointness of contexts in $\Sigma\Lambda$-derivations (cf. 3.2.11) allows us to assume extended contexts in $\Sigma\Lambda$-derivations that end in *ws:app*. Thus we can use the following, alternative form of the *ws:app* inference rule

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{B}{::}\mathfrak{d}(\mathbb{A})}{\Gamma \vdash_\Sigma (\mathbf{AB}){::}\mathfrak{r}(\mathbb{A})}\;ws{:}app$$

**Lemma 3.2.20** *Any valid signature is* **subterm-closed**, *that is, each subformula of a well-sorted formula is again well-sorted.*

**Proof:** Let $\mathcal{D}\colon \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ be a $\Sigma\Lambda$-derivation using the alternative *ws:app* rule as defined above (3.2.19), and let $\mathbf{B}$ be a subformula of $\mathbf{A}$, then we show that $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$ holds for some sort $\mathbb{B} \in \mathcal{S}$ by induction over the $\Sigma\Lambda$-derivation of $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$.

*ws:var* In this case $\mathbf{A} = \mathbf{B} \in \mathcal{V}$ and there is nothing to show.

*ws:td* By 3.2.9 and monotonicity 3.2.18.

*ws:app* In this case $\mathbf{A} \doteq \mathbf{CD}$ and $\mathbf{B}$ is a subformula of either $\mathbf{C}$ or $\mathbf{D}$, which are well-sorted, so we directly obtain the assertion by inductive hypothesis.

*ws:abs* Here we have $\mathbf{A} \doteq \lambda X_{\mathfrak{d}(\mathbb{A})}.\mathbf{C}$ with $\Gamma \vdash_\Sigma \mathbf{C}::\mathfrak{r}(\mathbb{A})$, so $\mathbf{B}$ is well-sorted by the inductive hypothesis, since either $\mathbf{B} \doteq X$ or $\mathbf{B}$ is a subformula of $\mathbf{C}$.

*ws:βη* If the $\Sigma\Lambda$-derivation ends in *ws:βη*, then there is a shorter $\Sigma\Lambda$-derivation $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{C}$ for some $\mathbb{C} \in \mathcal{S}$ and $\mathbf{B}$ is well-sorted[11] by inductive hypothesis. □

The property of subterm-closedness is natural in the context of mathematics, since it does not make sense to allow ill-formed subexpressions in well-formed expressions. This situation may, for instance, be different in the field of natural language processing. Non subterm-closed signatures would also cause technical problems, for example, structural induction would not be possible.

**Remark 3.2.21** For a fixed, valid signature $\Sigma$ we can simplify the inference system by dropping the premise $\vdash_{sig} \Sigma$ from the rule *ws:td*, since the proofs in the original system can be obtained from those in the simplified system by copying the validity proofs for $\Sigma$ into the *ws:td*-nodes.

**General Assumption 3.2.22** We assume that all signatures $\Sigma$ we speak about are valid and that for any constant $c_\alpha \in \Omega_\alpha$ there is a constant declaration $[c_\alpha::\mathbb{A}] \in \Sigma$. Note that by 3.2.10 we have $\alpha = \tau(\mathbb{A})$.

**Notation 3.2.23** In order to stress the relation of the set $\Omega$ with $\Sigma$, we often denote $\Omega$ with $\overline{\Sigma}$. Moreover we use the bar operator for the forgetful functor, which indicates the underlying unsorted objects of sorted ones.

## 3.3 Σ-Structures

With the previously defined concept of valid signatures we can now lay the framework of $\Sigma$-structures, which serves as an algebraic basis for our development of $\Sigma\Lambda$.

**Definition 3.3.1 (Sorted Collection)** In analogy to the typed case we define **sorted collections** of sets, functions, and relations by substituting sorts for types in the definitions. But we do not insist that $\mathcal{D}_\mathbb{A}$ and $\mathcal{D}_\mathbb{B}$ be disjoint in a sorted collection $\mathcal{D}_\mathcal{S}$ of sets for distinct sorts $\mathbb{A}$ and $\mathbb{B}$, since it is intended e.g. for well-sorted formulae to have multiple sorts. Note that sorted collections are also typed collections, since sorts have types: Let $\mathcal{D}_\mathcal{S}$ be a sorted collection, then $\overline{\mathcal{D}}_\mathcal{T}$ defined by $\overline{\mathcal{D}}_\alpha := \bigcup_{\tau(\mathbb{A})=\alpha} \mathcal{D}_\mathbb{A}$ is a typed collection. We call $\overline{\mathcal{D}} := \overline{\mathcal{D}}_\mathcal{T}$ the typed collection corresponding to $\mathcal{D} = \mathcal{D}_\mathcal{S}$.

[11] Now we see why we had to require the formula $\mathbf{B}$ to be well-sorted in *sort:η:top*, since otherwise a formula $\mathbf{B} := [\lambda X_\mathbb{A}.c]\mathbf{D}$ would be well-sorted for arbitrary well-typed formulae $\mathbf{D}$, whenever $c$ is a well-sorted constant, and then our system would not be subterm-closed any more.

**Definition 3.3.2 (Pre-Σ-Structures)** Let $\Sigma$ be a valid signature and $\mathcal{D}_\mathcal{S}$ a sorted collection of sets, then we call the triple $(\mathcal{D}_\mathcal{S}, @, \mathcal{I})$ a **pre-Σ-structure**, if

1. $\mathcal{A} := (\overline{\mathcal{D}}_\mathcal{T}, @, \mathcal{I})$ is a **partial pre-$\overline{\Sigma}$-structure** (cf. 2.1.9), where $\overline{\mathcal{D}}_\alpha := \bigcup_{\tau(\mathbb{A})=\alpha} \mathcal{D}_\mathbb{A}$,
2. $\mathcal{D}_\mathbb{A} @ \mathcal{D}_{\mathfrak{d}(\mathbb{A})} \subseteq \mathcal{D}_{\mathfrak{r}(\mathbb{A})}$ and $\mathbf{Dom}(@) \supseteq \mathcal{D}_\mathbb{A} \times \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$ for a functional sort $\mathbb{A} \in \mathcal{S}_{\alpha\to\beta}$.

Now we can adapt the nomenclature from definition 2.1.9 to the sorted case. In particular, the set $\mathcal{D}_\mathbb{A}$ is called the **universe of sort** $\mathbb{A}$. Note that in contrast to the pre-Ω-structures we require pre-Σ-structures to be total on the sorted universes $\mathcal{D}_\mathbb{A}$. For a pre-Σ-structure $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ we call the partial pre-$\overline{\Sigma}$-structure $\overline{\mathcal{A}} = (\overline{\mathcal{D}}, @, \mathcal{I})$ the **corresponding pre-$\overline{\Sigma}$-structure**. We call $\mathcal{A}$ **comprehension-closed** (**functional**), iff $\overline{\mathcal{A}}$ is.

Let $\mathcal{A} = (\mathcal{D}, @^\mathcal{A}, \mathcal{I})$ and $\mathcal{B} = (\mathcal{E}, @^\mathcal{B}, \mathcal{J})$ be pre-Σ-structures, then a $\overline{\Sigma}$-homomorphism $\kappa: \overline{\mathcal{A}} \longrightarrow \overline{\mathcal{B}}$ is called a **Σ-homomorphism**, iff $\kappa(\mathcal{D}_\mathbb{A}) \subseteq \mathcal{E}_\mathbb{A}$ for all sorts $\mathbb{A} \in \mathcal{S}$. $\kappa$ is called **Σ-monomorphism**, if it is injective, and a **Σ-epimorphism**, if it is surjective and moreover $\kappa(\mathcal{D}_\mathbb{A}) = \kappa(\mathcal{E}_\mathbb{A})$ for all $\mathbb{A} \in \mathcal{S}$. We call $\kappa$ a **Σ-isomorphism**, iff it is an injective Σ-epimorphism.

Note that this definition does not take the information given in the term declarations into account, but only concentrates on the sort structure. This is natural for pre-Σ-structures, since the underlying pre-$\overline{\Sigma}$-structures do not assume anything about comprehension-closedness, and therefore do not guarantee denotations for the formulae in term declarations. Hence it is not possible to give a definition that takes the term declarations into account either. This situation will be better in Σ-structures, which we are about to define. But first let us give the standard example for a pre-Σ-structure.

**Example 3.3.3** If $\Sigma$ is a valid signature and $\Gamma$ is a context, then *wsf*$(\Sigma, \Gamma)$ is a pre-Σ-structure, if $\mathbf{A}@\mathbf{B} = (\mathbf{AB})$, since *ws:app* ensures the totality condition 3.3.2.2.

Now we come to the more relevant notion of Σ-structure.

**Definition 3.3.4 (Σ-structure)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure and $\Gamma$ a variable context, then we call a function $\varphi: \mathbf{Dom}(\Gamma) \longrightarrow \mathcal{D}_\mathcal{S}$ a **Γ-assignment into** $\mathcal{A}$, iff $\varphi(X) \in \mathcal{D}_\mathbb{A}$ for every $X \in \mathbf{Dom}(\Gamma)$ with $\Gamma(X) = \mathbb{A}$. We call a functional pre-Σ-structure $\mathcal{A}$ a **Σ-structure**, iff it is comprehension-closed and for all term declarations $[\forall\Gamma.\mathbf{A}::\mathbb{A}] \in \Sigma$ and for all Γ-assignments $\varphi$ into $\mathcal{A}$ we have $\mathcal{I}_\varphi(\mathbf{A}) \in \mathcal{D}_\mathbb{A}$.

If $\mathbf{A}$ is a closed formulae, then $\mathcal{I}_\varphi(\mathbf{A})$ is independent of the Γ-assignment $\varphi$. In these cases we drop the reference from $\mathcal{I}_\varphi(\mathbf{A})$ and simply write $\mathcal{I}(\mathbf{A})$.

**Remark 3.3.5** Note that Γ-assignments need not exist, since the sets $\mathcal{D}_\mathbb{A}$ may be empty in Σ-structures. Thus if $\Gamma(X) = \mathbb{A}$ in a term declaration $[\forall\Gamma.\mathbf{A}::\mathbb{A} \in \Sigma]$, then the condition for Σ-structures is vacuously fulfilled. This is consistent with the intuition that term declarations specify objects of sort $\mathbb{A}$, which are instances of $\mathbf{A}$. Now if $\mathcal{D}_\mathbb{A}$ is empty, then there cannot be any objects to match $\mathbf{A}$ and therefore the term declaration does not contribute any objects for $\mathcal{D}_\mathbb{A}$. Emptiness of sorts is a problem for the soundness of refutation calculi, which we address in subsection 5.2.4.

**Lemma 3.3.6** *Let $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a Σ-structure, then for any Γ-assignment $\varphi$ into $\mathcal{A}$ we have $\mathcal{I}_\varphi(\mathbf{A}) \in \mathcal{D}_\mathbb{A}$. As a consequence $\mathcal{I}_\varphi$ is a Σ-homomorphism.*

**Proof:** We prove the assertion by an induction on the structure of $\mathcal{D}: \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$. For the ground cases we remark that, if $\mathcal{D}$ ends in *ws:var* or *ws:td*, the assertion holds, since $\varphi$ is a $\Gamma$-assignment, and $\mathcal{A}$ is a $\Sigma$-structure (where the assertion holds for term declarations by definition). In the cases where $\mathcal{D}$ ends in *ws:app* or *ws:abs* we obtain the assertion from the inductive hypothesis and the definition of $\mathcal{I}_\varphi$. For *ws:*$\beta\eta$ note that $\Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$ implies $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ (2.3.22). □

Now we state some lemmata, which are direct consequences of their counterparts in $\Lambda$. Here we take advantage of the fact that we have invested some extra work for $\Lambda$ by generalizing all notions to partial functions.

**Lemma 3.3.7** *Sorted $\beta\eta$-conversion is sound, i.e. if $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ is a $\Sigma$-structure and $\Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for any $\Gamma$-assignment $\varphi$.*

**Proof:** This result is an immediate consequence of the unsorted result for $\overline{\Sigma}$-structures (see lemma 2.3.22). □

In particular, we have the same tight correspondence between substitutions and variable assignments as in $\Lambda$.

**Theorem 3.3.8 (Substitution Value Theorem)** *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a $\Sigma$-structure, $\mathbf{A} \in wsf(\Sigma, \Gamma, [X::\mathbb{A}])$, $\mathbf{A} \in wsf(\Sigma, \Gamma)$, and $\varphi$ be a $\Gamma$-assignment, then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})$.*

**Proof:** By 2.3.17. □

**Definition 3.3.9** Let $\mathcal{A}$ be a pre-$\Sigma$-structure, then a congruence $\sim$ on $\overline{\mathcal{A}}$ is called a **$\Sigma$-congruence on** $\mathcal{A}$, iff $f \in \mathcal{D}_\mathbb{B}$ and $g \sim f$ imply $g \in \mathcal{D}_\mathbb{B}$. Here we have adapted the definition of a $\overline{\Sigma}$-congruence (cf. 2.3.1) by requiring a totality condition for domain sorts, that is analogous to that in the definition of $\Sigma$-structures.

A $\Sigma$-congruence $\sim$ is called **functional**, iff for all functional sorts $\mathbb{A} \in \mathcal{S}^f$ and all $f, g \in \mathcal{D}_\mathbb{A}$ the fact that $f@a \sim g@a$ for all $a \in \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$ implies $f \sim g$. Note that, since $\sim$ is a congruence, we also have the other direction, so $f@a \sim g@a$ for all $a \in \mathcal{D}_\beta$, iff $f \sim g$.

**Lemma 3.3.10** *If $\sim$ is a $\Sigma$-congruence on a pre-$\Sigma$-structure $\mathcal{A}$, then $\mathcal{A}/_\sim$ is a pre-$\Sigma$-structure as well, and $\pi_\sim$ is a $\Sigma$-homomorphism. Furthermore, $\mathbf{A}/_\sim$ is comprehension-closed, iff $\mathcal{A}$ is, and functional, if $\sim$ is.*

**Proof:** In the light of 2.3.3 and 3.3.2 it only remains to show that $\mathcal{D}^\sim_\mathbb{A} @^\sim \mathcal{D}^\sim_{\mathfrak{d}(\mathbb{A})} \subseteq \mathcal{D}^\sim_{\mathfrak{r}(\mathbb{A})}$ and that $@^\sim$ is a total function on $\mathcal{D}^\sim_\mathbb{A} \times \mathcal{D}_{f\mathfrak{d}(\mathbb{A})}$. So let $f \in \mathcal{D}_\mathbb{A}$ and $a \in \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$, then $[\![f]\!] \in \mathcal{D}^\sim_\mathbb{A}$ and $[\![a]\!] \in \mathcal{D}^\sim_{\mathfrak{d}(\mathbb{A})}$ and $[\![f]\!] @^\sim [\![a]\!] = [\![f@a]\!] \in \mathcal{D}^\sim_{\mathfrak{r}(\mathbb{A})}$, since $f@a \in \mathcal{D}_{\mathfrak{r}(\mathbb{A})}$. □

**Lemma 3.3.11** *Let $W := \{X^1_{\alpha_1}, \ldots, X^n_{\alpha_n}\}$ and $\Omega := \{c^1_{\beta_1}, \ldots, c^m_{\beta_m}\}$ be typed sets of variables and constants. Furthermore, let*

$$\begin{array}{rcl} W^\sharp & := & \{[X^1::\sharp(\alpha_1)], \ldots, [X^n::\sharp(\alpha_n)]\} \\ \Omega^\sharp & := & \{[c^1::\sharp(\beta_1)], \ldots, [c^m::\sharp(\beta_m)]\} \end{array}$$

*for some context $\Gamma$ with $\mathbf{Dom}(\Gamma) = W$, then*

1. *$\Omega^\sharp$ is a valid signature.*

2. *$\sharp$: $wff(\overline{\Omega}, W) \longrightarrow wsf(\Sigma, \Gamma)$ is a $\Omega$-monomorphism and $\flat$: $wsf(\Sigma, \Gamma) \longrightarrow wff(\overline{\Omega}, W)$ is a $\Omega$-epimorphism.*

3. *For $\mathbf{A} \in wff(\Omega, W)$ we have $\Sigma \vdash \mathbf{A}\colon \alpha$, iff $\Gamma \vdash_{\Omega^\sharp} \sharp(\mathbf{A})::\sharp(\alpha)$.*

4. *For $\mathbf{A} \in wsf(\Sigma, \Gamma)$ we have $\Gamma \vdash_{\Omega^\sharp} \mathbf{A}::\mathbb{A}$, iff $\Omega \vdash \flat(\mathbf{A})\colon \flat(\mathbb{A})$.*

5. *$\flat(\sharp(\mathbf{A})) = \mathbf{A}$ for al $\mathbf{A} \in wff(\Omega, W)$, thus $\flat$ is an inverse for $\sharp$.*

6. *$\sharp$ and $\flat$ are $\Omega$-isomorphisms, if $\Omega^\sharp$ is trivially sorted.*

**Proof:** Immediate from the definitions. □

**Remark 3.3.12** ΣΛ is a generalization of Λ.

**Proof:** If Σ is trivially sorted, then Σ contains exactly one term declaration $[c::\mathbb{B}]$ with $\tau(\mathbb{B}) = \alpha$ for each constant $c_\alpha \in \overline{\Sigma}_\alpha$, since **Rdom** is just the equality relation and we have assumed declarations for all constants in $\overline{\Sigma}$. Thus trivially sorted signatures are isomorphic (by $\tau$) to the signatures of 2.4.4. It is easy to see that sorted $\beta\eta$-conversion is just unsorted $\beta\eta$-conversion, which is sort-preserving even without the rule *ws:βη*, therefore *ws:βη* becomes redundant in the trivially sorted case. Thus the judgment $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ coincides with the judgment $\overline{\Sigma} \vdash \mathbf{A}\colon \tau(\mathbb{A})$, and therefore well-sortedness just reduces to well-typedness, and the functions $\sharp$ and $\flat$ are isomorphisms of pre-$\overline{\Sigma}$-structures. □

## 3.4 Σ-Substitutions

**Definition 3.4.1 (Σ-Substitution)** Let Γ and Δ be variable contexts, then we call a substitution $\sigma$ a **Σ-substitution with domain context Δ and codomain context** Γ, iff the judgment $\Gamma \vdash_\Sigma \sigma::\Delta$ is derivable in the following inference system:

$$\frac{}{\emptyset \vdash_\Sigma \emptyset::\emptyset}\; wsub{:}start$$

$$\frac{\Gamma \vdash_\Sigma \sigma::\Delta \quad \Gamma' \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma||\Gamma' \quad X \notin \mathbf{Dom}(\Gamma) \cup \mathbf{Dom}(\Gamma')}{\Gamma' \cup \Gamma \vdash_\Sigma \sigma, [\mathbf{A}/X]::\Delta, [X::\mathbb{A}]}\; wsub{:}ext$$

The set of Σ-substitutions is denoted by $\mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$. A Σ-substitution $\sigma \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$ is called a **Σ-renaming**, if it is a sort-preserving renaming substitution, that is, if $\sigma(X) = Y$ such that $\Delta(X) = \mathbb{A}$, then $\Gamma(Y) = \mathbb{A}$.

**Lemma 3.4.2** *Let $\Gamma \vdash_\Sigma \sigma::\Delta$, then we have*

1. $\mathbf{Dom}(\Delta) = \mathbf{Dom}(\sigma)$, $\mathbf{Intro}(\sigma) \subseteq \mathbf{Dom}(\Gamma)$, *and* $\mathbf{Dom}(\Delta) \cap \mathbf{Dom}(\Gamma) = \emptyset$, *so in particular, $\sigma$ is idempotent (cf. 2.3.4).*

2. *If* $\sigma = \sigma', [\mathbf{A}/X]$, *then* $\Delta = \Delta', [X :: \mathbb{A}]$, *and there is a context* $\Gamma' \subseteq \Gamma$ *such that* $\Gamma' \vdash_\Sigma \sigma' :: \Delta'$.
3. *If* $\sigma = \overline{[\mathbf{A}^k/X^k]}$ *and* $\Delta(X^i) = \mathbb{A}^i$, *then there are contexts* $\Gamma^i \subseteq \Gamma$ *such that* $\Gamma^i \vdash_\Sigma \mathbf{A}^i :: \mathbb{A}^i$ *for all* $i \leq k$.

**Proof:** Immediate from the definitions. □

We now want to show that Σ-instantiation, i.e. application of Σ-substitutions, preserves sorts.

**Theorem 3.4.3** *If* $\Xi, [X :: \mathbb{B}] \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ *and* $\Gamma \vdash_\Sigma \mathbf{B} :: \mathbb{B}$, *then* $\Xi, \Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{A}$.

**Proof:** To make the inductive hypothesis go through, we show a stronger version of the assertion. We show $\Xi, \Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{A}$ from $\mathcal{D}: \Xi' \vdash_{\Sigma'} \mathbf{A} :: \mathbb{A}$ where $\Xi' \subseteq \Xi$, and $\Sigma' \subseteq \Sigma$ by induction on the structure of $\mathcal{D}$. If $\mathcal{D}$ ends in *ws:td*, then there is a signature $\Sigma'' \subseteq \Sigma'$, some $\Xi'' \subseteq \Xi'$, and a subderivation $\mathcal{D}': \Xi'' \vdash_{\Sigma''} \mathbf{A} :: \mathbb{C}$ of $\mathcal{D}$. So by monotonicity there is a ΣΛ-derivation $\mathcal{D}'': \Xi, \Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{C}$. Now consider the following ΣΛ-derivations

$$\dfrac{\dfrac{\Gamma \vdash_\Sigma \mathbf{B} :: \mathbb{B} \quad \dfrac{\Xi, [X :: \mathbb{A}] \vdash_\Sigma \mathbf{A} :: \mathbb{A}}{\Xi \vdash_\Sigma (\lambda X_{\mathbb{B}}.\mathbf{A}) :: \mathbb{B} \to \mathbb{A}}\, ws{:}abs}{\Xi, \Gamma \vdash_\Sigma (\lambda X_{\mathbb{B}}.\mathbf{A})\mathbf{B} :: \mathbb{A}}\, ws{:}app \qquad \dfrac{\mathcal{D}''}{\Xi\Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{C}} \qquad *}{\Xi\Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{A}}\, ws{:}\beta\eta$$

This completes the proof for the *ws:td*. In the *ws:var* case we see that $\mathbf{A}$ is some variable $Y \in \mathbf{Dom}(\Xi')$ with $\Xi'(Y) = \mathbb{A}$ or $\mathbf{A} = X$. If $\mathbf{A} = Y \in \mathbf{Dom}(\Xi')$, then $\sigma(\mathbf{A}) = Y$ and thus $\Xi, \Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ by *ws:var* and monotonicity 3.2.18. On the other hand $\mathbf{A} = X$, then $[\mathbf{B}/X]\mathbf{A} = \mathbf{B}$ and we have $\Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{A}$ by assumption.

In the *ws:abs* case we have $\Xi' \vdash_{\Sigma'} (\lambda Y_{\mathbb{C}}.\mathbf{D}) :: \mathbb{C} \to \mathbb{D}$ for some sort $\mathbb{D}$, by inversion $\Xi', [Y :: \mathbb{C}] \vdash_{\Sigma'} \mathbf{D} :: \mathbb{D}$ and thus $\Xi, \Gamma, [Y :: \mathbb{C}] \vdash_\Sigma [\mathbf{B}/X]\mathbf{D} :: \mathbb{D}$ by inductive hypothesis. Thus we obtain the assertion by definition of substitution application.

The remaining cases *ws:app* and *ws:βη* can be proven similarly by direct applications of the inductive hypothesis and monotonicity. In particular, we do not need any argument about sorted βη-conversion in the *ws:βη* case, since inversion of that rule gives us a ΣΛ-derivation of $\Xi' \vdash_{\Sigma'} \mathbf{A} :: \mathbb{C}$, to which we can directly apply the inductive hypothesis. □

**Theorem 3.4.4** *If* $\Xi, \Delta \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ *and* $\Gamma \vdash_\Sigma \sigma :: \Delta$, *then* $\Xi, \Gamma \vdash_\Sigma \sigma(\mathbf{A}) :: \mathbb{A}$.

**Proof:** We prove the assertion by induction on $\mathcal{D}: \Gamma \vdash_\Sigma \sigma :: \Delta$ that $\Xi, \Gamma \vdash_\Sigma \sigma(\mathbf{A}) :: \mathbb{A}$. If $\mathcal{D}$ ends in *wsub:start*, then $\sigma$ is the empty substitution and the assertion is trivial. If $\mathcal{D}$ ends in *wsub:ext*, then $\sigma = \sigma', [\mathbf{B}/X]$ and $\Delta(X) = \mathbb{B}$. Furthermore, we have $\Gamma' \vdash_\Sigma \mathbf{B} :: \mathbb{B}$ and $\mathcal{D}: \Gamma'' \vdash_\Sigma \sigma' :: \Delta_{-X}$ for contexts $\Gamma', \Gamma'' \subseteq \Gamma$ with $\Gamma' || \Gamma''$ by by 3.4.2.2. By the previous theorem 3.4.3 we have $\Xi, \Gamma' \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{A}$. Now the inductive hypothesis gives us the assertion, since $\sigma(\mathbf{A}) = \sigma'([\mathbf{B}/X]\mathbf{A})$, as $\sigma$ is a Σ-substitution and consequently idempotent. □

**Corollary 3.4.5** *The following assertions are direct consequences of 3.4.4.*

1. *If* $\sigma \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$*, then* $\sigma(wsf_{\mathbb{A}}(\Sigma, \Xi, \Delta)) \subseteq wsf_{\mathbb{A}}(\Sigma, \Xi, \Gamma)$.
2. *The composition of two Σ-substitutions is again a Σ-substitution when defined.*
3. $\Xi, \Delta \vdash_\Sigma \mathbf{C} =_{\beta\eta} \mathbf{D}$ *and* $\Gamma \vdash_\Sigma \sigma :: \Delta$ *imply* $\Xi, \Gamma \vdash_\Sigma \sigma(\mathbf{C}) =_{\beta\eta} \sigma(\mathbf{D})$.

These considerations can also be used to give new, convenient inference rules, that are admissible in ΣΛ. Thus we will freely use these rules in ΣΛ-derivations, without changing our system. In particular, we only have to consider the original inference system from 3.2.7 in our meta-logical considerations.

**Definition 3.4.6 (Σ-Instantiation)** We define two inference rules that will become convenient in the following.

$$\frac{\Delta, [X :: \mathbb{B}] \vdash_\Sigma \mathbf{A} :: \mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{B} :: \mathbb{B} \quad \Gamma || \Delta \quad X \notin \mathbf{Dom}(\Gamma)}{\Delta \cup \Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A} :: \mathbb{A}} \, ws{:}inst$$

$$\frac{\Delta, \Xi \vdash_\Sigma \mathbf{A} :: \mathbb{A} \quad \Gamma \vdash_\Sigma \sigma :: \Xi \quad \mathbf{Dom}(\Gamma) \cap \mathbf{Dom}(\Delta) = \emptyset}{\Delta, \Gamma \vdash_\Sigma \sigma(\mathbf{A}) :: \mathbb{A}} \, ws{:}subst$$

Note that *ws:inst* is only a special case of *ws:subst*.

**Lemma 3.4.7** *The ws:inst and ws:subst rules are derived rules of* ΣΛ.

**Proof:** By 3.4.4. □

**Remark 3.4.8** On first sight the idea of term declarations, that is to allow the declarations of sort information for schematic formulae, would be more suitably treated by postulating the *ws:inst* inference rule instead of the much more powerful (and problematic) *ws:*$\beta\eta$-rule.

However, in such a system term declarations would to be severely restricted in order to obtain well-sortedness of sorted $\beta$-reduction. Consider, for instance, the following signature:

$$\Sigma := \{[+ :: \mathbb{N} \to \mathbb{N} \to \mathbb{N}], [(\lambda X_{\mathbb{N}}. + XX) :: \mathbb{N} \to \mathbb{E}], [1 :: \mathbb{N}]\}$$

In the *ws:inst*-system the judgment $\Gamma \vdash_\Sigma (\lambda X_{\mathbb{N}}. + XX)1 :: \mathbb{E}$ is derivable, but the judgment $\Gamma \vdash_\Sigma (+11) :: \mathbb{E}$ cannot be derived. Obviously, we can remedy this situation by giving the equivalent (leading to the same well-sorted formulae) declaration $[\forall [X :: \mathbb{N}]. + XX :: \mathbb{E}]$, but in the signature

$$\Sigma := \{[f :: \mathbb{A} \to \mathbb{C}], [a :: \mathbb{A}], [\forall G_{\mathbb{A} \to \mathbb{A}}. f(Ga) :: \mathbb{D}]\}$$

we have $\Gamma \vdash_\Sigma f((\lambda X_{\mathbb{A}}. X)a) :: \mathbb{D}$, but only $\Gamma \vdash_\Sigma fa :: \mathbb{C}$.

**Definition 3.4.9 (Σ-Algebra)** Let Σ be a valid signature, then a **pre-Σ-algebra** $\mathcal{A} = (\mathcal{D}, \mathcal{I})$ is a pre-Σ-structure such that

1. for all $\mathbb{A} \in \mathcal{S}^f$ we have $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{F}(\mathcal{D}_{\mathfrak{d}(\mathbb{A})}; \mathcal{D}_{\mathfrak{r}(\mathbb{A})})$,

2. $f@a = f(a)$.

Like in the unsorted case we call a pre-Σ-algebra **standard**, iff $\mathcal{D}_{\mathbb{A}\to\mathbb{B}} = \mathcal{F}(\mathcal{D}_{\mathbb{A}}; \mathcal{D}_{\mathbb{B}})$.

Note that $\overline{\mathcal{A}} = (\overline{\mathcal{D}}_\alpha, \mathcal{I})$ forms a pre-$\overline{\Sigma}$-algebra, therefore pre-Σ-algebras are functional. We call a pre-Σ-algebra $\mathcal{A}$ a **Σ-algebra**, iff it is a Σ-structure.

**Example 3.4.10 (Rudimentary Calculus)** Suppose we want to use the sorts to model the world of elementary analysis. This means we want to include the sets of real numbers ($\mathbb{R}$), the non-negative real numbers ($\mathbb{P}$), and the set of continuous functions on the reals. We denote the subclass of $k$-differentiable functions in $\mathcal{F}(A; B)$ by $\mathcal{C}^k(A; B)$ and the class of continuous functions with $\mathcal{C}^0(A; B)$.

Let Σ be a higher-order signature with $\mathcal{S} := \{\mathbb{P}, \mathbb{R}, \mathbb{C}, \mathbb{O}\}$ such that $\tau(\mathbb{P}) = \tau(\mathbb{R}) = \iota$, $\tau(\mathbb{C}) = (\iota \to \iota)$, $\mathfrak{r}(\mathbb{C}) = \mathfrak{d}(\mathbb{C}) = \mathbb{R}$. Suppose we additionally want to model a positive constant $\pi$, the identity function $i$, and absolute value functions $a$, and the differentiation operator $d$. With the signature

$$\Sigma = \{[\pi{::}\mathbb{P}], [a{::}(\mathbb{R} \to \mathbb{P})], [i{::}\mathbb{C}], [d{::}(\mathbb{C} \to \mathbb{R} \to \mathbb{R})], [\forall [X{::}\mathbb{P}].X{::}\mathbb{R}]\}$$

we have $\vdash_\Sigma a{::}\mathbb{R} \to \mathbb{P}$, $\vdash_\Sigma a{::}\mathbb{R} \to \mathbb{R}$, and $[F{::}\mathbb{C}] \vdash_\Sigma dFa{::}\mathbb{R}$.

The definitions $\mathcal{D}_{\mathbb{R}} := \mathrm{I\!R}$ and $\mathcal{D}_{\mathbb{C}} := \mathcal{C}^1(\mathrm{I\!R}; \mathrm{I\!R})$, together with the convention that the interpretation of $\pi$, $i$, $a$, and $d$ are the number $\pi$ (3.1415...), the identity function, absolute value function and $\mathcal{I}(d)$ the differentiation operator respectively specify a standard pre-Σ-algebra, whereas the setting $\mathcal{D}_{\mathbb{R}\to\mathbb{R}} := \mathcal{C}^0(\mathrm{I\!R}; \mathrm{I\!R})$ defines a class of non-standard pre-Σ-algebras.

## 3.5 Sorted Reduction

It is important to the program of this thesis that the fundamental operations of the calculus do not allow the formation of ill-sorted terms from well-sorted ones. This will ensure that our calculus never has to handle ill-sorted terms, even intermediately. We have seen in 3.4.4 that Σ-instantiation conserves sorts. In this subsection we will use this to show that, if $\Gamma \vdash_\Sigma \mathbf{A} {=_{\beta\eta}} \mathbf{B}$, then $\mathcal{S}^\Gamma_\Sigma(\mathbf{A}) = \mathcal{S}^\Gamma_\Sigma(\mathbf{B})$. This fact implies that sorted reduction is strongly normalizing and all results carry over from the typed case. For the confluence result we need that well-sorted formulae of functional type have unique domain sorts. We will show that the restrictions imposed on the validity of signatures indeed guarantee that all functional well-sorted formulae have unique supporting sorts, which exactly capture the intuition of mathematical practice, where functions have unique domains associated with them. This fact a posteriori justifies our definition of sorted $\eta$-conversion, which is a weak form of the extensionality principle, and therefore relies on the existence of unique domains for functions.

**Lemma 3.5.1** *If* $\Gamma \vdash_\Sigma \mathbf{A} {=_{\beta\eta}} \mathbf{B}$*, then* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$*, iff* $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}$*.*

**Proof:** We only have to show one direction of the equivalence, since $=_{\beta\eta}$ is symmetric. To make the induction go through, we have to show a slightly stronger result: if $\mathcal{C}{:}\, \Gamma \vdash_\Sigma \mathbf{A} {=_{\beta\eta}} \mathbf{B}$ and $\mathcal{A}{:}\, \Gamma \vdash_{\Sigma'} \mathbf{A}{::}\mathbb{A}$ with $\Sigma' \subseteq \Sigma$, then $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}$. We prove the assertion by induction on $(\mathcal{C}, \mathcal{A})$ with respect to the strict lexicographic ordering $\prec$ on pairs of ΣΛ-derivations.

We first convince ourselves that it is sufficient for our purposes to show that $\mathbf{B}$ is well-sorted ($\mathcal{A}'{:}\,\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{C}$ for some sort $\mathbb{C}$), since afterwards we can use the *ws:$\beta\eta$* rule to construct a $\Sigma\Lambda$-derivation that verifies the assertion: with monotonicity we obtain a $\Sigma\Lambda$-derivation $\mathcal{B}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ from $\mathcal{A}$, and therefore

$$\dfrac{\dfrac{\mathcal{B}}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}} \quad \dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{C}} \quad \Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}}\ \textit{ws:}\beta\eta$$

We now proceed to analyze the cases for the $\Sigma\Lambda$-derivation $\mathcal{A}$. If $\mathcal{A}$ ends in an application of the *ws:$\beta\eta$* rule, then we have the following situation:

$$\dfrac{\dfrac{\mathcal{E}}{\Gamma \vdash_{\Sigma'} \mathbf{P}{::}\mathbb{A}} \quad \dfrac{\mathcal{A}'}{\Gamma \vdash_{\Sigma'} \mathbf{A}{::}\mathbb{C}} \quad \dfrac{\mathcal{C}'}{\Gamma \vdash_{\Sigma'} \mathbf{P}{=}_{\beta\eta}\mathbf{A}}}{\Gamma \vdash_{\Sigma'} \mathbf{A}{::}\mathbb{A}}\ \textit{ws:}\beta\eta$$

Since $\mathcal{A}'$ is a proper subderivation of $\mathcal{A}$, we have $(\mathcal{C},\mathcal{A}') \prec (\mathcal{C},\mathcal{A})$ and therefore by inductive hypothesis for $\mathcal{A}'{:}\,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{C}$ and $\mathcal{C}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ we have $\Gamma \vdash_{\Sigma'} \mathbf{B}{::}\mathbb{C}$, which yields the assertion.

If $\mathcal{A}$ ends in *ws:td*, then we have $[\forall\Delta.\mathbf{A}{::}\mathbb{A}] \in \Sigma'$ for some context $\Delta \subseteq \Gamma$, and there is a signature $\Sigma'' \subseteq \Sigma$ such that $\mathcal{A}'{:}\,\Delta \vdash_{\Sigma''} \mathbf{A}{::}\mathbb{C}$ and $\mathbb{C}\ \mathbf{Rdom}\ \mathbb{A}$. Since $\mathcal{A}'$ is a proper subderivation of $\mathcal{A}$, we have $(\mathcal{C},\mathcal{A}') \prec (\mathcal{C},\mathcal{A})$, and therefore by inductive hypothesis $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{C}$.

Now that we have treated the *ws:$\beta\eta$* and *ws:td* cases for $\mathcal{A}$ we proceed by analyzing the $\Sigma\Lambda$-derivation $\mathcal{C}$. We first treat the four base cases separately in which $\mathcal{C}$ ends in some top-level $\beta\eta$-conversion:

$\beta$ Here $\mathbf{A} \doteq (\lambda X_{\mathbb{B}}.\mathbf{M})\mathbf{N}$ and $\mathbf{B} \doteq [\mathbf{N}/X]\mathbf{M}$, and we have $\Gamma \vdash_\Sigma \mathbf{N}{::}\mathbb{B}$ and $\Gamma,[X{::}\mathbb{B}] \vdash_\Sigma \mathbf{M}{::}\mathbb{C}$ for some sort $\mathbb{C}$ by inversion of *sort:$\beta$:top*, so by *ws:inst* we have $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{C}$, which completes this case in light of the remarks above.

$\beta^{-1}$ Here $\mathbf{B} \doteq (\lambda X_{\mathbb{B}}.\mathbf{M})\mathbf{N}$ and $\mathbf{A} \doteq [\mathbf{N}/X]\mathbf{M}$, and just as above we have $\Gamma \vdash_\Sigma \mathbf{N}{::}\mathbb{B}$ and $\Gamma,[X{::}\mathbb{B}] \vdash_\Sigma \mathbf{M}{::}\mathbb{C}$ for some sort $\mathbb{C}$, and therefore

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{N}{::}\mathbb{B} \qquad \dfrac{\Gamma,[X{::}\mathbb{B}] \vdash_\Sigma \mathbf{M}{::}\mathbb{C}}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{B}}.\mathbf{M}{::}\mathbb{B} \to \mathbb{C}}\ \textit{ws:abs}}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{C}}\ \textit{ws:app}$$

$\eta$ Here $\mathbf{A} \doteq (\lambda X.\mathbf{B}X)$, so by inversion we have $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{C}$.

$\eta^{-1}$ Here $\mathbf{B} \doteq (\lambda X.\mathbf{A}X)$, so by monotonicity we have

$$\dfrac{\dfrac{\Gamma,[X{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathbf{A}{::}\mathbb{A} \qquad \Gamma,[X{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma X{::}\mathfrak{d}(\mathbb{A})}{\Gamma,[X{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathbf{A}X{::}\mathfrak{r}(\mathbb{A})}\ \textit{ws:app}}{\Gamma \vdash_\Sigma \lambda X_{\mathfrak{d}(\mathbb{A})}.\mathbf{A}X{::}\mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})}\ \textit{ws:abs}$$

which yields the assertion of the theorem. Clearly we obtain the stronger assertion of the induction with monotonicity.

If $\mathcal{C}$ ends in *tr:app:fn*, then $\mathbf{A} \doteq \mathbf{CD}$ and $\mathbf{B} \doteq \mathbf{C'D}$ where $\Gamma \vdash_\Sigma \mathbf{C} =_{\beta\eta} \mathbf{C'}$. So the only interesting case for $\mathcal{A}$ is *ws:app*, since *ws:abs* and *ws:var* cannot apply, and we have treated *ws:$\beta\eta$* and *ws:td*. In this case we have $\mathbb{A} = \mathfrak{r}(\mathbb{C})$, ΣΛ-derivations $\Gamma \vdash_\Sigma \mathbf{C}::\mathbb{C}$, and $\Gamma \vdash_\Sigma \mathbf{D}::\mathfrak{d}(\mathbb{C})$ and by inductive hypothesis $\Gamma \vdash_\Sigma \mathbf{C'}::\mathbb{C}$. Therefore we obtain $\Gamma \vdash_\Sigma \mathbf{B}::\mathfrak{r}(\mathbb{C})$ by a single application of *ws:app*. It is easily seen that the cases *tr:app:arg* and *tr:abs* can be treated with similar arguments.

If $\mathcal{C}$ ends in *ms:ref*, then we have $\mathbf{A} \doteq \mathbf{B}$, and the assertion is trivial. So in order to complete the proof we only have to treat the case, where $\mathcal{C}$ ends in an application of the *ms:trans* rule. Here we have subderivations $\mathcal{C}' : \Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{C}$ and $\mathcal{C}'' : \Gamma \vdash_\Sigma \mathbf{C} =_{\beta\eta} \mathbf{B}$ of $\mathcal{C}$, therefore by inductive hypothesis we have a ΣΛ-derivation $\mathcal{A}' : \Gamma \vdash_\Sigma \mathbf{C}::\mathbb{A}$ and again by inductive hypothesis $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}$. Note that, since we have chosen $\prec$ to be the lexicographic ordering, the second application of the inductive hypothesis is independent of the size of $\mathcal{A}'$. □

**Theorem 3.5.2** *Sorted $\beta\eta$-reduction on wsf$(\Sigma, \Gamma)$ is terminating.*

**Proof:** This result is a direct consequence of lemma 3.2.10: any sequence of sorted $\beta\eta$-reductions is also a sequence of (unsorted) $\beta\eta$-reductions and those always terminate. □

We now prove that valid signatures respect function domains in the sense that for every formula $\mathbf{A}$ of functional sort and any sorts $\mathbb{A}, \mathbb{B}$ of $\mathbf{A}$, we must have $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$, and therefore $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$, which we call the **supporting sort** of $\mathbf{A}$.

**Definition 3.5.3 (Depth)** We define the **depth** $\mathbf{dp}(\mathbf{A})$ of a formula $\mathbf{A}$ inductively on the structure of $\mathbf{A}$ by setting the depth of variables and constants to 0 and the depths of applications and λ-abstractions to the maximum of depths of the immediate subformulae incremented by 1. This definition is only a special case of the general definition of depth for trees, setting the depth of leaves to 0. It gives us a notion of depth for ΣΛ-derivations, that we use in the following.

**Theorem 3.5.4** *If $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{B}$, then $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$.*

**Proof:** Let $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{E}$ be ΣΛ-derivations and $\mu(\mathcal{A}, \mathcal{B}) := (\mu_1(\mathcal{A}, \mathcal{B}), \mu_2(\mathcal{A}, \mathcal{B}), \mathcal{E})$, where $\mu_1(\mathcal{A}, \mathcal{B}) := \max(\mathbf{dp}(\mathcal{A}), \mathbf{dp}(\mathcal{B}))$ and $\mu_2(\mathcal{A}, \mathcal{B}) := (\mathbf{dp}(\mathcal{A}), \mathbf{dp}(\mathcal{B}))$. Furthermore let $\prec$ be the strict lexicographic ordering on triples for the component orderings $<$ on natural numbers, the multiset ordering on pairs of natural numbers, and the structural ordering on ΣΛ-derivations.

Let $\mathcal{A} : \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, $\mathcal{B} : \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$, and $\mathcal{E} : \Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$. To make the induction go through we prove the stronger assertion that in this case $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$ by induction on $\mu(\mathcal{A}, \mathcal{B}, \mathcal{C})$ with respect to $\prec$.

If $\mathcal{A}$ ends in *ws:$\beta\eta$*, then we have the following situation:

$$\frac{\dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{C}::\mathbb{A}} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{C}} \quad \Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\ ws{:}\beta\eta$$

If $\mathbf{dp}(\mathcal{A}) \leq \mathbf{dp}(\mathcal{B})$, then $\mu_1(\mathcal{A}', \mathcal{A}'') < \mathbf{dp}(\mathcal{A}) \leq \mu_1(\mathcal{A}, \mathcal{B})$, so $\mu(\mathcal{A}', \mathcal{A}'') \prec \mu(\mathcal{A}, \mathcal{B})$, and thus by inductive hypothesis $\mathbb{A}$ **Rdom** $\mathbb{C}$. Furthermore we have $\mu_1(\mathcal{A}'', \mathcal{B}) = \mathbf{dp}(\mathcal{B}) = \mu_1(\mathcal{A}, \mathcal{B})$, but $\mu_2(\mathcal{A}'', \mathcal{B}) < \mu_2(\mathcal{A}, \mathcal{B})$, so $\mu(\mathcal{A}'', \mathcal{B}) \prec \mu(\mathcal{A}, \mathcal{B})$, and thus by inductive hypothesis $\mathbb{C}$ **Rdom** $\mathbb{B}$. If on the other hand, $\mathbf{dp}(\mathcal{B}) < \mathbf{dp}(\mathcal{A})$, then $\mu_1(\mathcal{A}', \mathcal{A}'') < \mathbf{dp}(\mathcal{A}) = \mu_1(\mathcal{A}, \mathcal{B})$, so by inductive hypothesis $\mathbb{A}$ **Rdom** $\mathbb{B}$. Furthermore $\mu_1(\mathcal{A}'', \mathcal{B}) < \mu_1(\mathcal{A}, \mathcal{B})$, therefore $\mathbb{C}$ **Rdom** $\mathbb{B}$. This completes the case where $\mathcal{A}$ ends in *ws:βη*, since **Rdom** is transitive. Note that for this argument we have only used the fact that $\mathcal{A}'$ and $\mathcal{A}''$ are subderivations of $\mathcal{A}$. This makes this argument widely applicable in this proof.

If $\mathcal{A}$ ends in *ws:td*, then there is a term declaration $[\forall\Delta.\mathbf{A}::\mathbb{A}] \in \Sigma$ for some variable context $\Gamma' \subseteq \Gamma$, a signature $\Sigma' \subseteq \Sigma$, and a ΣΛ-subderivation $\mathcal{A}' : \Gamma' \vdash_{\Sigma'} \mathbf{A}::\mathbb{C}$ of $\mathcal{A}$, for some sort $\mathbb{C}$ **Rdom** $\mathbb{A}$. Now $\mathbf{dp}(\mathcal{A}') < \mathbf{dp}(\mathcal{A})$ implies $\mu_1(\mathcal{A}', \mathcal{B}) \leq \mu_1(\mathcal{A}, \mathcal{B})$ and $\mu_2(\mathcal{A}', \mathcal{B}) < \mu_2(\mathcal{A}, \mathcal{B})$, so by inductive hypothesis we have $\mathbb{C}$ **Rdom** $\mathbb{C}$ and thus $\mathbb{A}$ **Rdom** $\mathbb{B}$ by transitivity of **Rdom** . Clearly the cases where $\mathcal{B}$ ends in *ws:βη* or *ws:td* are analogous, since sorted $\beta\eta$-equality is symmetric.

Now we proceed by analyzing the cases for $\mathcal{E}$. If $\mathcal{E}$ ends in *ms:ref*, then then we have $\mathbf{A} \doteq \mathbf{B}$. So if $\mathcal{A}$ ends in *ws:var*, then we can assume that $\mathcal{B}$ does too, since *ws:app* and *ws:abs* do not apply and we have already treated *ws:βη* and *ws:td*. In the remaining case we have $\mathbb{A} = \mathbb{B}$, which clearly entails $\mathbb{A}$ **Rdom** $\mathbb{B}$. If $\mathcal{A}$ and $\mathcal{B}$ both end in *ws:app*, then $\mathbf{A} \doteq \mathbf{CD} \doteq \mathbf{B}$, and there are subderivations $\mathcal{A}' : \mathbf{C}::\mathbb{C}$ and $\mathcal{B}' : \mathbf{C}::\mathbb{D}$ of $\mathcal{A}$ and $\mathcal{B}$ that satisfy the inductive hypothesis, so we have $\mathbb{C}$ **Rdom** $\mathbb{D}$ and thus $\mathbb{A} = \mathfrak{r}(\mathbb{C})$ **Rdom** $\mathfrak{r}(\mathbb{D}) = \mathbb{B}$. Finally, if $\mathcal{A}$ and $\mathcal{B}$ both end in *ws:abs*, then $\mathbf{A} = \lambda X_{\mathbb{C}}.\mathbf{D} = \mathbf{B}$ and there are subderivations $\mathcal{A}' : \mathbf{D}::\mathbb{D}$ and $\mathcal{B}' : \mathbf{D}::\mathbb{E}$, so by inductive hypothesis $\mathbb{D}$ **Rdom** $\mathbb{E}$, and thus $\mathbb{A} = \mathbb{C} \to \mathbb{D}$ **Rdom** $\mathbb{C} \to \mathbb{E} = \mathbb{B}$.

If $\mathcal{E}$ ends in *sort:β:top*, then $\mathbf{A} \doteq (\lambda X_{\mathbb{C}}.\mathbf{C})\mathbf{D}$ and $\mathbf{B} \doteq [\mathbf{D}/X]\mathbf{C}$. Since we have treated the cases *ws:βη* and *ws:td*, we can assume that $\mathcal{A}$ ends in *ws:app* and we have the following situation

$$\dfrac{\dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{C}}.\mathbf{C}::\mathbb{E}} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{D}::\mathfrak{d}(\mathbb{E})}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{C}}.\mathbf{C})\mathbf{D}::\mathfrak{r}(\mathbb{E}) = \mathbb{A}}\ ws{:}app$$

with an argument exactly like the above we treat the cases, where $\mathcal{A}'$ ends in *ws:βη* or *ws:td*, so we can assume that $\mathcal{A}'$ ends in *ws:abs* and we have the following situation:

$$\dfrac{\dfrac{\dfrac{\mathcal{C}}{\Gamma, [X::\mathbb{C}] \vdash_\Sigma \mathbf{C}::\mathbb{A}}}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{C}}.\mathbf{C}::\mathbb{C} \to \mathbb{A}}\ ws{:}abs \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{D}::\mathbb{C}}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{C}}.\mathbf{C})\mathbf{D}::\mathbb{A}}\ ws{:}app$$

If $\mathbf{C} \doteq X$, then $\mathbf{B} \doteq \mathbf{D}$ and $\mathbb{C} = \mathbb{A}$, thus we obtain $\mathbb{A}$ **Rdom** $\mathbb{B}$ by inductive hypothesis for $\mathcal{A}''$, $\mathcal{B}$, and $\mathcal{E}$. With this argument we can for the rest of this case assume that $\mathbf{C} \not\doteq X$. If $\mathcal{B}$ ends in *ws:var*, then $\mathbf{B} \doteq \mathbf{C} \in \mathbf{Dom}(\Gamma)$, since $\mathbf{C} \not\doteq X$. Thus we obtain the assertion by inductive hypothesis for $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{E}$. If $\mathcal{B}$ ends in *ws:app*, then $\mathbf{C} \doteq \mathbf{EF}$, $\mathbf{B} = [\mathbf{D}/X](\mathbf{EF})$, and $\mathcal{B}$ is of the form

$$\dfrac{\dfrac{\mathcal{B}'}{\Gamma \vdash_\Sigma [\mathbf{D}/X]\mathbf{E}{::}\mathbb{E}} \quad \Gamma \vdash_\Sigma [\mathbf{D}/X]\mathbf{F}{::}\mathfrak{d}(\mathbb{E})}{\Gamma \vdash_\Sigma ([\mathbf{D}/X]\mathbf{E})([\mathbf{D}/X]\mathbf{F}){::}\mathfrak{r}(\mathbb{E}) = \mathbb{B}}\ \textit{ws:app}$$

by an argument as above $\mathcal{C}$ has the form

$$\dfrac{\dfrac{\mathcal{C}'}{\Gamma \vdash_\Sigma \mathbf{E}{::}\mathbb{F}} \quad \Gamma, [X{::}\mathbb{A}] \vdash_\Sigma \mathbf{F}{::}\mathfrak{d}(\mathbb{F})}{\Gamma, [X{::}\mathbb{A}] \vdash_\Sigma \mathbf{EF}{::}\mathfrak{r}(\mathbb{F}) = \mathbb{A}}\ \textit{ws:app}$$

so we can construct a ΣΛ-derivation $\mathcal{C}''$

$$\dfrac{\dfrac{\dfrac{\mathcal{C}'}{\Gamma, [X{::}\mathbb{A}] \vdash_\Sigma \mathbf{E}{::}\mathbb{F}}}{\Gamma, [X{::}\mathbb{A}] \vdash_\Sigma (\lambda X_{\mathbb{C}}.\mathbf{E}){::}\mathbb{C} \to \mathbb{F}}\ \textit{ws:abs} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{D}{::}\mathbb{C}}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{C}}.\mathbf{E})\mathbf{D}{::}\mathbb{F}}\ \textit{ws:app}$$

Clearly we have $\mathbf{dp}(\mathcal{C}'') \leq \mathbf{dp}(\mathcal{A})$, so $\mu(\mathcal{C}'', \mathcal{B}') < \mu(\mathcal{A}, \mathcal{B})$, and therefore the inductive hypothesis guarantees that $\mathbb{E}$ **Rdom** $\mathbb{F}$, so $\mathbb{A} = \mathfrak{r}(\mathbb{F})$ **Rdom** $\mathfrak{r}(\mathbb{E}) = \mathbb{B}$. The case where $\mathcal{B}$ ends in *ws:abs* can be treated with analogous methods and completes the analysis for the case where $\mathcal{E}$ ends in *sort:β:top*. If $\mathcal{E}$ ends in *sort:η:top*, then $\mathbf{A} \doteq (\lambda X_{\mathbb{C}}.\mathbf{B}X)$ and we can assume that we have the following form for $\mathcal{A}$:

$$\dfrac{\dfrac{\dfrac{\mathcal{C}}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}} \quad \dfrac{}{[X{::}\mathbb{C}] \vdash_\Sigma X{::}\mathbb{C}}\ \textit{ws:var}}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{C}}.\mathbf{B}{::}\mathbb{C} \to \mathbb{A}}\ \textit{ws:app}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{C}}.\mathbf{B}X){::}\mathbb{A}}\ \textit{ws:abs}$$

thus we directly obtain the assertion by the inductive hypothesis for $\mathcal{C}$, $\mathcal{B}$, and the the ΣΛ-derivation consisting only of a single *ms:ref* step. Now it only remains to check the inductive cases for $\mathcal{E}$. If $\mathcal{E}$ ends in *tr:app:fn* or *tr:app:arg*, then $\mathbf{A}$ and $\mathbf{B}$ must be applications, so we can assume that $\mathcal{A}$ and $\mathcal{B}$ end in *ws:app*. Thus we have ΣΛ-subderivations $\mathcal{A}'$, $\mathcal{B}'$, and $\mathcal{E}'$ that meet the assumptions of the inductive hypothesis, which then yields $\mathbb{A}$ **Rdom** $\mathbb{B}$. It can easily be seen that the cases *tr:abs*, *ms:trans*, and *eq:sym*, can be handled with related methods. Thus we have finally completed our analysis of all possible cases for $\mathcal{E}$ and thus proven the assertion. □

**Corollary 3.5.5** *If* $\Gamma \vdash_\Sigma (\lambda X_{\mathbb{B}}.\mathbf{A}){::}\mathbb{A}$, *then* $\mathbb{B} = \mathfrak{d}(\mathbb{A})$.

**Proof:** By lemma 3.2.20 there is a sort $\mathbb{C}$ such that $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{C}$, so by *ws:abs* we have $\Gamma \vdash_\Sigma (\lambda X_{\mathbb{B}}.\mathbf{A}){::}\mathbb{B} \to \mathbb{C}$, so we have $\mathbb{A}$ **Rdom** $\mathbb{B} \to \mathbb{C}$, and therefore $\mathbb{B} = \mathfrak{d}(\mathbb{A})$. □

**Remark 3.5.6** The previous corollary allows us to infer the sorts of bound variables in well-sorted formulae. This enables us to simplify the notation by dropping the sort in λ-abstractions, if we specify the sort. For example, if $\mathbf{B} \doteq (\lambda \overline{X^k{::}\mathbb{B}^k}.\mathbf{A})$ and $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}$, then $\mathbb{B}^i = \mathfrak{d}^i(\mathbb{A})$. Therefore we often write $\mathbf{B}$ as $\lambda \overline{X^k}.\mathbf{A}$.

**Definition 3.5.7** Let $\mathbf{A}$ be a well-sorted formula, then we call the unique sort $\mathbb{B}^i$ such that $\mathbb{B}^i = \mathfrak{d}^i(\mathbb{A})$ for all sorts $\mathbb{A}$ with $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ the $i^{th}$ **argument sort** of $\mathbf{A}$ ($\mathbf{supp}^i(\mathbf{A})$). We also call the first argument sort of $\mathbf{A}$ the **supporting sort of A** and denote it by $\mathbf{supp}(\mathbf{A})$.

**Remark 3.5.8** The set of argument sorts is effectively computable. If $\mathbf{A}$ is an abstraction of the form $\lambda X_{\mathbb{A}}.\mathbf{B}$, then $\mathbf{supp}(\mathbf{A}) = \mathbb{A}$ and $\mathbf{supp}^{i+1}(\mathbf{A}) = \mathbf{supp}^i(\mathbf{B})$, if $\mathbf{A} \doteq \mathbf{BC}$, then $\mathbf{supp}^i(\mathbf{A}) = \mathbf{supp}^{i+1}(\mathbf{B})$. The supporting sorts of constants and variables can be read off their declarations.

**Remark 3.5.9** The theorems above are the reason for requiring the **Rdom** proviso for any term declaration to be added to a signature by the rule *sig:td*. We do not need such a proviso for the *sig:const* rule, since for new constants there is no sort information present that would have to be respected.

Most mathematicians would agree that functional extensionality relies heavily on the notion of explicitly specified domains of functions. Unique supporting sorts are intended to syntactically capture this intuition in ΣΛ. Indeed in mathematics, functions are assumed to have unique (explicitly specified) domains, and must therefore be distinguished from restrictions to subdomains. For example, the addition function on the reals must be distinguished from the addition function on the natural numbers, and in general functions $f$ and $g$ should only be considered the same, if $fa = ga$ for all $a$ in the common (explicitly specified) domain of $f$ and $g$. Observing these distinctions is necessary for a correct treatment of extensional higher-order calculi, and they must be reflected in the syntax of any such calculus. This fact is taken into account in our definition of Σ-structures by requiring that all functions in $\mathcal{D}_{\mathbb{A}}$ are total on $\mathcal{D}_{\mathfrak{d}(\mathbb{A})}$.

**Theorem 3.5.10** *Sorted βη-reduction is confluent.*

**Proof sketch:** Since sorted $\beta\eta$-reduction is terminating, we only have to show that it is **weakly confluent**, i.e. if $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}$ and $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{C}$, then there is a formula $\mathbf{D}$ such that $\Gamma \vdash_\Sigma \mathbf{B} \rightarrow^*_{\beta\eta} \mathbf{D}$ and $\Gamma \vdash_\Sigma \mathbf{C} \rightarrow^*_{\beta\eta} \mathbf{D}$. This can be shown in three steps. The first two steps show that sorted $\beta$-conversion is weakly confluent, and that $\beta-$ and $\eta$-reduction commute, i.e. if $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_\beta \mathbf{B} \rightarrow_\eta \mathbf{C}$, then there is a formula $\mathbf{D}$ such that $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_\eta \mathbf{D} \rightarrow_\beta \mathbf{C}$ and vice versa. This can be done with standard arguments from [Bar80, HS86], since the sort conditions in *sort:β:top* only concern well-sortedness and $\beta\eta$-reduction conserves sorts by 3.5.1. Thus for any $\beta\eta$-reduction of the form $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_{\beta\eta} \mathbf{B}$, there is a $\beta\eta$-reduction $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^*_\beta \mathbf{C} \rightarrow^*_\eta \mathbf{B}$. Now it only remains to be shown that sorted $\eta$-reduction is weakly confluent.

Let $\mathbf{A} := \mathbf{C}[\mathbf{B}]$ denote a formula $\mathbf{A}$, with one occurrence of a subformula $\mathbf{B}$ in the context $\mathbf{C}$. Then we have three cases for $\eta$-reduction:

1. $\Gamma \vdash_\Sigma \mathbf{C}[\lambda X_{\mathbb{A}}.\mathbf{P}X] \rightarrow_\beta \mathbf{C}[\mathbf{P}]$,

2. $\Gamma \vdash_\Sigma \mathbf{C}[\lambda X_{\mathbb{A}}.\mathbf{P}X] \to_\beta \mathbf{C}[\lambda X_{\mathbb{A}}.\mathbf{P}'X]$,

3. $\Gamma \vdash_\Sigma \mathbf{C}[\lambda X_{\mathbb{A}}.\mathbf{P}X] \to_\beta \mathbf{C}'[\lambda X_{\mathbb{A}}.\mathbf{P}X]$,

where $\Gamma \vdash_\Sigma \mathbf{C} \to_\eta^* \mathbf{C}'$ and $\Gamma \vdash_\Sigma \mathbf{P} \to_\eta \mathbf{P}'$. By inversion we have $\Gamma \vdash_\Sigma \mathbf{P}::\mathbb{B}$ for some $\mathbb{B}$ with $\mathfrak{d}(\mathbb{B}) = \mathbb{A}$, and by 3.5.1 we have $\Gamma \vdash_\Sigma \mathbf{P}'::\mathbb{B}$. Thus we can $\eta$-reduce $\lambda X_{\mathbb{A}}.\mathbf{P}'X$ to $\mathbf{P}'$, and join 1. and 2. by $\mathbf{C}[\mathbf{P}']$. Moreover we can join 2. and 3. by $\mathbf{C}'[\mathbf{P}]$. □

**Remark 3.5.11** In light of the previous theorem it makes sense to speak of *the* **sorted $\beta$-normal form** and *the* **sorted (long) $\beta\eta$-normal form**. Since these normal forms are also unsorted normal forms all results and definitions from the typed case (cf. 2.3) carry over to ΣΛ. Furthermore, lemma 3.5.1 implies that both notions of sorted normalization conserve the sets of sorts of formulae.

Now we use the results of this section to define suitable notions of term algebras.

**Definition 3.5.12 (Term Σ-Structure)** Let $\mathcal{D}_{\mathbb{A}}$ be the set of well-sorted formulae $\mathbf{A}$ in sorted $\beta\eta$-normal form such that $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$. Furthermore, let $\mathbf{A}@\mathbf{B}$ be the sorted $\beta\eta$-normal form of $(\mathbf{AB})$ and $\mathcal{I} := \mathrm{Id}_{\overline{\Sigma}}$, then we call $\mathcal{TS}(\Sigma,\Gamma) := (\mathcal{D}_{\mathcal{S}}, @, \mathcal{I})$ the **term structure for Σ and Γ**.

This definition is justified by the following lemma.

**Lemma 3.5.13** *$\mathcal{TS}(\Sigma,\Gamma)$ is a Σ-structure with $\mathcal{I}_\varphi(\mathbf{A}) = \varphi(\mathbf{A})\!\downarrow$.*

**Proof:** By 2.3.14 $\overline{\mathcal{TS}(\Sigma,\Gamma)} = \mathcal{TS}(\Omega)$ is a pre-$\overline{\Sigma}$-structure. Let Δ be a finite variable context such that $\mathbf{Dom}(\Delta) \cap \mathbf{Dom}(\Gamma) = \emptyset$ and $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\varphi$ be a Δ-assignment into $\mathcal{TS}(\Sigma,\Gamma)$, then $\varphi$ is a Σ-substitution, since it is a Δ-assignment. We can convince ourselves that $\mathcal{I}_\varphi(\mathbf{A}) = \sigma(\mathbf{A})\!\downarrow$ by a simple induction on the structure of formulae using

$$\sigma(\lambda X.\mathbf{A})@\mathbf{B} = (\lambda X.\sigma_{-X}\mathbf{A})@\mathbf{B} = \sigma, [\mathbf{B}/X]\mathbf{A} = \mathcal{I}_{\varphi,[\mathbf{B}/X]}(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{A})@\mathbf{B}$$

Thus $\mathcal{I}_\varphi(\mathbf{A}) = \sigma(\mathbf{A})\!\downarrow \in \mathcal{TS}_{\mathbb{A}}(\Sigma,\Gamma)$ by 3.5.1 and 3.4.4. In other words, $\mathcal{TS}(\Sigma,\Gamma)$ is comprehension-closed. Finally, for any term declaration $[\forall\Delta.\mathbf{A}::\mathbb{A}] \in \Sigma$ we have $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ by *ws:td*, and thus we can verify $\mathcal{I}_\varphi \in \mathcal{TS}_{\mathbb{A}}(\Sigma,\Gamma)$ with the same arguments. □

We now convince ourselves that we need only consider term declarations of a certain syntactically restricted form, since all other term declarations can be replaced by term declarations in this form without changing the set of well-formed formulae.

**Lemma 3.5.14** *Let Σ be a set of term declarations, $\mathcal{D} = [\forall\Gamma.\mathbf{B}::\mathbb{B}] \in \Sigma$, and let $\mathcal{D}' = [\forall\Gamma.\mathbf{B}'::\mathbb{B}]$, where $\mathbf{B}'$ is the $\beta\eta$-normal form of $\mathbf{B}$. Furthermore, let $\Sigma' = \{\mathcal{D}' \mid \mathcal{D} \in \Sigma\}$, then*

1. *$\vdash_{sig} \Sigma$ implies $\vdash_{sig} \Sigma'$*

2. *$\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, iff $\Gamma \vdash_{\Sigma'} \mathbf{A}::\mathbb{A}$*

3. *$\Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$, iff $\Gamma \vdash_{\Sigma'} \mathbf{A}=_{\beta\eta}\mathbf{B}$*

**Proof:** We prove the assertions by simultaneous induction on the structure of the ΣΛ-derivations involved.

1. If $\mathcal{E}\colon \vdash_{sig} \Sigma$ ends in *sig:td*, then $\Sigma = \Delta, [\forall\Gamma.\mathbf{B}::\mathbb{B}]$ and $\Gamma \vdash_\Delta \mathbf{B}::\mathbb{C}$ such that $\mathbb{B}$ **Rdom** $\mathbb{C}$. Let $\Delta' := \{\mathcal{D}' \mid \mathcal{D} \in \Delta\}$, then we have $\vdash_{sig} \Delta'$ by inductive hypothesis 1, and $\Gamma \vdash_{\Delta'} \mathbf{B}::\mathbb{C}$ by inductive hypothesis 2 (the proof of $\Gamma \vdash_\Delta \mathbf{B}::\mathbb{C}$ is a subderivation of $\mathcal{E}$). By 3.5.1 we have $\Gamma \vdash_{\Delta'} \mathbf{B}'::\mathbb{C}$, and therefore we obtain $\vdash_{sig} \Sigma'$ by a single application of *sig:td*. The *sig:const* case is trivial, since constants are already in $\beta\eta$-normal form. Note that the lemma is not true for long $\beta\eta$-normal forms, since we cannot add $\eta$-expanded forms of constants without declaring constants with a *sig:const* rule first.

2. Let $\mathcal{E}\colon \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, we only show that $\Gamma \vdash_{\Sigma'} \mathbf{A}::\mathbb{A}$, since the other direction is trivial. The only interesting cases are those, where $\mathcal{E}$ ends in *ws:td* or *ws:*$\beta\eta$, since *ws:var* is trivial, and *ws:app* and *ws:abs* are direct consequences of the inductive hypothesis. For the *ws:td* case let $\mathcal{E}$ end in

$$\frac{[\forall\Gamma'.\mathbf{A}::\mathbb{A}] \in \Sigma \quad \Gamma' \subseteq \Gamma}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\, ws{:}td$$

   and let $\mathbf{B}$ be the $\beta\eta$-normal form of $\mathbf{A}$. Then by the construction we must have $[\forall\Gamma'.\mathbf{B}::\mathbb{A}] \in \Sigma'$, so we can get $\Gamma \vdash_{\Sigma'} \mathbf{B}::\mathbb{A}$ by *ws:td*, and moreover $\Gamma \vdash_{\Sigma'} \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ by 3. Finally, we obtain the assertion $(\Gamma \vdash_{\Sigma'} \mathbf{A}::\mathbb{A})$ with 3.5.1. The case where $\mathcal{E}$ ends in *ws:*$\beta\eta$ is similarly obtained with the inductive hypothesis and 3.

3. The assertion is a direct consequence of the inductive hypothesis for the inductive rules for congruence judgments. Furthermore, the preconditions of *sort:*$\beta$*:top* and *sort:*$\eta$*:top* only consist of well-sortedness conditions, so the assertion is a simple consequence of 2.

□

**General Assumption 3.5.15** In the following we only consider signatures $\Sigma$ such that in all term declarations $[\forall\Gamma.\mathbf{A}::\mathbb{A}] \in \Sigma$ the formula $\mathbf{A}$ is a $\beta\eta$-normal form. As the previous lemma shows, this is only a syntactic restriction, since for any valid, unrestricted signature $\Sigma$ we can give a valid, restricted signature $\Sigma'$ that yields the same well-sorted formulae. Therefore this restrictions do not amount to a restriction of the expressive power of ΣΛ.

## 3.6 Sort Inclusion

We can make the following observation: if $\Gamma \vdash_\Sigma X::\mathbb{B}$ and $\Gamma(X) = \mathbb{A}$, then $\mathcal{D}_\mathbb{A} \subseteq \mathcal{D}_\mathbb{B}$. This is just the situation that is captured with the notion of sort inclusion in sorted logics without term declarations. In such systems the subsort relation is the smallest partial ordering that contains a set of subsort declarations. The subsort relation plays such a central role in these systems that they are collectively called "order-sorted". For specifying mathematics in ΣΛ the notion of subsorting also plays an important role, since it allows to specify taxonomic hierarchies of sorts, which, for instance, occur in the definitional hierarchies of mathematics and help to guide mathematical intuition. Since subsorting in $\Sigma\Lambda$, where we have term declarations, is a derived relation, we do not have to treat it in our meta-logical development. On the object level (and for computation) however, it is a useful notion to employ.

**Definition 3.6.1 (Subsort Relation)** We say that $\mathbb{A}$ is a **subsort of** $\mathbb{B}$ in $\Sigma$, iff $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$, and write $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$.

In contrast to the first-order systems, the subsort relation in $\Sigma\Lambda$ is not finite, even when we assume a finite set of base sorts. Thus the relation cannot be pre-computed in advance. Furthermore, it is not clear whether the sort-checking problem is decidable (see the discussion in subsection 4.3), which is another reason for the limited practical usefulness of the full subsorting relation. One way out of this situation is to define a computable partial ordering relation that approximates the full subsort relation by closing a set of subsort declarations under certain inductive principles that induce subsort relations on higher types from such for lower types.

For our approach it is only essential that the subsort declarations are correct, i.e. that $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$, whenever $\mathbb{A} \leq \mathbb{B}$ is a subsort declaration. Completeness, i.e. that the subsort relation defined from the declarations captures *all* semantical subset relation, only plays a role for the effectiveness of actual computational algorithms, since the term declaration mechanism assures completeness of the calculi, even if we chose an empty set of term declarations.

**Definition 3.6.2** Let $\mathcal{R}$ be a typed binary relation on sorts such that $\mathbb{A} \sqsubseteq \mathbb{B}$ whenever $\mathcal{R}(\mathbb{A}, \mathbb{B})$, then we call $\mathcal{R}$ an **approximation of $\sqsubseteq$ in** $\Sigma$.

We now identify special term declarations that we use to build a concrete approximation of $\sqsubseteq$.

**Definition 3.6.3 (Subsort Declaration)** We abbreviate term declarations of the form $[\forall\Gamma.(\lambda\overline{Y^k}.Z)::\mathbb{C}]$ by $[\mathbb{A} \leq \mathbb{B}]$, and call them **subsort declarations**, iff

1. $\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}$,
2. $\Gamma(Z) = \mathbb{A}$ or there is a number $i \leq k$ such that $Z = Y^i$ and $\mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$.

We denote the set of subsort declarations in $\Sigma$ by $\mathcal{SD}(\Sigma)$.

**Lemma 3.6.4** *If* $[\mathbb{A} \leq \mathbb{B}] \in \Sigma$, *then* $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$.

**Proof sketch:** Let $[\mathbb{A} \leq \mathbb{B}] := [\forall\Gamma.(\lambda\overline{Y^k}.Z)::\mathbb{C}] \in \Sigma$, $\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}$, and $\Gamma(Z) = \mathbb{A}$ or there is a number $i \leq k$ such that $Z = Y^i$ and $\mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$. Furthermore, let $\Gamma' = \Gamma, \overline{[X^k::\mathfrak{d}^k(\mathbb{C})]}$, then we have a $\Sigma\Lambda$-derivation of the form

$$\cfrac{\cfrac{\cfrac{}{\Gamma' \vdash_\Sigma (\lambda\overline{Y^k}.Z)::\mathbb{C}}\; ws{:}td}{(\lambda\overline{Y^k}.Z)\overline{X^k}::\mathfrak{r}^k(\mathbb{C})}\; ws{:}app^k}{\Gamma' \vdash_\Sigma Z::\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}}\; ws{:}\beta\eta$$

where $\Gamma'(Z) = \Gamma(Z) = \mathbb{A}$ or there is a number $i \le k$ such that $\Gamma'(Z) = \Gamma'(X^i) = \mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$. This yields the assertion. $\square$

**Definition 3.6.5 (Sort Inclusion)** Let $\mathcal{R}$ be an approximation of $\sqsubseteq$, then the following inference system is called the $\Sigma\Lambda$ **subsort inference system for** $\mathcal{R}$

$$\frac{\mathcal{R}(\mathbb{A},\mathbb{B}) \quad \vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{B}}\, ior{:}start \qquad\qquad \frac{\vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{A}}\, ior{:}ref$$

$$\frac{\vdash_{sig} \Sigma \quad \mathbb{A} \in \mathcal{BS}^f}{\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})}\, ior{:}nat$$

$$\frac{\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{B} \quad \Sigma \vdash \mathbb{B} \le_{\mathcal{R}} \mathbb{C}}{\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{C}}\, ior{:}trans \qquad\qquad \frac{\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{B}}{\Sigma \vdash \mathbb{C} \to \mathbb{A} \le_{\mathcal{R}} \mathbb{C} \to \mathbb{B}}\, ior{:}cov$$

We call the relation $\mathcal{R}^{\le}$, defined by $\mathcal{R}^{\le}(\mathbb{A},\mathbb{B})$, iff $\Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{B}$, the **ordering relation for** $\mathcal{R}$.

**Theorem 3.6.6** *If $\mathcal{R}$ is an approximation of the subsort relation of $\Sigma$, then the relation $\mathcal{R}^{\le}$ is also an approximation. Moreover $\mathcal{R}^{\le}$ is a quasi-ordering.*

**Proof:** We prove that $[X::\mathbb{A}] \vdash_{\Sigma} X::\mathbb{B}$ by induction on the structure of $\mathcal{D}: \Sigma \vdash \mathbb{A} \le \mathbb{B}$. We first consider the base cases. If $\mathcal{D}$ ends in *ior:start*, we obtain the assertion from the hypothesis, that $\mathcal{R}$ is an approximation, and the *ior:ref* case corresponds to the *ws:var* rule. For the *ior:nat* rule consider the following $\Sigma\Lambda$-derivation,

$$\cfrac{\cfrac{\cfrac{}{[X::\mathbb{A}] \vdash_{\Sigma} X::\mathbb{A}}\, ws{:}var \qquad \cfrac{}{[Y::\mathfrak{d}(\mathbb{A})] \vdash_{\Sigma} Y::\mathfrak{d}(\mathbb{A})}\, ws{:}var}{[X::\mathbb{A}], [Y::\mathfrak{d}(\mathbb{A})] \vdash_{\Sigma} XY::\mathfrak{r}(\mathbb{A})}\, ws{:}app}{[X::\mathbb{A}] \vdash_{\Sigma} (\lambda Y_{\mathfrak{d}(\mathbb{A})}.XY)::\mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})}\, ws{:}abs$$

which yields $[X::\mathbb{A}] \vdash_{\Sigma} X::\mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})$ by a single application of lemma 3.5.1.

For the inductive cases we see that the *ior:trans* rule can be recovered with the *ws:inst* rule

$$\cfrac{\cfrac{\mathcal{A}}{[X::\mathbb{A}] \vdash_{\Sigma} X::\mathbb{B}} \qquad \cfrac{\mathcal{B}}{[Y::\mathbb{B}] \vdash_{\Sigma} Y::\mathbb{C}}}{[X::\mathbb{A}] \vdash_{\Sigma} X::\mathbb{C}}\, ws{:}inst$$

For *ior:cov* let $\mathcal{A}: \Sigma \vdash \mathbb{A} \le_{\mathcal{R}} \mathbb{B}$, then by inductive hypothesis we have a $\Sigma\Lambda$-derivation $\mathcal{A}': [Z::\mathbb{A}] \vdash_{\Sigma} Z::\mathbb{B}$. Furthermore, we have

$$\cfrac{\cfrac{}{[X::\mathbb{C} \to \mathbb{A}] \vdash_{\Sigma} X::\mathbb{C} \to \mathbb{A}}\, ws{:}var \qquad \cfrac{}{[Y::\mathbb{C}] \vdash_{\Sigma} Y::\mathbb{C}}\, ws{:}var}{[X::\mathbb{C} \to \mathbb{A}], [Y::\mathbb{C}] \vdash_{\Sigma} XY::\mathbb{A}}\, ws{:}app$$

and thus

$$\dfrac{\dfrac{[X::\mathbb{C} \to \mathbb{A}], [Y::\mathbb{C}] \vdash_\Sigma XY::\mathbb{A} \qquad \dfrac{\mathcal{A}'}{[Z::\mathbb{A}] \vdash_\Sigma Z::\mathbb{B}}}{[X::\mathbb{C} \to \mathbb{A}], [Y::\mathbb{C}] \vdash_\Sigma XY::\mathbb{B}}\ \textit{ws:inst}}{[X::\mathbb{A} \to \mathbb{C}] \vdash_\Sigma (\lambda Y_{\mathbb{C}}.XY)::\mathbb{C} \to \mathbb{B}}\ \textit{ws:abs}$$

Just as above we can conclude $[X::\mathbb{C} \to \mathbb{A}] \vdash_\Sigma X::\mathbb{C} \to \mathbb{B}$ with lemma 3.5.1. The claim that $\mathcal{R}^{\leq}$ is a quasi-ordering is an obvious consequence of the admissibility of *ior:trans*. □

**Notation 3.6.7** As a consequence of the previous theorem semantic subsort relation is a partial ordering and closed under *ior:cov* and *ior:nat*, since it also is an approximation of $\sqsubseteq$. We have seen in 3.6.4 the set of subsort declarations for an approximation of $\sqsubseteq$. Thus the relation given by the judgment $\Sigma \vdash \mathbb{A} \leq_{\mathcal{SD}(\Sigma)} \mathbb{B}$ is an approximation of $\sqsubseteq$, which we simply denote by $\Sigma \vdash \mathbb{A} \leq \mathbb{B}$.

**Definition 3.6.8 (Weakening)** Let $\mathcal{R}$ be an approximation of the subsort relation in $\Sigma$, then the following inference rule is called the **weakening rule for** $\mathcal{R}$:

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{B}}\ \textit{ws:weaken}(\mathcal{R})$$

As above we denote the inference rule *ws:weaken*$(\mathcal{SD}(\Sigma))$ just by *ws:weaken*.

**Lemma 3.6.9** *Let $\mathcal{R}$ be an approximation of the subsort relation in $\Sigma$, then*

1. *ws:weaken$(\mathcal{R})$ is admissible in $\Sigma\Lambda$.*
2. *A formula $\mathbf{A}$ is well-sorted, iff the set $\mathcal{S}(\mathbf{A})$ of sorts of $\mathbf{A}$ is a nonempty upper segment for $\mathcal{R}^{\leq}$.*

**Proof:** By *ws:inst* and *ws:weaken*$(\mathcal{R})$ and the relevant definitions. □

**Theorem 3.6.10** *Let $\mathcal{R}$ be an approximation of the subsort relation in $\Sigma$, then $\mathbb{A}$ **Rdom** $\mathbb{B}$ whenever $\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$. In particular, we have $\tau(\mathbb{A}) = \tau(\mathbb{B})$.*

**Proof:** By theorem 3.6.6 we have $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$, but we also have $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{A}$ by *ws:var*, therefore $\mathbb{A}$ **Rdom** $\mathbb{B}$ by 3.5.4. □

**Remark 3.6.11** As a consequence the sets $\mathcal{S}_\alpha$ are mutually incomparable, i.e. if $\mathbb{A} \in \mathcal{S}_\alpha$, $\mathbb{B} \in \mathcal{S}_\beta$, and $\alpha \neq \beta$, then we can never have $\Gamma \vdash_\Sigma \mathbb{A} \sqsubseteq \mathbb{B}$. The most important consequence of this is that we can only have finite ascending and descending chains of sorts with respect to $\sqsubseteq$.

**General Assumption 3.6.12** Let $\Sigma$ be a sorted higher-order signature, and let $\sim$ be the equivalence relation induced by $\leq_\Sigma$, that is, $\Sigma \vdash \mathbb{A} \sim \mathbb{B}$, iff $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$ and $\Sigma \vdash \mathbb{B} \sqsubseteq \mathbb{A}$. Observe that for $\Sigma \vdash \mathbb{A} \sim \mathbb{B}$ we can derive $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ whenever we can derive $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{B}$ by application of the *ws:weaken* inference rule.

If we pass to the quotient signature $\Sigma'$ with respect to $\sim$, that is, for any equivalence class in $\mathcal{S}$ we pick a representative and replace sorts by their representative, then we get a signature $\Sigma'$, where $\sqsubseteq'$ is a partial ordering, which entails that $\sim'$ is trivial. Furthermore, $\Sigma'$ is valid whenever $\Sigma$ is, since the inference system for valid signatures is only concerned about sorts in term declarations respecting function domains, and we have $\mathbb{A}$ **Rdom** $\mathbb{B}$ whenever $\mathbb{A} \sim \mathbb{B}$ by 3.6.10

If we always take care to pick a representative of maximal length, then we can never have $\mathfrak{d}(\mathbb{C}) \to \mathfrak{r}(\mathbb{C}) \sqsubset \mathbb{C}$ for any base sort $\mathbb{C} \in \mathcal{BS}^f$, since we always have the converse by *ior:nat*. Thus we can assume that $\sqsubseteq$ is a partial ordering with $\mathbb{C} \sqsubset \mathfrak{d}(\mathbb{C}) \to \mathfrak{r}(\mathbb{C})$ for all functional base sorts $\mathbb{C} \in \mathcal{BS}^f$. While we will not need this assumption in the following, other signatures would in some sense be redundant and thus non-optimal in practice.

**Remark 3.6.13 (Contravariance and Function Restriction)** In our sort system the rule for **contravariance in the domain sort**

$$\frac{\Sigma \vdash \mathbb{A} \leq \mathbb{B}}{\Sigma \vdash \mathbb{B} \to \mathbb{C} \leq \mathbb{A} \to \mathbb{C}}\ \textit{ior:contra}$$

which corresponds to function restriction cannot be admissible, since it contradicts 3.6.10. This defect in symmetry is intended, since we want functions to have unique supporting sorts (cf. 3.5.9). The natural notion of semantics for Church-style $\lambda$-calculi for mathematics with covariance seems to be a total function semantics, where a declaration of the form $[f::\mathbb{A} \to \mathbb{B}]$ has the intended semantics that the denotation of $f$ is a total (on some predefined universe) function that, when restricted to some subdomain $\mathcal{D}_\mathbb{A}$, yields values in the subset $\mathcal{D}_\mathbb{B}$ of the universe. While this is a reasonable semantics for many applications (even the $\eta$-rule can be given a reasonable semantics), it is not the one intended in this thesis. For $\lambda$-calculi with contravariance see [NQ92, KP93]. In our system we do not have to treat function restriction as a built-in implicit notion, since we can make it explicit: for any formula $\mathbf{A}$ of function sort $\mathbb{A}$, the function $\mathcal{I}_\varphi(\lambda X_\mathbb{B}.\mathbf{A}X)$ is the restriction of $\mathcal{I}_\varphi(\mathbf{A})$ to the subset $\mathcal{D}_\mathbb{B} \subseteq \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$, if $\mathbb{B}$ is a subsort of $\mathfrak{d}(\mathbb{A})$.

**Remark 3.6.14 (Top Sort)** In contrast to many other expositions of sorted logics we do not require the existence of a maximal sort (usually called **top sort**), since this concept does not mix well with $\lambda$-calculi that respect function domains. Even if we postulate a top sort $\mathbb{A}$ for each base type $\alpha$, then the function types do not have maximal sorts. Consider for instance, the sorts $\mathbb{A} \to \mathbb{A}$ and $\mathbb{B} \to \mathbb{A}$ that are incomparable for any other sort $\mathbb{B}$ of type $\alpha$. For $\lambda$-calculi with contravariance the concept of a top sort $\top$ is no problem, if it is accompanied by a least sort $\bot$, as then the sorts $\bot \to \top$ and $\top \to \bot$ are the top and least sorts of functional type.

**Remark 3.6.15** Considering the semantics for $\Sigma\Lambda$, we see that a more general version of the *ior:cov* inference rule

$$\frac{\Sigma \vdash \mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B}) \quad \Sigma \vdash \mathfrak{r}(\mathbb{A}) \leq \mathfrak{r}(\mathbb{B})}{\Sigma \vdash \mathbb{A} \leq \mathbb{B}}\ ior{:}cov'$$

would not be correct. Let $\mathcal{D}_{\mathbb{A}}$ and $\mathcal{D}_{\mathbb{B}}$ be the classes of surjective functions in $\mathcal{F}(\mathcal{D}_{\mathfrak{d}(\mathbb{A})}; \mathcal{D}_{\mathfrak{r}(\mathbb{A})})$ and $\mathcal{F}(\mathcal{D}_{\mathfrak{d}(\mathbb{B})}; \mathcal{D}_{\mathfrak{r}(\mathbb{B})})$. If $\Sigma \vdash \mathfrak{r}(\mathbb{A}) \leq \mathfrak{r}(\mathbb{B})$, then $\mathcal{D}_{\mathfrak{r}(\mathbb{A})} \subseteq \mathcal{D}_{\mathfrak{r}(\mathbb{B})}$ in any $\Sigma$-structure $(\mathcal{D}, @, \mathcal{I})$ but functions from $\mathcal{D}_{\mathbb{A}}$ would in general not be surjective in the larger codomain $\mathcal{D}_{\mathfrak{r}(\mathbb{B})}$ and therefore not in $\mathcal{D}_{\mathbb{B}}$. For similar reasons an inference rule like

$$\frac{\mathbb{F} \leq \mathbb{G}}{\mathfrak{r}(\mathbb{F}) \leq \mathfrak{r}(\mathbb{G})}\ ior{:}nat^{-1}$$

cannot be correct. Let $\mathcal{D}_{\mathbb{A}} := \{a\}$, $\mathcal{D}_{\mathbb{B}} := \{b, d\}$, $\mathcal{D}_{\mathbb{C}} := \{b, c\}$, $\mathfrak{r}(\mathbb{F}) = \mathbb{B}$, $\mathfrak{d}(\mathbb{F}) = \mathbb{A}$, and $\mathbb{G} := \mathbb{A} \to \mathbb{C}$. Then the definitions $\mathcal{D}_{\mathbb{F}} := \{\{(a, c)\}\}$ and $\mathcal{D}_{\mathbb{G}} := \{\{(a, b)\}, \{(a, c)\}\}$ satisfy the preconditions ($\mathcal{D}_{\mathbb{F}} \subseteq \mathcal{D}_{\mathbb{G}}$), but clearly we do not have $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}_{\mathbb{B}}$.

**Definition 3.6.16** A signature $\Sigma$ is called **regular**, iff each well-sorted formula has a unique least sort with respect to $\leq_{\Sigma}$. In regular signatures we denote the **unique least sort of A** by $\mu\mathcal{S}_{\Sigma}(\mathbf{A})$.

**Remark 3.6.17** Regular signatures are very desirable for practical purposes, since they allows to label terms $\mathbf{A}$ only with the sort $\mu\mathcal{S}_{\Sigma}(\mathbf{A})$, instead of $\mathcal{S}_{\Sigma}(\mathbf{A})$, which is a – in general rather large – set of sorts. It will turn out that regularity is also a precondition for our pre-unification algorithm. Unfortunately, it will also turn out that the regularity is an undecidable property for signatures in general (cf. 4.6.2). In $\lambda$-calculi with intersection sorts (see the discussion in 3.1.6) signatures would be trivially regular. However, in such systems the problem whether for any given sort $\mathbb{A}$ there is a closed formula $\mathbf{A}$ with $\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}$ is undecidable.

**Example 3.6.18** Let $\Sigma := \{[a{::}\mathbb{A}], [a{::}\mathbb{B}], [\mathbb{A} \leq \mathbb{C}], [\mathbb{B} \leq \mathbb{C}]\}$ be a valid signature, then $\mu\mathcal{S}_{\Sigma}(a) = \{\mathbb{A}, \mathbb{B}\}$, and therefore $\Sigma$ is not regular.

# 4 Computational Aspects $\Sigma\Lambda$

In this section we investigate computational properties and algorithms for the judgments defined in section 5 and take a look at the $\Sigma$-unification and $\Sigma$-matching problems respectively. The algorithms will be central inference procedures for the resolution calculus presented in section 6. Building upon the notion of general binding we give a set of transformations for general $\Sigma$-unification and pre-$\Sigma$-unification, which will be shown correct and complete based on methods from [Sny91].

Unfortunately, many of the discussed problems will obtain turn out to be undecidable, and we will have to take special precautions in our attempts to mechanize $\Sigma\Lambda$. There are two sources of undecidability in $\Sigma\Lambda$, namely

- term declarations (regularity and $\Sigma$-unification are undecidable even for the first-order fragment [SS89]).
- $\beta\eta$-equality (unsorted unification for second-order $\lambda$-calculi is undecidable [Hue73, Luc72, Gol81, Far91a])

However, we conjecture that the undecidability problem is mainly a theoretical one, since the class of formulae, where undecidability occurs will not appear in most practical theorem proving applications. In particular, if $\Sigma\Lambda$ is used as a language for mathematical problem specification, the complexity of term declarations is not a source of undecidability, since the term declaration mechanism is mainly used to code taxonomic hierarchies, where the declaration formulae are members of subclasses (e.g. higher-order patterns), where $\Sigma$-unification is decidable.

## 4.1 Structure Theorem

The key tool for the investigation of well-sorted formulae is the structure theorem, which we are about to prove. The principal difficulty of $\Sigma\Lambda$ is that the property of well-sortedness is highly non-structural, which makes the classical deduction methods, such as unification, that analyze the structure of formulae difficult. The structure theorem recovers structural properties of well-sorted formulae by linking the sort information (the existence of certain term declarations) with structural information about normal forms.

**Definition 4.1.1 (Semi-Structural $\Sigma\Lambda$-Derivation)** We call a $\Sigma\Lambda$-derivation $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ in **semi-structural** form, iff $\mathcal{A}$ is of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma,\Xi^l \vdash_\Sigma \mathbf{H}::\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma,\Xi^l \vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{B})}}{\Gamma,\Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}::\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})}\;ws{:}app^m}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}}\;ws{:}abs^l \qquad \dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}} \qquad \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\;ws{:}\beta\eta$$

where

1. $l = \mathbf{ln}(\mathbb{A})$, $m = l + \mathbf{ln}(\tau(\mathbb{B})) - \mathbf{ln}(\tau(\mathbb{A})) \geq 0$,
2. $\Xi^j$ is the variable context $[X^1{::}\mathfrak{d}(\mathbb{A})], \ldots, [X^j{::}\mathfrak{d}^l(\mathbb{A})]$,
3. the subderivations $\mathcal{D}^i\colon \Xi^l, \Gamma \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})$ are also semi-structural,
4. one of the following holds for the $\Sigma\Lambda$-derivation $\mathcal{H}\colon \Gamma, \Xi^l \vdash_\Sigma \mathbf{H}{::}\mathbb{B}$:
   (a) $\mathbf{H}$ is a variable with $\Gamma, \Xi^l(\mathbf{H}) = \mathbb{B}$, and $\mathcal{H}$ consists of a single *ws:var* step. Note that $\mathbf{H} = \mathbf{head}(\mathbf{A})$, since $\Xi^l, \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}$.
   (b) $\mathbf{H} = \theta(\mathbf{B})$ for some term declaration $[\forall\Delta.\mathbf{B}{::}\mathbb{B}] \in \Sigma$ and some substitution $\theta$, and $\mathcal{H}$ is a $\Sigma\Lambda$-derivation of the form

$$\dfrac{\dfrac{[\forall\Delta'.\mathbf{B}{::}\mathbb{B}] \in \Sigma}{\Delta \vdash_\Sigma \mathbf{B}{::}\mathbb{B}}\,ws{:}td \qquad \dfrac{\mathcal{E}}{\Gamma, \Xi^l \vdash_\Sigma \theta{::}\Delta}}{\Gamma, \Xi^l \vdash_\Sigma \theta(\mathbf{B}){::}\mathbb{B}}\,ws{:}inst$$

   where the subderivations $\mathcal{E}^i$ of $\mathcal{E}\colon \Gamma, \Xi^l \vdash_\Sigma \theta{::}\Delta$ are again semi-structural. In this case $\Xi^l, \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}$ entails that we have the following three cases for the head of $\mathbf{B}$: $\mathbf{head}(\mathbf{B}) = \mathbf{head}(\mathbf{A})$, or $\mathbf{head}(\mathbf{B}) = j$, or $\mathbf{head}(\mathbf{B}) \in \mathbf{Dom}(\Delta)$.

The subderivations $\mathcal{D}^i$ and $\mathcal{E}^i$ are called the **principal subderivations** of $\mathcal{A}$.

If $\mathbf{A} \doteq \mathbf{H}\overline{\mathbf{D}^s}$, where $s = m - l = \mathbf{ln}(\tau(\mathbb{B})) - \mathbf{ln}(\tau(\mathbb{A})) \geq 0$, then $\mathcal{A}$ is of the form[12]

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma \vdash_\Sigma \mathbf{H}{::}\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})}}{\Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}{::}\mathfrak{r}^s(\mathbb{B})}\,ws{:}app^s \qquad *}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}\,\overline{X^l}{::}\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})}\,ws{:}app^l}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^s}\,\overline{X^l}{::}\mathbb{A}}\,ws{:}abs^l \qquad * \quad \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^s}\,\overline{X^l} =_\eta \mathbf{H}\overline{\mathbf{D}^s}}{\Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}{::}\mathbb{A}}\,ws{:}\beta\eta$$

Clearly this $\Sigma\Lambda$-derivation is redundant, since in this case $\mathfrak{r}^s(\mathbb{B}) = \mathbb{A}$, and thus the judgment $\Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}{::}\mathbb{A}$ is derived twice in $\mathcal{A}$. Thus we can shorten it to the $\Sigma\Lambda$-derivation

$$\dfrac{\dfrac{\mathcal{H}}{\Gamma \vdash_\Sigma \mathbf{H}{::}\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})}}{\Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}{::}\mathfrak{r}^s(\mathbb{B})}\,ws{:}app^s$$

We call $\Sigma\Lambda$-derivations of this form in **restricted semi-structural form**. Note that $\Sigma\Lambda$-derivations in restricted semi-structural form can always be trivially extended to such in semi-structural form.

[12]We have abbreviated some subderivations by *, in order to conserve space.

**Theorem 4.1.2 (Structure Theorem)** *If* $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$*, then there is a semi-structural* $\Sigma\Lambda$*-derivation of* $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$.

**Proof sketch:** The course of the proof (which will take up the rest of this subsection) will be to define a relation $\mathcal{SR}$, which captures the content of the structure theorem, and a relation $\mathcal{LR}$, that transports the structure information from the base sorts to the arrow sorts. These relations coincide on the base sorts, and we show in 4.1.10 that $\mathcal{LR} \subseteq \mathcal{SR}$ and in 4.1.11 that all well-sorted formulae are in $\mathcal{LR}$, which together entail the assertion of the structure theorem. $\square$

**Definition 4.1.3 (Structure Relation)** Let $\mathbf{A}$ be a well-sorted formula with $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, then $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$ (we say the **structure relation** $\mathcal{SR}$ holds on $\mathbf{A}$ and $\mathbb{A}$), iff there is a semi-structural $\Sigma\Lambda$-derivation of $\mathcal{A}: \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$. Similarly we define the **restricted structure relation** $\mathcal{RSR}$ by the existence of a $\Sigma\Lambda$-derivation in restricted semi-structural form. Note that $\mathcal{RSR}$ is a sub-relation of $\mathcal{SR}$, since $\Sigma\Lambda$-derivations in restricted semi-structural form can always be extended to such in semi-structural form. For a substitution $\sigma$ with $\Gamma \vdash_\Sigma \sigma::\Delta$ we write $\Gamma \vdash_\Sigma \mathcal{SR}(\sigma;\Delta)$, iff for all $X \in \mathbf{Dom}(\sigma)$ we have $\Gamma \vdash_\Sigma \mathcal{SR}(\sigma(X);\Delta(X))$.

Note that $\mathcal{SR}$ as well as $\mathcal{RSR}$ are monotonic, i.e. $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$, if $\Delta \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$ and $\Delta \subseteq \Gamma$.

We now prove some technical lemmata, which we need in the proofs later on.

**Lemma 4.1.4** *Let* $\Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$*, then* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{B};\mathbb{A})$ *iff* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$.

**Proof:** Let $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$, so we have a semi-structural $\Sigma\Lambda$-derivation of the form

$$\dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}} \quad \dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}} \quad \dfrac{\mathcal{C}}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}=_{\beta\eta}\mathbf{A}}}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\ \mathit{ws}{:}\beta\eta$$

Let $\mathcal{C}': \Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$, then we can obtain a $\Sigma\Lambda$-derivation $\mathcal{C}'': \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}=_{\beta\eta}\mathbf{B}$ with *ms:trans*. Moreover, by 3.5.1 there is a $\Sigma\Lambda$-derivation $\mathcal{A}'': \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}$, thus

$$\dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}} \quad \dfrac{\mathcal{C}''}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}=_{\beta\eta}\mathbf{B}}}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}}\ \mathit{ws}{:}\beta\eta$$

is a semi-structural $\Sigma\Lambda$-derivation that verifies the claim. $\square$

**Lemma 4.1.5** *If* $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A};\mathbb{A})$ *and* $\Gamma' \vdash_\Sigma \mathcal{SR}(\mathbf{C};\mathfrak{d}(\mathbb{A}))$ *with* $\Gamma||\Gamma'$*, then we have* $\Gamma',\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{AC};\mathfrak{r}(\mathbb{A}))$.

**Proof:** Let $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ be in restricted semi-structural form

$$\frac{\dfrac{\mathcal{H}}{\Gamma \vdash_\Sigma \mathbf{H}{::}\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})}}{\Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}{::}\mathfrak{r}^s(\mathbb{B})}\ \mathit{ws{:}app}^s$$

where $s = \mathbf{ln}(\tau(\mathbb{B})) - \mathbf{ln}(\tau(\mathbb{A})) \geq 0$. Let $s' := \mathbf{ln}(\tau(\mathbb{B})) - \mathbf{ln}(\tau(\mathfrak{r}(\mathbb{A}))) = s + 1 \geq 0$, then $\mathbf{AC} = \mathbf{H}\overline{\mathbf{D}^s}\mathbf{C} = \mathbf{H}\overline{\mathbf{D}}^{s'}$, if we set $\mathbf{D}^{s+1} := \mathbf{C}$, and therefore the $\Sigma\Lambda$-derivation

$$\frac{\dfrac{\mathcal{H}}{\Gamma \vdash_\Sigma \mathbf{H}{::}\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})} \quad \dfrac{\mathcal{D}^{s+1}}{\Gamma' \vdash_\Sigma \mathbf{D}^{s+1}{::}\mathfrak{d}^{s+1}(\mathbb{B})}}{\Gamma', \Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^s}{::}\mathfrak{r}^{s'}(\mathbb{B})}\ \mathit{ws{:}app}^{s'}$$

is in restricted semi-structural form, if we take $\mathcal{D}^{s'}\colon \Gamma' \vdash_\Sigma \mathbf{C}{::}\mathfrak{d}(\mathbb{A})$ to be the $\Sigma\Lambda$-derivation in semi-structural form guaranteed by the assumption $\Gamma' \vdash_\Sigma \mathcal{SR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$. $\square$

**Lemma 4.1.6** *If* $\Gamma, [Y{::}\mathbb{C}] \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{D})$, *then* $\Gamma \vdash_\Sigma \mathcal{SR}(\lambda Y_{\mathbb{C}}.\mathbf{A}; \mathbb{C} \to \mathbb{D})$.

**Proof:** Let $\Gamma' = \Gamma, [Y{::}\mathbb{C}]$ and $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{D}$ be of the form

$$\frac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma', \Xi^l \vdash_\Sigma \mathbf{H}{::}\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma', \Xi^l \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})}}{\Gamma', \Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}{::}\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{D})}\ \mathit{ws{:}app}^m}{\Gamma' \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}{::}\mathbb{D}}\ \mathit{ws{:}abs}^l \qquad \mathcal{B} \quad \dfrac{\mathcal{C}}{\Gamma' \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}}}{\Gamma' \vdash_\Sigma \mathbf{A}{::}\mathbb{D}}\ \mathit{ws{:}\beta\eta}$$

where $\mathcal{B}\colon \Gamma' \vdash_\Sigma \mathbf{A}{::}\mathbb{D}$. We have $\mathcal{C}'\colon \Gamma \vdash_\Sigma \lambda Y_{\mathbb{C}}.\mathbf{A} =_{\beta\eta} \lambda Y\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}$ by *tr:abs*, thus the following $\Sigma\Lambda$-derivation $\mathcal{A}'$ is in semi-structural form

$$\frac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma', \Xi^l \vdash_\Sigma \mathbf{H}{::}\mathbb{B}} \qquad \dfrac{\mathcal{D}^i}{\Gamma', \Xi^l \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})}}{\Gamma', \Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}{::}\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^{l+1}(\mathbb{C} \to \mathbb{D})}\ \mathit{ws{:}app}^m}{\Gamma \vdash_\Sigma \lambda Y\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}{::}\mathbb{C} \to \mathbb{D}}\ \mathit{ws{:}abs}^{l+1} \qquad \mathcal{B}' \quad \dfrac{\mathcal{C}'}{\Gamma \vdash_\Sigma \lambda Y\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \lambda Y_{\mathbb{C}}.\mathbf{A}}}{\Gamma \vdash_\Sigma \lambda Y_{\mathbb{C}}.\mathbf{A}{::}\mathbb{C} \to \mathbb{D}}\ \mathit{ws{:}\beta\eta}$$

where $\mathcal{B}'$ is the obvious proof of well-sortedness of $\lambda Y_{\mathbb{C}}.\mathbf{A}$ obtained from $\mathcal{B}$ with *ws:abs*. $\square$

**Remark 4.1.7** If we look closely at the proof above we can see that $\mathbf{dp}(\mathcal{A}') = \mathbf{dp}(\mathcal{A}) + 1$, since $\mathbf{dp}(\mathcal{B}') = \mathbf{dp}(\mathcal{B}) + 1$, $\mathbf{dp}(\mathcal{C}') = \mathbf{dp}(\mathcal{C}) + 1$, and there is an additional *ws:app* step in the left branch of $\mathcal{A}'$.

We now want to define a second relation $\mathcal{LR}$ for the structure theorem and investigate its relation to $\mathcal{SR}$.

**Definition 4.1.8 (Logical Relation)** The **logical structure relation** is inductively defined by

1. $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, iff $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$ and $\mathbb{A} \in \mathcal{S}^{nf}$.
2. $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, iff $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, and for all formulae $\mathbf{C}$ with $\Gamma' || \Gamma$ and $\Gamma' \vdash_\Sigma \mathcal{LR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$ we have $\Gamma', \Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$.

For a $\Sigma$-substitution $\sigma$ with $\Gamma \vdash_\Sigma \sigma{::}\Delta$ we define $\mathcal{LR}(\sigma, \Delta)$ to hold, iff $\Gamma \vdash_\Sigma \mathcal{LR}(\sigma(X); \Delta(X))$ holds for all $X \in \mathbf{Dom}(\sigma)$. Note that for the identity substitution $\emptyset$ we can vacuously conclude $\Gamma \vdash_\Sigma \mathcal{LR}(\emptyset, \emptyset)$.

The course of the proof of the structure theorem is to show that for a given signature $\Sigma$ and context $\Gamma$ the relation $\mathcal{LR}$ subsumes the relation $\mathcal{SR}$. Then we will prove that for all well-sorted formulae $\mathbf{A}$ of sort $\mathbb{A}$ we have $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$ and therefore $\mathcal{SR}(\mathbf{A}; \mathbb{A})$, which is just the assertion of the structure theorem. However, we first need a couple of technical lemmata.

**Lemma 4.1.9 (Closure under Head-Equality)** *If* $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$, *then we have* $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, *iff* $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{B}; \mathbb{A})$ *holds.*

**Proof:** For $\mathbb{A} \in \mathcal{S}^{nf}$ we obtain the assertion with 4.1.4. Let $\Gamma || \Gamma'$ and $\Gamma' \vdash_\Sigma \mathcal{LR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$, then $\Gamma', \Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$ and $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{BC}; \mathfrak{r}(\mathbb{A}))$, since $\Gamma', \Gamma \vdash_\Sigma \mathbf{AC} =_{\beta\eta} \mathbf{BC}$ by *tr:app:fn* and the inductive hypothesis, therefore we obtain $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{B}; \mathbb{A})$ by definition of $\mathcal{LR}$. □

**Lemma 4.1.10** *We have the following dependencies between* $\mathcal{SR}$, $\mathcal{RSR}$, *and* $\mathcal{LR}$:

1. *If* $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$, *then* $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$.
2. *If* $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, *then* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$.

**Proof:** We prove the two assertions by joint induction on the structure of $\tau(\mathbb{A})$. If $\mathbb{A} \in \mathcal{S}^{nf}$, then the claim is trivial, since the relations $\mathcal{LR}$ and $\mathcal{SR}$ coincide for non-functional sorts. So let $\mathbb{A} \in \mathcal{S}^{f}$.

For the first assertion let $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$, $\Gamma' || \Gamma$, and $\Gamma' \vdash_\Sigma \mathcal{LR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$. Then by the second inductive hypothesis we have $\Gamma' \vdash_\Sigma \mathcal{SR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$ and thus $\Gamma', \Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$ by 4.1.5, so by the first inductive hypothesis we have $\Gamma', \Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$, which gives $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$ by definition of $\mathcal{LR}$.

For the inductive case of the second assertion let $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$ and $Y \notin \mathbf{Dom}(\Gamma)$ be a variable, clearly the $\Sigma\Lambda$-proof for $[Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma Y{::}\mathfrak{d}(\mathbb{A})$, that consists of a single *ws:var*-node, is in restricted semi-structural form, so wee have $[Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{RSR}(Y; \mathfrak{d}(\mathbb{A}))$ and $[Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{LR}(Y; \mathfrak{d}(\mathbb{A}))$ by the first inductive hypothesis. Thus by definition of $\mathcal{LR}$ we see that $\Gamma, [Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{LR}(\mathbf{A}Y; \mathfrak{r}(\mathbb{A}))$ and get $\Gamma, [Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{SR}(\mathbf{A}Y; \mathfrak{r}(\mathbb{A}))$ by the second inductive hypothesis. By lemma 4.1.6 we have $\Gamma \vdash_\Sigma \mathcal{SR}(\lambda Y_{\mathfrak{d}(\mathbb{A})}.\mathbf{A}Y; \mathbb{A})$, thus we have completed the second assertion with 4.1.4, since $Y \notin \mathbf{Free}(\mathbf{A})$. □

**Lemma 4.1.11** *If* $\Xi, \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *and* $\Gamma \vdash_\Sigma \mathcal{LR}(\theta; \Delta)$*, then* $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$.

**Proof:** By induction over the proof $\mathcal{D}{:}\, \Gamma \vdash_\Sigma \mathbf{A}{:}\,\mathbb{A}$:

*ws:var* Here $\mathbf{A}$ is a variable and $\Delta(\mathbf{A}) = \mathbb{A}$ or $\Xi(\mathbf{A}) = \mathbb{A}$. In the first case $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$, since we have required that $\Gamma \vdash_\Sigma \mathcal{LR}(\theta; \Delta)$. In the second case $\theta(\mathbf{A}) = \mathbf{A}$ and $\Xi, \Delta \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$, since the ΣΛ-derivation $\Xi, \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ which consists of a single *ws:var* node, is in restricted semi-structural form. Thus we obtain the assertion by 4.1.10.

*ws:td* We have $\mathcal{SR}(\theta; \Delta)$ by 4.1.10, so there is a semi-structural proof of $\mathcal{E}{:}\, \Gamma \vdash_\Sigma \theta{::}\Delta$, thus the ΣΛ-derivation

$$\dfrac{\dfrac{[\forall\Delta'.\mathbf{A}{::}\mathbb{A}] \in \Sigma}{\Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\,ws{:}td \qquad \dfrac{\mathcal{E}}{\Gamma \vdash_\Sigma \theta{::}\Delta}}{\Xi, \Delta \vdash_\Sigma \theta(\mathbf{A}){::}\mathbb{A}}\,ws{:}inst$$

is in restricted semi-structural form. Therefore we obtain the assertion in the *ws:td*-case by the first assertion in 4.1.10.

*ws:abs* In this case $\mathbf{A} = (\lambda X.\mathbf{C})$, $\mathbb{A} = \mathbb{B} \to \mathbb{C}$ and $\mathcal{D}$ ends in

$$\dfrac{\Xi, \Delta, [X{::}\mathbb{B}] \vdash_\Sigma \mathbf{C}{::}\mathbb{C}}{\Xi, \Delta \vdash_\Sigma \lambda X_{\mathbb{B}}.\mathbf{C}{::}\mathbb{B} \to \mathbb{C}}\,ws{:}abs$$

Let $\Gamma' \vdash_\Sigma \mathcal{LR}(\mathbf{B}; \mathbb{B})$ such that $\Gamma'||\Gamma$ and $\theta' := \theta, [\mathbf{B}/X]$, then we have $\Gamma', \Gamma \vdash_\Sigma \mathcal{LR}(\theta'; \Delta, [X{::}\mathbb{B}])$ and $\Gamma', \Gamma \vdash_\Sigma \theta(\lambda X_{\mathbb{B}}.\mathbf{C})\mathbf{B} \to^h_\beta \theta([\mathbf{B}/X]\mathbf{C}) \doteq \theta'(\mathbf{C})$. By inductive hypothesis we get $\Gamma', \Gamma \vdash_\Sigma \mathcal{LR}(\theta'(\mathbf{C}); \mathbb{C})$ and therefore $\Gamma \vdash_\Sigma \mathcal{LR}(\theta(\lambda X_{\mathbb{B}}.\mathbf{C})\mathbf{B}; \mathbb{C})$ by 4.1.9, so by definition of $\mathcal{LR}$ $\Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); {::}\mathbb{A})$.

*ws:app* In this case $\mathbf{A} \doteq \mathbf{BC}$, $\mathbb{A} = \mathfrak{r}(\mathbb{B})$ and $\mathcal{D}$ ends in

$$\dfrac{\Xi', \Delta' \vdash_\Sigma \mathbf{B}{::}\mathbb{B} \quad \Xi'', \Delta'' \vdash_\Sigma \mathbf{C}{::}\mathfrak{d}(\mathbb{B}) \quad \Xi', \Delta'||\Xi'', \Delta''}{\Xi, \Delta \vdash_\Sigma \mathbf{BC}{::}\mathfrak{r}(\mathbb{B})}\,ws{:}app$$

By inductive hypothesis we have $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{B}); \mathbb{B})$ and $\Xi', \Gamma' \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{C}); \mathfrak{d}(\mathbb{B}))$. Furthermore, $\mathbb{B} \in \mathcal{S}^f$, thus we obtain $\Xi', \Gamma', \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{BC}); \mathfrak{r}(\mathbb{B}))$ by definition of $\mathcal{LR}$.

*ws:βη* Here, we have the following situation:

$$\dfrac{\dfrac{\mathcal{A}}{\Xi, \Delta \vdash_\Sigma \mathbf{P}{::}\mathbb{A}} \quad \Xi, \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{B} \quad \Xi, \Delta \vdash_\Sigma \mathbf{P} =_{\beta\eta} \mathbf{A}}{\Xi, \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\,ws{:}\beta\eta$$

By inductive hypothesis for $\mathcal{A}$ we have that $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{B}); \mathbb{A})$, so with 4.1.9 we obtain $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$, since $\Xi, \Gamma \vdash_\Sigma \theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$.

□

Finally, we have obtained all the partial results. we need to assemble the proof of the structure theorem.

**Corollary 4.1.12 (Structure Theorem)** *If* $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, *then* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$.

**Proof:** Let $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, by lemma 4.1.11 where we take $\theta$ to be the identity substitution we see that $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A};\mathbb{A})$, and with lemma 4.1.10 we obtain the assertion. □

Note that the structure theorem does not make any claim about the uniqueness of the semi-structural ΣΛ-derivation it guarantees. In fact, as the following example shows, there may be several, which even use different term declarations.

**Example 4.1.13** Let $\Sigma := \{[\forall[X::\mathbb{C}].X::\mathbb{A}], [(\lambda X_{\mathbb{A}}.X)::\mathbb{B}\to\mathbb{A}], [a::\mathbb{B}], [a::\mathbb{C}]\}$, then

$$\cfrac{\cfrac{[\forall[X::\mathbb{C}].X::\mathbb{A}]\in\Sigma}{[X::\mathbb{C}]\vdash_\Sigma X::\mathbb{A}}\,ws{:}td \qquad \cfrac{\cfrac{}{\vdash_\Sigma \emptyset::\emptyset}\,wsub{:}start \quad \cfrac{[a::\mathbb{C}]\in\Sigma}{\vdash_\Sigma a::\mathbb{C}}\,ws{:}td}{\vdash_\Sigma [a/X]::[X::\mathbb{C}]}\,wsub{:}ext}{\Gamma\vdash_\Sigma a::\mathbb{A}}\,ws{:}inst$$

and

$$\cfrac{\cfrac{\cfrac{[a::\mathbb{B}]\in\Sigma}{\vdash_\Sigma a::\mathbb{B}}\,ws{:}td \qquad \cfrac{[(\lambda X.X)::\mathbb{B}\to\mathbb{A}]\in\Sigma}{\vdash_\Sigma (\lambda X.X)::\mathbb{B}\to\mathbb{A}}\,ws{:}td}{\vdash_\Sigma (\lambda X.X)a::\mathbb{A}}\,ws{:}app}{\vdash_\Sigma a::\mathbb{A}}\,ws{:}\beta\eta$$

are semi-structural ΣΛ-derivations for $\Gamma \vdash_\Sigma a::\mathbb{A}$.

## 4.2 General Bindings

One of the key steps in sort computation and Σ-unification consists in solving the following problem: given a sort $\mathbb{A}$ and an atom $\mathbf{C}$, find the most general well-sorted formula of sort $\mathbb{A}$ that has head $\mathbf{C}$. Such formulae are called general bindings (cf. 4.2.2) of sort $\mathbb{A}$ for the head $\mathbf{C}$.

**Example 4.2.1** In ΣΛ this problem requires a more careful investigation than in Λ. Consider, for instance, the following signature,

$$\Sigma := \{[a::\mathbb{A}], [b::\mathbb{B}], [f::(\mathbb{B}\to\mathbb{B}\to\mathbb{B})], [\forall[X::\mathbb{B}].(faX)::\mathbb{A}], [\forall[X::\mathbb{B}].(fXb)::\mathbb{A}]\}$$

If Γ be a context with $\Gamma(Z) = \mathbb{B}\to\mathbb{B}$, $\Gamma(X) = \Gamma(Y) = \mathbb{B}$, then the most general formulae with the head $f$ and sort

- $\mathbb{B}$ is $fXY$
- $\mathbb{A}$ are $faX$ and $fXb$

- $(\mathbb{B} \to \mathbb{A})$ are $\lambda X_{\mathbb{B}}.fa(ZX)$ and $\lambda X_{\mathbb{B}}.f(ZX)b$.

Note that both pairs of solutions are incomparable by the instantiation ordering $\preceq$.

Now we formally define general bindings.

**Definition 4.2.2 (General Binding)** Let $\Gamma$ be a variable context, and $\mathbb{A}$ and $\mathbb{B}$ be sorts with $\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})$, where $l = \mathbf{ln}(\mathbb{A})$ and $m = l + \mathbf{ln}(\tau(\mathbb{B})) - \mathbf{ln}(\tau(\mathbb{A})) \geq 0$. Furthermore let $\mathcal{C}$ be the variable context

$$\mathcal{C} := [H^1{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^1(\mathbb{B})], \ldots, [H^m{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^m(\mathbb{B})]$$

Then the formula

$$\mathbf{G} := (\lambda X^1_{\mathfrak{d}^1(\mathbb{A})} \ldots X^l_{\mathfrak{d}^l(\mathbb{A})}.\mathbf{K}\mathbf{V}^1 \ldots \mathbf{V}^m)$$

is called a **general binding of sort** $\mathbb{A}$, if $\mathbf{V}^i = (H^i X^1 \ldots X^l)$, where $H^i$ are variables not in $\mathbf{Dom}(\Gamma)$, and for $\mathbf{K}$ one of the following holds:

1. $\mathbf{K} = X^j$ and $\mathbb{B} = \mathfrak{d}^j(\mathbb{A})$.
2. $\mathbf{K} \in \mathbf{Dom}(\Gamma)$ and $\Gamma(\mathbf{K}) = \mathbb{B}$.
3. $\mathbf{K} \doteq \overline{[\mathbf{W}^n/Y^n]}\mathbf{B}$, where
   (a) there is a term declaration $[\forall[Y^1{::}\mathbb{C}^1], \ldots, [Y^n{::}\mathbb{C}^n].\mathbf{B}{::}\mathbb{B}] \in \Sigma$
   (b) $\mathbf{W}^i := (K^i X^1 \ldots X^l)$
   (c) $K^i$ are variables not in $\mathbf{Dom}(\Gamma)$ chosen distinct from the $H^i$.

   In this case we have to augment the context $\mathcal{C}$ by the variable declarations

   $$[K^1{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathbb{C}^1], \ldots, [K^n{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathbb{C}^n]$$

   for the variables $K^i$.

We call $\mathcal{C}$ the **context of variables introduced for G**. We now characterize **G** by the possible cases for $\mathbf{head}(\mathbf{G})$.

- If **G** is flexible, then we call **G** a **general weakening binding of sort** $\mathbb{A}$.
- If **G** is rigid and $\mathbf{head}(\mathbf{A})$ is constant, then we call **G** an **imitation binding**.
- If $\mathbf{head}(\mathbf{G}) = j$ (recall our convention in 2.3.20), then we call **G** a $j$**-projection bindings** of sort $\mathbb{A}$.

We denote the set of all general bindings of sort $\mathbb{A}$, head $h$, and introduced context $\mathcal{C}$ by $\mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$ and that of all weakening bindings of sort $\mathbb{A}$ by $\mathcal{G}^w_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$. Note that since valid signatures are finite, the set of all weakening bindings is also finite for a given sort.

Finally, we need the set $\mathcal{A}^h_{\mathbb{A}}(\Sigma, \Delta, \mathcal{C})$ of general bindings of sort $\mathbb{A}$ that **approximate** a given head $h$ by projection, imitation, or weakening. We define this set by

$$\mathcal{A}^h_{\mathbb{A}}(\Sigma, \Delta, \mathcal{C}) := \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Delta, \mathcal{C}) \cup \mathcal{G}^j_{\mathbb{A}}(\Sigma, \Delta, \mathcal{C}) \cup \mathcal{G}^w_{\mathbb{A}}(\Sigma, \Delta, \mathcal{C})$$

The definition of general bindings closely corresponds to the definition of semi-structural $\Sigma\Lambda$-derivations. In particular, we have two possible forms for general bindings, the first (classical) one obtains the sort information from the head variable, whereas the second one obtains the sort information from a term declaration. In $\Lambda$ we only have the first form, since we do not have term declarations. Consequently, general bindings are unique up to the choice of new variables and consist only of the head and of variables. In $\Sigma\Lambda$ each term declaration, that has the appropriate head and meets certain conditions contributes a general binding.

**Lemma 4.2.3** *If* $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$, *then* $\Gamma, \mathcal{C} \vdash_\Sigma \mathbf{G}{::}\mathbb{A}$ *and* $\mathbf{head}(\mathbf{G}) = h$.

**Proof:** We use the notation of definition 4.2.2. Let $\mathbf{G} = (\lambda X^1 \ldots X^l.\mathbf{K}\mathbf{V}^1 \ldots \mathbf{V}^m)$ and $\Xi := [X^1{::}\mathfrak{d}^1(\mathbb{A})], \ldots, [X^l{::}\mathfrak{d}^l(\mathbb{A})]$, then we have

$$\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Xi, \mathcal{C}, \Gamma \vdash_\Sigma \mathbf{K}{::}\mathbb{B}} \qquad \dfrac{\mathcal{C} \vdash_\Sigma H^i{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^i(\mathbb{B}) \quad \Xi \vdash_\Sigma X^j{::}\mathfrak{d}^j(\mathbb{A})}{\Xi, \mathcal{C} \vdash_\Sigma \mathbf{V}^i{::}\mathfrak{d}^i(\mathbb{B})}\, ws{:}app^l}{\Xi, \mathcal{C}, \Gamma \vdash_\Sigma \mathbf{K}\overline{\mathbf{V}^m}{::}\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})}\, ws{:}app^m}{\Gamma, \mathcal{C} \vdash_\Sigma \lambda\overline{X^l}.\mathbf{K}\overline{\mathbf{V}^m}{::}\mathfrak{r}^l(\mathbb{A})}\, ws{:}abs^l$$

where $\mathcal{H}$ is one of the following $\Sigma\Lambda$-derivations.

$$\dfrac{\mathcal{C} \cup \Gamma(\mathbf{K}) = \mathbb{B}}{\mathcal{C}, \Gamma \vdash_\Sigma \mathbf{K}{::}\mathbb{B}}\, ws{:}var \qquad \dfrac{\Xi(X^k) = \mathfrak{d}^k(\mathbb{A})}{\Xi, \Gamma \vdash_\Sigma X^k{::}\mathfrak{d}^k(\mathbb{A}) = \mathbb{B}}\, ws{:}var$$

$$\dfrac{\dfrac{[\forall\overline{[Y^n{::}\mathbb{C}^n]}.\mathbf{B}{::}\mathbb{B}] \in \Sigma}{\overline{[Y^n{::}\mathbb{C}^n]} \vdash_\Sigma \mathbf{B}{::}\mathbb{B}}\, ws{:}td \qquad \dfrac{\dfrac{\mathcal{C} \vdash_\Sigma K^i{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathbb{C}^i \qquad \Xi \vdash_\Sigma X^j{::}\mathfrak{d}^j(\mathbb{A})}{\Xi, \mathcal{C} \vdash_\Sigma \mathbf{W}^i{::}\mathbb{C}^i}\, ws{:}app^l}{\Xi, \mathcal{C} \vdash_\Sigma \overline{[\mathbf{W}^n/Y^n]}{::}\overline{[Y^n{::}\mathbb{C}^n]}}\, wsub{:}ext^n}{\Xi, \mathcal{C} \vdash_\Sigma \mathbf{K}{::}\mathbb{B}}\, ws{:}subst$$

Thus we have verified the assertion. □

**Theorem 4.2.4 (General Binding Theorem)** *Let* $\mathbf{A}$ *be a well-sorted formula with* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *and* $\mathbf{head}(\mathbf{A}) = h$, *then there exists a general binding* $\mathbf{G} \in \mathcal{A}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$ *and a* $\Sigma$*-substitution* $\rho$ *such that* $\Gamma \vdash_\Sigma \rho{::}\mathcal{C}$ *and* $\mathcal{C}, \Gamma \vdash_\Sigma \rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$. *Let* $\mathcal{A}{:}\, \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *be a semi-structural* $\Sigma\Lambda$*-derivation, then there are semi-structural* $\Sigma\Lambda$*-derivations* $\mathcal{R}^i$ *that witness* $\Gamma \vdash_\Sigma \mathcal{SR}(\rho; \mathcal{C})$ *with* $\mathbf{dp}(\mathcal{R}^i) < \mathbf{dp}(\mathcal{A})$.

**Proof:** Let $\mathcal{A}: \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ be the semi-structural $\Sigma\Lambda$-derivation assumed in the assertion, then $\mathcal{A}$ is of the form

$$\cfrac{\cfrac{\cfrac{\cfrac{\mathcal{H}}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}::\mathbb{B}} \qquad \cfrac{\mathcal{D}^i}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{B})}}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}::\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})}\,ws{:}app^m}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}}\,ws{:}abs^l \qquad * \quad *}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\,ws{:}\beta\eta$$

Let $\mathbf{G} = (\lambda X^1_{\mathfrak{d}^1(\mathbb{A})} \ldots X^l_{\mathfrak{d}^l(\mathbb{A})}.\mathbf{K}\mathbf{V}^1 \ldots \mathbf{V}^m)$ be the general binding as defined in 4.2.2. The two possibilities for $\mathbf{H}$ and $\mathcal{H}$ give us the two possibilities in the choice of $\mathbf{K}$.

1. $\Gamma, \Xi^l(\mathbf{H}) = \mathbb{B}$ and $\mathcal{H}$ consists of a single *ws:var* step. In this case let $\mathbf{K} = \mathbf{H}$ and $\mathcal{C}$ be just as defined in 4.2.2 and furthermore

$$\rho := [\lambda\overline{X^l}.\mathbf{D}^1/H^1], \ldots, [\lambda\overline{X^l}.\mathbf{D}^m/H^m]$$

   so $\Gamma \vdash_\Sigma \rho(\mathbf{G}) \doteq \lambda\overline{X^l}.\rho(\mathbf{K})\overline{\rho(\mathbf{V}^m)} =_{\beta\eta} \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}$, and thus $\mathbf{A}$ is really a $\beta\eta$-instance of $\mathbf{G}$.

2. $\mathbf{H} = \theta(\mathbf{B})$ for some term declaration $[\forall\Delta.\mathbf{B}::\mathbb{B}] \in \Sigma$ with $\Delta = \overline{[Y^n::\mathbb{C}^n]}$ and some substitution $\theta$. In this case $\mathcal{H}$ is a $\Sigma\Lambda$-derivation of the form

$$\cfrac{\cfrac{[\forall\Delta.\mathbf{B}::\mathbb{B}] \in \Sigma}{\Delta \vdash_\Sigma \mathbf{B}::\mathbb{B}}\,ws{:}td \qquad \cfrac{\mathcal{E}}{\Gamma, \Xi^l \vdash_\Sigma \theta::\Delta}}{\Gamma, \Xi^l \vdash_\Sigma \theta(\mathbf{B})::\mathbb{B}}\,ws{:}inst$$

   In this case let $\mathbf{K} = \overline{[\mathbf{W}^n/Y^n]}\mathbf{B}$, and $\mathcal{C}$ be defined as in 4.2.2, and

$$\rho := [\lambda\overline{X^l}.\theta(Y^1)/K^1], \ldots, [\lambda\overline{X^l}.\theta(Y^t)/K^t], [\lambda\overline{X^l}.\mathbf{D}^1/H^1], \ldots, [\lambda\overline{X^l}.\mathbf{D}^m/H^m]$$

   We can easily verify that $\Gamma, \Xi^l \vdash_\Sigma \rho::\mathcal{C}, \mathcal{C}'$. Now $\mathbf{W}^i \doteq K^i\overline{X^l}$, so $\rho(\mathbf{W}^i) = \theta(Y^i)$ and $\rho(\mathbf{K}) = \rho(\overline{[\mathbf{W}^t/Y^t]}\mathbf{B}) = \theta(\mathbf{B})$, therefore $\Gamma \vdash_\Sigma \rho(\mathbf{G}) =_{\beta\eta} (\lambda\overline{X^l}.\theta(\mathbf{B})\overline{\mathbf{D}^m}) =_{\beta\eta} \mathbf{A}$.

If we apply lemma 4.1.6 $l$ times to $\mathcal{D}^i$, then we obtain semi-structural $\Sigma\Lambda$-derivations $\widetilde{\mathcal{D}}^i: \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{D}^i::\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^i(\mathbb{B})$. Moreover $\mathbf{dp}(\widetilde{\mathcal{D}}^i) = \mathbf{dp}(\mathcal{D}^i) + l < \mathbf{dp}(\mathcal{A})$, since there are $l$ *ws:abs* nodes in $\mathcal{A}$ below $\mathcal{D}^i$. The same argument holds for the $\mathcal{E}^i$. Collecting the $\mathcal{D}^i$ and possibly the $\mathcal{E}^i$ gives us the semi-structural $\Sigma\Lambda$-derivations $\mathcal{R}^i$ with the appropriate depth conditions. □

**Remark 4.2.5** Let $\Sigma$ be a trivially sorted signature, then $\Sigma = \{[c::\sharp(\alpha)] \mid c \in \overline{\Sigma}_\alpha\}$, so general bindings of head $h$ and sort $\sharp(\alpha)$ have the form

$$\mathbf{G} := \lambda\overline{X_{\mathfrak{d}^n(\sharp(\alpha))}}.h(H^1\overline{X^n}) \ldots (H^m\overline{X^n})$$

if $h \in \mathcal{V}_\beta \cup \overline{\Sigma}_\beta$ where $\mathbf{ln}(\flat(\beta)) = m$ and the $H^i$ are new variables of sort $\overline{\mathfrak{d}^n(\sharp(\alpha))} \to \mathfrak{d}^i(\sharp(\beta))$. Note that this is just (up to $\eta$-equality) Huet's definition of a partial binding (cf. [Sny91]).

**Remark 4.2.6** Note that for an implementation of Σ-unification, the set $\mathcal{G}^h_{\mathbb{A}}(\Sigma,\Gamma,\mathcal{C})$ is not an optimal set of general bindings. We have only concentrated on giving a complete set of general bindings. In particular, the subsort relation is not integrated into the concept of general binding leading to some redundancy in the search for unifiers. A concrete implementation of ΣΛ would take care to rule out this redundancy.

## 4.3 Sort Computation

In this subsection we analyze the sort computation problem. It will turn out, that the sort computation problem for arbitrary signatures and the higher-order matching problem are interreducible. Unfortunately, the latter is only known to be decidable in subcases [Hue76, Dow92, Wol93], and the issue is still open for the general case.

**Definition 4.3.1** Let $\mathbf{A} \in \mathit{wff}_\alpha(\Omega)$ be a formula in $\beta$-normal form, then we define the sort system $\mathcal{S}^{\mathbf{A}}$ by

$$\begin{array}{rcl} \mathcal{BS}^{\mathbf{A}} & := & \sharp(\mathcal{BT}) \cup \{\mathbb{A}\} \\ \tau(\mathbb{A}) & := & \alpha \\ \mathfrak{d}(\mathbb{A}) & := & \mathfrak{d}(\sharp(\alpha)) \\ \mathfrak{r}(\mathbb{A}) & := & \mathfrak{r}(\sharp(\alpha)) \end{array}$$

Note that we have $\mathbb{A}$ **Rdom** $\sharp(\alpha)$, since $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\sharp(\alpha))$ and $\mathfrak{r}(\mathbb{A}) = \mathfrak{r}(\sharp(\alpha))$.

Let $W := \{X^1_{\beta_1},\ldots,X^n_{\beta_n}\}$ be a finite set of variables, then we can define a signature $\Sigma^{\mathbf{A}}$ by choosing $\overline{\Sigma^{\mathbf{A}}} := \Omega$ and $\Sigma := \Sigma^\sharp, [\forall W^\sharp.\sharp(\mathbf{A})::\mathbb{A}]$. From lemma 3.3.11 we know that $\Sigma^\sharp$ is a valid signature and $W^\sharp \vdash_{\Sigma^\sharp} \sharp(\mathbf{A})::\sharp(\alpha)$, so $\Sigma^{\mathbf{A}}$ is a valid signature by $\mathit{sig}-\mathit{td}$.

**Lemma 4.3.2** *Let $\mathbf{B} \in \mathit{wff}_\alpha(\Omega)$ be an arbitrary formula, then the following assertions are equivalent:*

1. *The judgment $\mathbf{Free}(\mathbf{B})^\sharp \vdash_{\Sigma^{\mathbf{A}}} \mathbf{B}::\mathbb{A}$ is provable in ΣΛ, but $\mathbf{Free}(\mathbf{B})^\sharp \vdash_{\Sigma^\sharp} \mathbf{B}::\mathbb{A}$ is not.*
2. *There exists a substitution $\rho \in \mathbf{SUB}(\Omega)$ such that $\rho(\mathbf{B}) =_{\beta\eta} \mathbf{A}$.*

**Proof:** Let assertion 1. be valid, then by a close inspection we observe that there is only one general binding of head $\mathbf{head}(\mathbf{B})$ and sort $\mathbb{A}$, namely, $\sharp(\mathbf{A})$ itself, since we have chosen $\mathbb{A}$ to be a base sort, which has length zero. Thus we get the sufficiency direction by the general binding lemma 4.2.4.

The sets $\mathit{wff}_\alpha(\overline{\Sigma}, W)$ are isomorphic to the $\mathit{wsf}_{\sharp(\alpha)}(\Sigma^\sharp, W^\sharp)$ and even to $\mathit{wsf}_{\sharp(\alpha)}(\Sigma^{\mathbf{A}}, W^\sharp)$, since the new term declaration cannot result in any new judgments $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$, where $\mathbb{B}$ does not contain $\mathbb{A}$. So the substitution $\sharp(\rho)$ is also a Σ-substitution, and therefore we obtain the necessitation direction by *ws:inst*. □

**Definition 4.3.3 (Σ-Matching)** Let $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$, such that $\mathbb{A}$ **Rdom** $\mathbb{B}$, then a Σ-substitution $\theta$ such that $\Gamma \vdash_\Sigma \theta::\Delta$ and $\Gamma \vdash_\Sigma \theta(\mathbf{A}) =_{\beta\eta} \mathbf{B}$ is called the **Σ-matcher of A to B**.

We call the problem of finding Σ-matchers for given formulae the **Σ-matching problem**. The matching problem for Λ is called the **higher-order matching problem**. It is known to be decidable for higher-order patterns [Mil92] and formulae of order less than tree [Hue76, Dow92].

In fact, the general binding theorem 4.2.4 can be read as an algorithm for sort computation using higher-order matching:

**Theorem 4.3.4** *Sort computation in* $\Sigma\Lambda$ *can be reduced to higher-order matching.*

**Proof:** We first note that for a well-typed formula $\mathbf{A}_\alpha$ there is a $\Sigma\Lambda$-proof of $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, iff $\mathbf{A}$ is the $\Sigma$-instance of a general binding of sort $\mathbb{A}$. Indeed if $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, then by the general binding lemma 4.2.4, there is a general binding $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$ and a substitution $\rho$ such that $\Gamma \vdash_\Sigma \rho{::}\mathcal{C}$ and $\Gamma \vdash_\Sigma \rho(\mathbf{G}){=}_{\beta\eta}\mathbf{A}$. On the other hand, if there exist a well-sorted formula $\mathbf{G}$ of sort $\mathbb{A}$ (the general binding) and a $\Sigma$-substitution $\rho$, then $\rho(\mathbf{G})$ is also of sort $\mathbb{A}$ by 4.2.4.

Now let $\mathbf{Mat}(\flat(\mathbf{A}), \flat(\mathbf{B}))$ be a complete set of higher-order matchers (cf. definition 4.3.3) of well-typed formulae $\flat(\mathbf{A})$ and $\flat(\mathbf{B})$, then the following inference rule together with those for $\Sigma$-substitutions (cf. 3.4.1) give an alternative inference system for well-sortedness:

$$\frac{\Gamma \vdash_\Sigma \sharp(\rho){::}\mathcal{C} \quad \rho \in \mathbf{Mat}(\flat(\mathbf{A}), \flat(\mathbf{G})) \quad \mathbf{G} \in \mathcal{G}_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\ \textit{ws:alt}$$

We now convince ourselves that this inference system terminates modulo higher-order matching. Let $\mathcal{A}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ be in semi-structural form, then the general binding theorem (4.2.4) also gives us semi-structural $\Sigma\Lambda$-derivations $\mathcal{A}^i{:}\,\Gamma \vdash_\Sigma \rho(X^i){::}\mathcal{C}(X^i)$ for all $X^i \in \mathbf{Dom}(\rho)$ such that $\mathbf{dp}(\mathcal{A}^i) < \mathbf{dp}(\mathbf{A})$. Thus, if we consider the set $\mathcal{N}$ of depths of semi-structural $\Sigma\Lambda$-derivations associated with the subgoal judgments $\Gamma \vdash_\Sigma \mathbf{A}^i{::}\mathbb{A}^i$, then each backward application of *ws:alt* with subsequent decomposition of $\Gamma \vdash_\Sigma \sharp(\rho){::}\mathcal{C}$ by *wsub:ext* yields a multiset $\mathcal{N}'$ with $\mathcal{N}' < \mathcal{N}$ where $<$ is the multiset ordering on natural numbers. □

**Corollary 4.3.5** *The problem of sort computation in* $\Sigma\Lambda$ *and the problem of higher-order matching in wff*$(\Omega)$ *are interreducible and therefore equivalent.*

It is not clear whether the higher-order matching problem in typed $\lambda$-calculi is decidable in general. However, there are large subclasses where matching is known to be decidable, for instance, the class of higher-order patterns [Mil92] or that of third-order formulae [Hue76, Dow92]. Restricting term declarations to these or other such classes yield instances of $\Sigma\Lambda$, where the sort computation problem is decidable, which is of course desirable for all practical applications. We conjecture that all term declarations that come up when a user codes a mathematical theory into a sort structure will be simple and the usual matching and unification procedures will terminate on them. Practice in programming languages has shown that humans have great difficulties thinking in terms of functions of order greater than three. Therefore it would probably – for all practical (theorem proving) purposes – not be a significant restriction to restrict term declarations to third order logic.

## 4.4 Σ-Unification Problems

In this subsection we define Σ-unification and present the Σ-unification problems, the basic structures that are manipulated by our Σ-unification algorithms. Originally a higher-order unification problem consists in finding a substitution $\sigma$ such that $\sigma(\mathbf{A}) =_{\beta\eta} \sigma(\mathbf{B})$ for given formulae $\mathbf{A}$ and $\mathbf{B}$. Naturally in $\Sigma\Lambda$ we have to restrict our attention to substitutions $\sigma$ that are well-sorted. Furthermore, as formulae are only well-sorted with respect to a certain variable context, it is advantageous to record this context as part of the problem.

Most unification algorithms solve unification problems by transforming sets of pairs $\mathbf{A} =^? \mathbf{B}$ of formulae to a solved form, from which a solution can be directly read off. The presentation of unification algorithms as inference rules is a variant of the presentation as systems of transformations, which was introduced in [MM73]. Since in this thesis we do not consider higher-order unification for its own sake, but in the context of a higher-order resolution calculus, we have to consider unification problems that are even more general than sets of pairs. In particular, for resolution we need to manipulate quantified formulae, where the sequencing of quantifiers induces dependencies on variables. These dependencies (variables $Y$ that were existentially quantified in the scope of a universal quantifier $\forall X$ may not be free in any formula instantiated for $X$) have to be respected in order to obtain a sound calculus. Note that this dependency is usually coded into the Skolem functions in the first-order case of resolution. Thus our $\Sigma$-unification problems will be built up from a variable context, a variable condition, and a set of pairs. To make this formal, we have to generalize our notion of variable context by marking the variables with labels $+$, $-$, and $0$, in order to distinguish between variables for which we may ($+$, coming from universal variables) or may not ($-$ coming from existential variables) substitute, and those that come from variables that used to be locally bound ($0$).

**Definition 4.4.1 (Annotated Variable Contexts)** Let $\Gamma$ be a typed partial function on $\mathcal{V}_\mathcal{T}$ that associates with each variable a sort $\mathbb{A}$ and an **annotation** $\pm \in \{+, 0, -\}$, then we call $\Gamma$ an **annotated variable context**. To distinguish non-annotated variable contexts from annotated ones we sometimes speak of **proper variable contexts**.

As in the case of usual variable contexts we write $\Gamma$ as a set of **annotated variable declarations** of the form $[X^\pm :: \mathbb{A}]$, if $\Gamma(X) = (\mathbb{A}, \pm)$, and call $\pm$ the **annotation of** $X$ **in** $\Gamma$. If the annotation of $X$ is $+$ ($-$) in $\Gamma$, then we call it **positive** (**negative**), otherwise ($0$) **locally bound**, and we indicate this by annotating $X$ with $\pm$ as in $X^\pm$.

Obviously any annotated variable context can be made into a variable context by projection on the first component, so we can use all of the machinery developed so far. Furthermore, we can obtain a variable context $\Gamma^+$ ($\Gamma^-, \Gamma^0$) from $\Gamma$ by restricting $\Gamma$ to the positive (negative, locally bound) variables.

**Definition 4.4.2 (Variable Condition, $\mathcal{R}_\Gamma$-Substitution)** Let $\Gamma$ be an annotated variable context and $\mathcal{R} \subseteq \mathbf{Dom}(\Gamma^+) \times \mathbf{Dom}(\Gamma^-)$, then $\mathcal{R}$ is called a **variable condition**.

A $\Sigma$-substitution $\sigma$ with $\Gamma^-, \Delta \vdash_\Sigma \sigma :: \Gamma^+$ is called an $\mathcal{R}_\Gamma$**-substitution** assuming $\Gamma$, iff $Y \notin \mathbf{Free}(\sigma(X))$ for all $(X, Y) \in \mathcal{R}$. Thus the intuitive meaning of a pair $(X^+, Y^-)$ in a variable condition $\mathcal{R}$ for $\Gamma$ is that no formula containing $Y^-$ as a free variable can be legally substituted for $X^+$.

For a variable condition $\mathcal{R}$ and an annotated variable context $\Delta$ we define a judgment $\Delta \vdash \overline{\mathcal{R}}(X^+, \mathbf{A})$, called the **associated substitution condition** $\overline{\mathcal{R}}$ to hold on $X^+$ and $\mathbf{A}$, iff

1. $\Delta, \Gamma \vdash_\Sigma \mathbf{A} :: \Gamma^+(X)$,

2. $X \notin \mathbf{Free}(\mathbf{A})$,
3. no variable $Y \in \mathbf{Free}(\mathbf{A})$ is an $\mathcal{R}$-image of $X^+$, $(\{X^+\} \times \mathbf{Free}(\mathbf{A}) \cap \mathcal{R} = \emptyset)$.

Thus we can rephrase the condition on $\mathcal{R}_\Gamma$-substitutions as $\Delta \vdash_\Sigma \overline{\mathcal{R}}(X, \sigma(X))$ for all $X \in \mathbf{Dom}(\sigma)$. For a given variable condition $\mathcal{R}$ for $\Gamma$ and an $\mathcal{R}_\Gamma$-substitution $\sigma$ we will often need the following variable condition

$$\mathcal{R}(\sigma) := \{(Z, W) \in \mathcal{R} \mid Z \notin \mathbf{Dom}(\sigma)\} \cup \{(Z, W) \mid Z \in \mathbf{Intro}(\sigma), \mathcal{R}(X, W), X \in \mathbf{Dom}(\sigma)\}$$

for $\Gamma$. In most applications $\sigma$ only consists of one pair $[\mathbf{A}/X]$, in this case we write $\mathcal{R}(\sigma)$ as $\mathcal{R}(\mathbf{A}/X)$.

**Definition 4.4.3 (More General)** Let $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}$, then we say that $\mathbf{A}$ is **more general than B**, iff there exists a $\Sigma$-substitution $\sigma \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$ such that $\Gamma \vdash_\Sigma \sigma(\mathbf{A}) =_{\beta\eta} \mathbf{B}$. In this case we call $\mathbf{B}$ an **instance of A**, and denote this fact by $\Gamma \vdash_\Sigma \mathbf{A} \preceq \mathbf{B}$. Note that the resulting **instance relation** $\preceq$ is a partial ordering relation on well-sorted formulae.

**Definition 4.4.4** Let $\sigma$ and $\theta$ be $\Sigma$-substitutions such that $\Gamma \vdash_\Sigma \sigma::\Delta$, $\Gamma \vdash_\Sigma \theta::\Delta'$, $\Delta || \Delta'$, $\Gamma || \Gamma'$, and $\Xi \subseteq \mathbf{Dom}(\Delta) \cap \mathbf{Dom}(\Delta')$, then

- $\sigma$ and $\theta$ are **equal over** $\Xi$ ($\sigma = \theta[\Xi]$), iff for all $X \in \mathbf{Dom}(\Xi)$ we have $\sigma(X) = \theta(X)$.
- $\sigma$ and $\theta$ are **$\beta\eta$-equal over** $\Xi$ ($\Gamma \vdash_\Sigma \sigma =_{\beta\eta} \theta[\Xi]$), iff for all $X \in \mathbf{Dom}(\Xi)$ we have $\Gamma, \Gamma' \vdash_\Sigma \sigma(X) =_{\beta\eta} \theta(X)$.
- $\sigma$ is **more general than** $\theta$ **over** $\Xi$ ($\Gamma \vdash_\Sigma \sigma \preceq_{\beta\eta} \theta[\Xi]$), iff there is a substitution $\rho \in \mathbf{wsSub}(\Sigma, \Gamma' \to \Gamma)$ such that $\Gamma, \Gamma' \vdash_\Sigma \theta =_{\beta\eta} \rho \circ \sigma[\Xi]$.

If $\sigma$ and $\theta$ are $\Sigma$-substitutions such that $\mathbf{Dom}(\sigma) \subseteq \mathbf{Dom}(\theta)$ and $\sigma \preceq_{\beta\eta} \theta[\mathbf{Dom}(\sigma)]$, then we call $\sigma$ an **approximation** of $\theta$.

To ease the load of notation we denote the judgment $\Gamma \vdash_\Sigma \sigma =_{\beta\eta} \theta[\mathbf{Dom}(\Gamma^+)]$ by $\Gamma \vdash_\Sigma \sigma =_{\beta\eta} \theta[\mathcal{E}]$ for any equational problem $\mathcal{E} = \langle \Gamma : \mathcal{R} \rangle . \mathcal{E}'$ and we sometimes even drop the $[\Xi]$ in $\Gamma \vdash_\Sigma \sigma \preceq \theta[\Xi]$, if it is clear from the context.

**Definition 4.4.5 ($\Sigma$-Unification Problem)** Let $\Gamma$ be an annotated variable context and $\mathcal{R}$ a variable condition for $\Gamma$. Then we call a triple $\langle \Gamma : \mathcal{R} \rangle . \mathbf{C}$ a **$\Sigma$-unification problem**, iff $\mathbf{C}$ is of the form $\mathbf{C} \doteq \mathbf{P}_1 \wedge \ldots \wedge \mathbf{P}_n$, where the so-called **pairs** $\mathbf{P}_i$ are of the form $\mathbf{P}_i = (\mathbf{A}_i =^? \mathbf{B}_i)$ for some $\mathbf{A}_i, \mathbf{B}_i \in \mathit{wsf}(\Sigma, \Gamma)$ or $\mathbf{P}_i = \top_o$. We call a formula $\mathbf{A}$ in $\mathbf{A} =^? \mathbf{B}$ **flexible**, iff $\mathbf{head}(\mathbf{A}) \in \mathbf{Dom}(\Gamma^+)$ and **rigid** otherwise. Huet's classification of pairs into the categories *rigid/rigid*, *flex/rigid*, and *flex/flex* will play a great role in our analysis of $\Sigma$-unification. Since each $\Sigma$-unification problem $\mathcal{E} = \langle \Gamma : \mathcal{R} \rangle . \mathbf{C}$ determines a unique variable condition $\mathcal{R}_\Gamma$, we say that $\sigma$ is an $\mathcal{E}$-substitution, iff $\sigma$ is an $\mathcal{R}_\Gamma$-substitution.

**Remark 4.4.6** Note that our $\Sigma$-unification problems are generalizations of Miller's unification problems with a mixed prefix from [Mil91], since the mode of the quantifications in a prefix can be coded into our annotations, and the sequencing in Miller's prefix naturally leads to a variable condition.

**Definition 4.4.7 ($\Sigma$-Unifier)** Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{F}$ be a $\Sigma$-unification problem, then we call an $\mathcal{R}_\Gamma$-substitution $\sigma$ a **$\Sigma$-unifier of** $\mathcal{E}$, if $\sigma$ **solves all pairs in** $\mathcal{F}$, i.e. $\Delta \vdash_\Sigma \sigma(\mathbf{A}) =_{\beta\eta} \sigma(\mathbf{B})$ for all pairs $\mathbf{A} =^? \mathbf{B} \in \mathcal{F}$. We call a $\Sigma$-unification problem $\mathcal{E}$ **$\Sigma$-unifiable**, iff there is a $\Sigma$-unifier for $\mathcal{E}$, and we denote the set of $\Sigma$-unifiers of a $\Sigma$-unification problem $\mathcal{E}$ with $\mathbf{wsU}(\Sigma, \mathcal{E})$.

**General Assumption 4.4.8 ($\alpha$-Conversion for $\Sigma$-Unification Problems)** Let $\Gamma = \Delta, [X^+ :: \mathbb{A}]$ and $\Gamma' = \Delta, [Y^+ :: \mathbb{A}]$ be annotated variable contexts, and let $\mathcal{R}$ be a variable condition for $\Gamma$. Then $\mathcal{R}' := \mathcal{R}([Y/X])$ is a variable context for $\Gamma'$ and $\mathcal{R}'$, which can be obtained from $\mathcal{R}$ by systematically replacing all occurrences of $X$ in $\mathcal{R}$ by $Y$. Furthermore, we have $\mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}) = \mathbf{wsU}(\Sigma, \langle\Gamma'\colon\mathcal{R}'\rangle.[Y/X]\mathcal{E})$ up to variable renaming.

In the following we will consider the declaration $\langle\Gamma\colon\mathcal{R}\rangle.$ in a $\Sigma$-unification problem as a binder for all variables in $\mathbf{Dom}(\Gamma)$, and we will keep $\alpha$-conversion for $\Sigma$-unification problems implicit, renaming them whenever variable disjointness is required.

Note that $\Sigma$-unifiability does not entail that both formulae of a pair have identical sets of sorts, since these sets may grow as more term declarations become applicable with instantiation. For instance, consider the unification problem $\langle\Gamma\colon\emptyset\rangle.F =^? G$ where $\Gamma(F) = \mathbb{A}$, $\Gamma(G) = \mathbb{B}$, and $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$. Nevertheless, $\Sigma$-unifiable pairs must have the same types, and moreover the sorts must obey the **Rdom** relation.

**Lemma 4.4.9** *If $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\Delta \vdash_\Sigma \mathbf{B}::\mathbb{B}$, and $\mathbf{A}$ and $\mathbf{B}$ are $\Sigma$-unifiable, then $\mathbb{A}$ **Rdom** $\mathbb{B}$.*

**Proof:** Let $\theta$ be a $\Sigma$-unifier for $\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A} =^? \mathbf{B}$, then $\theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$, and we have $\Gamma \vdash_\Sigma \theta(\mathbf{A})::\mathbb{A}$ and $\Gamma \vdash_\Sigma \theta(\mathbf{B})::\mathbb{B}$ by *ws:subst* and $\Gamma \vdash_\Sigma \theta(\mathbf{A})::\mathbb{B}$ by 3.5.1. Now we get the assertion with 3.5.4. □

**Lemma 4.4.10** *If $\Gamma \vdash_\Sigma \mathbf{B} =_{\beta\eta} \mathbf{B}'$, $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A} =^? \mathbf{B} \wedge \mathcal{F}$, and $\mathcal{E}' = \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A} =^? \mathbf{B}' \wedge \mathcal{F}$, then $\mathbf{wsU}(\Sigma, \mathcal{E}) = \mathbf{wsU}(\Sigma, \mathcal{E}')$.*

**Proof:** By definition of $\Sigma$-unifiers. □

**General Assumption 4.4.11** Since we are only interested in $\Sigma$-unification problems, where $\Sigma$-unification does not fail trivially, we assume for all $\Sigma$-unification problems $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A}^1 =^? \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^? \mathbf{B}^n$ that $\Gamma \vdash_\Sigma \mathbf{A}^i::\mathbb{A}^i$, $\Gamma \vdash_\Sigma \mathbf{B}^i::\mathbb{B}^i$, and $\mathbb{A}^i$ **Rdom** $\mathbb{B}^i$ for all $i \leq n$. Note that we can decide whether a well-sorted unification problem meets this condition, because we can compute the argument sorts. Furthermore, we assume all $\mathbf{A}^i$ and $\mathbf{B}^i$ to be in $\beta$-normal form.

**Definition 4.4.12 ($\Sigma$-Solved Form)** Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}'$ be a $\Sigma$-unification problem with $\Gamma^+(X) = \mathbb{A}$. Then we call a pair $X^+ =^? \mathbf{A}$ **$\Sigma$-solved in** $\mathcal{E}$, iff $\Gamma \vdash_\Sigma \overline{\mathcal{R}}(X^+, \mathbf{A})$, and moreover $X^+$ is not free elsewhere in $\mathcal{E}'$.

We call a variable $X$ **$\Sigma$-solved in** $\mathcal{E}$, iff there is a $\Sigma$-solved pair $X =^? \mathbf{A}$ in $\mathcal{E}'$, and we call $\mathcal{E}$ in **$\Sigma$-solved form**, if all of its pairs are $\Sigma$-solved in $\mathcal{E}$. A $\Sigma$-unification problem $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.X^1 =^? \mathbf{A}^1 \wedge \ldots \wedge X^n =^? \mathbf{A}^n$ in $\Sigma$-solved form determines a unique $\mathcal{R}_\Gamma$-substitution

$\sigma_{\mathcal{E}} := [\mathbf{A}^1/X_1], \ldots, [\mathbf{A}^n/X^n]$ that will turn out to be the unique most general $\Sigma$-unifier of $\mathcal{E}$ (cf. 4.4.14). This result justifies the name $\Sigma$-solved.

Note that any $\Sigma$-unification problem $\mathcal{E}$ can always be written as $\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{F} \wedge \mathcal{G}$, where $\mathcal{G}$ is the set of pairs in $\mathcal{E}$ that are $\Sigma$-solved in $\mathcal{E}$. We call $\mathcal{G}$ the **$\Sigma$-solved part of $\mathcal{E}$**, and denote it by $\mathcal{E}_\sigma$, if $\sigma = \sigma_{\mathcal{G}}$ is the $\mathcal{R}_\Gamma$-substitution that corresponds to $\mathcal{G}$.

**Definition 4.4.13 (Complete Set of $\Sigma$-Unifiers)** Let $\mathcal{E}$ be a $\Sigma$-unification problem, then a subset $\Psi \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$ is called a **complete set of $\Sigma$-unifiers of $\mathcal{E}$**, iff for all $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$ there is a $\sigma \in \Psi$ with $\Gamma \vdash_\Sigma \sigma \preceq \theta[\mathcal{E}]$.

If the singleton set $\{\sigma\} \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$ is a complete set of $\Sigma$-unifiers for $\mathcal{E}$, then $\sigma$ is called a **most general $\Sigma$-unifier for $\mathcal{E}$**. We call a complete set $\Phi \in \mathbf{wsU}(\Sigma, \mathcal{E})$ of $\Sigma$-unifiers **minimal** or a **set of most general $\Sigma$-unifiers**, iff any $\sigma, \theta \in \Phi$ are $\preceq$-incomparable, i.e. we do not have $\Gamma \vdash_\Sigma \sigma \preceq \theta[\mathcal{E}]$.

Transformation-based unification methods attempt to reduce the input systems to solved systems, which represent their unifiers. The fundamental connection between solved systems and $\Sigma$-unifiers is the following fact, which shows that solved systems indeed represent their own solutions.

**Lemma 4.4.14** *Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}_\sigma$ be a $\Sigma$-unification problem in $\Sigma$-solved form, then $\sigma$ is a most general $\Sigma$-unifier for $\mathcal{E}$. In particular, for any $\Sigma$-unifier of $\mathcal{E}$ we have $\theta =_{\beta\eta} \theta \circ \sigma[\mathcal{E}]$.*

**Proof:** Clearly $\sigma$ is an $\mathcal{R}_\Gamma$-substitution, since $\mathcal{E}$ is in $\Sigma$-solved form. Moreover $\sigma(X) = \sigma(\mathbf{A})$ for any pair $X =^? \mathbf{A} \in \mathcal{E}_\sigma$, thus $\sigma$ is a $\Sigma$-unifier for $\mathcal{E}$. If $\theta \in \mathbf{wsSub}(\Sigma, \Gamma^+ \to \Delta)$ is a $\Sigma$-unifier for $\mathcal{E}$, then $\Gamma, \Delta \vdash_\Sigma \theta \circ \sigma(X^i) \doteq \theta(\mathbf{A}^i) =_{\beta\eta} \theta(X^i)$ and $\theta(Y) = \theta \circ \sigma(Y)$ for $Y \notin \mathbf{Dom}(\Gamma^+)$, so that indeed $\theta =_{\beta\eta} \theta \circ \sigma$. □

In general however, a $\Sigma$-unification problem $\mathcal{E}$ does not have a single most general $\Sigma$-unifier and may not even have most general $\Sigma$-unifiers at all, even if it is $\Sigma$-unifiable; this behavior is not a particular feature of the $\Sigma\Lambda$ system, since it is well-known that it is already the case for $\Lambda$ [Gou66]. The next lemma will be used to show that we need not be concerned with $\Sigma$-solved pairs, when computing $\Sigma$-unifiers. It is therefore consistent with the intuition that the $\Sigma$-solved part of a system is merely a record of an answer substitution being constructed.

**Lemma 4.4.15** $\mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E} \wedge \mathcal{E}_\sigma) = \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\sigma(\mathcal{E}) \wedge \mathcal{E}_\sigma)$

**Proof:** Clearly we have $\theta \in \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E} \wedge \mathcal{E}_\sigma)$, iff $\theta \in \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}_\sigma) \cap \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E})$, so $\theta = \theta \circ \sigma$ by 4.4.14. Now $\theta \circ \sigma \in \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E})$, iff $\theta \in \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\sigma(\mathcal{E}))$, which gives the assertion. □

**Definition 4.4.16 (Complete $\Sigma$-Unification Procedure)** An inference system $\mathcal{I}$ is called a **complete $\Sigma$-unification procedure**, iff for every $\Sigma$-unification problem $\mathcal{E}$ and each $\Sigma$-substitution $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$ there is a system $\mathcal{F}$ in $\Sigma$-solved form such that $\mathcal{E} \vdash_{\mathcal{I}} \mathcal{F}$ and $\Gamma \vdash_\Sigma \sigma_{\mathcal{F}} \preceq \theta[\mathcal{E}]$.

**Remark 4.4.17** The algorithms for $\Sigma$-unification consist in the process of systematically exploring the search trees generated by the respective inference systems from a root $\mathcal{E}$. The leaves of these trees are labeled with $\Sigma$-unification problems, where none of the rules apply. A leaf is called a **success node**, if the corresponding $\Sigma$-unification problem is in $\Sigma$-solved form, otherwise it is called a **failure node**.

We will prove (cf. 4.5.15) that the set of substitutions corresponding to the $\Sigma$-unification problems of the success nodes of the unification tree are complete sets of $\Sigma$-unifiers of $\mathcal{E}$.

## 4.5 General $\Sigma$-Unification ($\Sigma\mathcal{UT}$)

We present a $\Sigma$-unification algorithm that solves $\Sigma$-unification problems by transforming them into $\Sigma$-solved form. For a given $\Sigma$-unification problem $\mathcal{E}$ it returns a complete set of $\Sigma$-unifiers, if $\mathcal{E}$ is $\Sigma$-unifiable, and fails otherwise. It is a generalization of Huet's higher-order unification algorithm [Hue72, Hue76], as presented in [SG89].

**Definition 4.5.1 ($\mathcal{SIM}$: Simplification of $\Sigma$-Unification Problems)** Let $\mathcal{SIM}$ be the following inference system:

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.(\lambda X_{\mathbb{A}}.\mathbf{A}) =^? (\lambda Y_{\mathbb{A}}.\mathbf{B}) \wedge \mathcal{E} \quad Z \notin \mathbf{Dom}(\Gamma)}{\langle\Gamma, [Z^0{::}\mathbb{A}]\colon\mathcal{R}\rangle.[Z/X]\mathbf{A} =^? [Z/Y]\mathbf{B} \wedge \mathcal{E}}\,\mathcal{SIM}(\alpha)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.(\lambda X_{\mathbb{A}}.\mathbf{A}) =^? \mathbf{B} \wedge \mathcal{E} \quad Z \notin \mathbf{Dom}(\Gamma)}{\langle\Gamma, [Z^0{::}\mathbb{A}]\colon\mathcal{R}\rangle.[Z/X]\mathbf{A} =^? (\mathbf{B}Z) \wedge \mathcal{E}}\,\mathcal{SIM}(\eta)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E} \wedge \top_o}{\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}}\,\mathcal{SIM}(\top_o) \qquad \frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A} =^? \mathbf{A} \wedge \mathcal{E}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}}\,\mathcal{SIM}(triv)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.h\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \overline{\Sigma} \cup \mathbf{Dom}(\Gamma^0) \cup \mathbf{Dom}(\Gamma^-)}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{SIM}(dec)$$

This set of rules is used with the convention that all formulae are eagerly reduced to head normal form, i.e. each rule consists of two parts, first applying the transformation, and then head reducing to head normal form. Furthermore, we apply these rules with the understanding that the operators $\wedge$ and $=^?$ are commutative $(\mathcal{E} \wedge \mathcal{F}) = (\mathcal{F} \wedge \mathcal{E})$ and associative $(\mathcal{E} \wedge (\mathcal{F} \wedge \mathcal{G})) = ((\mathcal{E} \wedge \mathcal{F}) \wedge \mathcal{G})$. Note that, in contrast to [Mil92], we consider the quantifier prefixes as declarations and therefore do not need inference rules for quantifier exchange.

Clearly the set $\mathcal{SIM}$ of transformations is terminating and confluent up to $\alpha$-equivalence. Thus we can use it to reduce unification problems to a unique normal form (we have assumed implicit $\alpha$-conversion for $\Sigma$-unification problems) which we call **$\mathcal{SIM}$-normal form**. By inspection we can easily see, that in $\mathcal{SIM}$-normal forms all pairs must be of the form $h\overline{\mathbf{U}} =^? h\overline{\mathbf{V}}$, where $h$ and $k$ are constants or variables.

We sometimes use the following inference rule that combines the $\mathcal{SIM}$ inference rules, in order to make $\mathcal{SIM}$ simplifications explicit:

$$\frac{\mathcal{E}}{\mathcal{E}'}\,\mathcal{SIM}$$

if $\mathcal{E}'$ is the $\mathcal{SIM}$-normal form of $\mathcal{E}$.

**Remark 4.5.2** At first glance the rule $\mathcal{SIM}(\eta)$ seems to need the further assumptions $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$ and $\mathfrak{d}(\mathbb{B}) = \mathbb{A}$ in order to be sound, since it corresponds to *sort:η:top*, which has similar preconditions. But these assumptions are trivially entailed for Σ-unifications with the **Rdom** -assumptions from 4.4.11.

**Lemma 4.5.3** *If* $\mathcal{D}: \mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{E}'$, *then* $\mathbf{wsU}(\Sigma, \mathcal{E}) = \mathbf{wsU}(\Sigma, \mathcal{E}')[\mathcal{E}]$.

**Proof:** Clearly it suffices to show the assertion for the case, where $\mathcal{D}$ consists of a single rule application. The assertion is trivial for $\mathcal{SIM}(triv)$ and $\mathcal{SIM}(\top_o)$. For $\mathcal{SIM}(\alpha)$ let

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.(\lambda X_{\mathbb{A}}.\mathbf{A}) =^? (\lambda Y_{\mathbb{A}}.\mathbf{B}) \quad Z \notin \mathbf{Dom}(\Gamma)}{\langle\Gamma'\colon\mathcal{R}\rangle.[Z/X]\mathbf{A} =^? [Z/Y]\mathbf{B} \wedge \mathcal{E}}\,\mathcal{SIM}(\alpha)$$

where $\Gamma' := \Gamma, [Z^0::\mathbb{A}]$. We first convince ourselves that any Σ-substitution $\sigma$ is an $\mathcal{R}_\Gamma$-substitution, iff it is an $\mathcal{R}_{\Gamma'}$-substitution. So let $\sigma$ be an $\mathcal{R}_\Gamma$-substitution. Since we have assumed that $Z^0 \notin \mathbf{Dom}(\Gamma)$, $\sigma$ is also an $\mathcal{R}_{\Gamma'}$-substitution. If on the other hand $\sigma$ is an $\mathcal{R}_{\Gamma'}$-substitution, then $Z^0 \notin \mathbf{Im}(\sigma)$, since it is locally bound, therefore we have $\Gamma^-, \Delta \vdash_\Sigma \sigma::\Gamma^+$.

Furthermore, let $\theta$ be a Σ-substitution with $\Delta, \Gamma^- \vdash_\Sigma \theta::\Gamma^+$, then $\Delta, \Gamma^- \vdash_\Sigma \theta(\lambda X.\mathbf{A}) = \theta(\lambda Y.\mathbf{B})$, iff $\Delta, \Gamma^- \vdash_\Sigma \theta(\lambda Z.[Z/X]\mathbf{A}) = \theta(\lambda Z.[Z/Y]B)$ by $\alpha$-conversion and $\Delta, \Gamma^- \vdash_\Sigma (\lambda Z.\theta([Z/X]\mathbf{A}) = (\lambda Z.\theta([Z/Y]B)$, since we can assume that $X, Y \neq \mathbf{Dom}(\Gamma)$. However the last condition is equivalent to $\theta([Z/X]\mathbf{A}) = \theta([Z/Y]B)$ by *ms:trans*. Thus the sets of substitutions that solve $\mathcal{E}$ and $\mathcal{E}'$ are identical.

In the presence of $\mathcal{SIM}(\alpha)$ the rule $\mathcal{SIM}(\eta)$ is equivalent to a direct consequence of $\eta$-conversion: let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.(\lambda X_{\mathbb{A}}.\mathbf{A}) =^? \mathbf{B}$, then $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$ and $\mathfrak{d}(\mathbb{B}) = \mathbb{A}$, since we have restricted Σ-unification problems by an **Rdom** condition 4.4.11, so by *sort:η:top* $\Gamma \vdash_\Sigma \mathbf{B}{=}_\eta(\lambda X_{\mathbb{A}}.\mathbf{B}X)$. Therefore the set of Σ-unifiers does not change by replacing $\mathbf{B}$ by its $\eta$-expansion $(\lambda X_{\mathbb{A}}.\mathbf{B}X)$. Now we obtain the assertion for $\mathcal{SIM}(\eta)$ by that for $\mathcal{SIM}(\alpha)$.

In the $\mathcal{SIM}(dec)$ case we have

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.h\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \overline{\Sigma} \cup \mathbf{Dom}(\Gamma)}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{SIM}(dec)$$

Let $\theta \in \mathbf{wsU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E})$, so $\theta(\mathbf{U}^i){=}_\beta\theta(\mathbf{V}^i)$ for all $1 \le i \le n$ and therefore

$$\theta(h\overline{\mathbf{U}^n}) \doteq h\overline{\theta(\mathbf{U}^n)}{=}_\beta h\overline{\theta(\mathbf{V}^n)} \doteq \theta(h\overline{\mathbf{V}^n})$$

Thus for any atom $h$ we have $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$. ∎

**Definition 4.5.4 ($\Sigma\mathcal{UT}$: Transformations for $\Sigma$-Unification)** Let $\Sigma\mathcal{UT}$ be the system $\mathcal{SIM}$, augmented by the following inference rules. Just as in $\mathcal{SIM}$ leave the associativity and commutativity of $\wedge$ and $=^?$ implicit:

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.h\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \mathbf{Dom}(\Gamma^+)}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}}\ \Sigma\mathcal{UT}(dec)$$

For the following rules let $\mathbf{G} \in \mathcal{A}^h(\Sigma, \Delta, \mathcal{C})$ be a general binding of sort $\mathbb{A}$ that approximates the head $h$.

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma^+(F) = \mathbb{A} \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F^+, \mathbf{G})}{\langle\Gamma, \mathcal{C}\colon\mathcal{R}[\mathbf{G}/F]\rangle.F =^? \mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E})}\ \Sigma\mathcal{UT}(\mathit{flex}/\mathit{orig})$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.F\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma^+(F) = \mathbb{A} \quad \Gamma^+(H) = \mathbb{B} \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F^+, \mathbf{G})}{\langle\Gamma, \mathcal{C}\colon\mathcal{R}[\mathbf{G}/F]\rangle.F =^? \mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E})}\ \Sigma\mathcal{UT}(guess)$$

This set of rules is used with the convention, that all formulae are eagerly reduced to $\mathcal{SIM}$-normal form, i.e. each rule consists of two parts, first applying the transformation, and maximally $\mathcal{SIM}$-reducing afterwards.

**Remark 4.5.5** Our $\Sigma\mathcal{UT}(\mathit{flex}/\mathit{orig})$ rule subsumes Huet's rules of imitation ($\mathbf{G}$ has head $h$) and projection ($\mathbf{G}$ is a projection binding) transformations (see [Sny91]), since $\mathcal{A}^h(\Sigma, \Delta, \mathcal{C})$ contains imitation and projection bindings. Note that $\mathcal{A}^h(\Sigma, \Delta, \mathcal{C})$ also contains weakening bindings, which correspond to the concept of a weakening transformation, which is needed in $\Sigma\Lambda$, where we use term declarations to model subsorting.

Note that the rule $\Sigma\mathcal{UT}(guess)$ is finitely branching in our context, since the set of general bindings $\mathbf{G}$ of sort $\mathbb{A}$ is bounded by the number of term declarations in $\Sigma$ and the number of variable declarations in $\Gamma$. However, since the sorts of general bindings are about the only constraints on the set of applicable general bindings, the branching factor of the $\Sigma$-unification algorithm corresponding to these rules makes it infeasible in practice.

**Remark 4.5.6** If sort computation should turn out to be decidable the inference rule $\Sigma\mathcal{UT}(elim)$, defined below, is effective, and can be added to $\mathcal{SIM}$.

$$\frac{\langle\Gamma, [F^+{::}\mathbb{A}], \overline{[X_n^0{::}\partial^n(\mathbb{A})]}\colon\mathcal{R}\rangle.F\overline{X_n^0} =^? \mathbf{A} \wedge \mathcal{E} \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F^+, \lambda\overline{X_n}.\mathbf{A})}{\langle\Gamma\colon\mathcal{R}[\mathbf{A}/F]\rangle.F =^? (\lambda\overline{X_n}.\mathbf{A}) \wedge [(\lambda\overline{X_n}.\mathbf{A})/F]\mathcal{E}}\ \Sigma\mathcal{UT}(elim)$$

In contrast to the substitution in $\Sigma\mathcal{UT}(\mathit{flex}/\mathit{orig})$ and $\Sigma\mathcal{UT}(guess)$, where well-sortedness of the added pair is guaranteed by 4.2.3, we have to check for well-sortedness before eliminating the variable $F$. The following example shows that this gives practical improvements.

**Example 4.5.7** Let $\Sigma := \{[a{::}\mathbb{C} \to \mathbb{B}], [c{::}\mathbb{C}]\}$, then

$$\dfrac{\dfrac{\dfrac{\langle [F^+{::}\mathbb{A} \to \mathbb{B}], [Y^-{::}\mathbb{A}]{:}\,\emptyset\rangle.FY =^? ac}{\langle [F^+{::}\mathbb{A} \to \mathbb{B}], [G^+{::}\mathbb{A} \to \mathbb{C}], [Y^-{::}\mathbb{A}]{:}\,\emptyset\rangle.a(GY) =^? ac \wedge F =^? \lambda Z_{\mathbb{A}}.a(GZ)}\ \Sigma\mathcal{UT}(\mathit{flex/orig})}{\langle [F^+{::}\mathbb{A} \to \mathbb{B}], [G^+{::}\mathbb{A} \to \mathbb{C}], [Y^-{::}\mathbb{A}]{:}\,\emptyset\rangle.GY =^? c \wedge F =^? \lambda Z_{\mathbb{A}}.a(GZ)}\ \Sigma\mathcal{UT}(\mathit{dec})}{\langle [F^+{::}\mathbb{A} \to \mathbb{B}], [G^+{::}\mathbb{A} \to \mathbb{C}], [Y^-{::}\mathbb{A}]{:}\,\emptyset\rangle.c =^? c \wedge F =^? \lambda Z_{\mathbb{A}}.ac \wedge G =^? \lambda W_{\mathbb{A}}.c}\ \Sigma\mathcal{UT}(\mathit{flex/orig})$$

The inference rule $\Sigma\mathcal{UT}(\mathit{elim})$ immediately computes the $\Sigma$-unifier $[\lambda X_{\mathbb{A}}.ac/F]$ and therefore replaces two applications of $\Sigma\mathcal{UT}(\mathit{flex/orig})$ and one of $\mathcal{SIM}(\mathit{dec})$.

For showing the soundness of $\Sigma\mathcal{UT}$ we start out with some technical lemmata, which will allow us later on to prove the soundness theorem. In particular, the sort conditions in the inference rules enforce the preservation of well-sortedness of the $\Sigma$-unifiers.

**Lemma 4.5.8** *Let $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{E}'$ by a single application of $\Sigma\mathcal{UT}(\mathit{dec})$ to a pair $H\overline{\mathbf{U}^n} =^? H\overline{\mathbf{V}^n}$, then for any substitution $\theta$ we have*

1. *If $H \in \mathbf{supp}(\theta)$, then $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E}')$ implies that $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$.*
2. *If $H \notin \mathbf{supp}(\theta)$, then $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$, iff $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E}')$.*

**Proof:** Let $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E}')$, so $\theta(\mathbf{U}^i) =_\beta \theta(\mathbf{V}^i)$ for all $1 \leq i \leq n$ and therefore

$$\theta(H\overline{\mathbf{U}^n}) \doteq \theta(H)\overline{\theta(\mathbf{U}^n)} =_\beta \theta(H)\overline{\theta(\mathbf{V}^n)} \doteq \theta(H\overline{\mathbf{V}^n})$$

Thus for any atom $H$ we have $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$. Now let $H \notin \mathbf{Dom}(\theta)$ and $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$, then $\theta(H) = H$, so in this case we have $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E}')$. □

By applying the rules $\Sigma\mathcal{UT}(\mathit{flex/orig})$ and $\Sigma\mathcal{UT}(\mathit{guess})$ we effectively commit ourselves to a particular approximation of a solution, and thus cannot reasonably expect to conserve the set of $\Sigma$-unifiers.

**Lemma 4.5.9** *If $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{E}'$ by a $\Sigma\mathcal{UT}$-derivation only containing applications of the rules $\Sigma\mathcal{UT}(\mathit{flex/orig})$ and $\Sigma\mathcal{UT}(\mathit{guess})$, then $\mathbf{wsU}(\Sigma, \mathcal{E}') \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$.*

**Proof:** The transformations $\Sigma\mathcal{UT}(\mathit{flex/orig})$ and $\Sigma\mathcal{UT}(\mathit{guess})$ can be divided into three parts, first adding a pair $X =^? \mathbf{G}$, then eliminating the variable, and finally $\mathcal{SIM}$-reducing. Clearly adding a new pair does not create new $\Sigma$-unifiers, so we must have $\mathbf{wsU}(\Sigma, \mathcal{E} \wedge X =^? \mathbf{A}) \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$. Thus obtain the assertion with 4.4.15 and 4.5.3. □

**Lemma 4.5.10** *If $\mathcal{E} \vdash_{\Sigma\mathcal{UT}(\mathit{elim})} \mathcal{E}'$, then $\mathbf{wsU}(\Sigma, \mathcal{E}) = \mathbf{wsU}(\Sigma, \mathcal{E}')$.*

**Proof:** Let $\Gamma^0(X_i) = \mathfrak{d}^n(\mathbb{A})$ and $\Gamma^+(F) = \mathbb{A}$ and $\mathcal{E} = \langle \Gamma{:}\,\mathcal{R}\rangle.F\overline{X^0_n} =^? \mathbf{A} \wedge \mathcal{E}'$ be a $\Sigma$-unification problem and $(F\overline{X}) =^? \mathbf{A}$ be the pair that the rule $\Sigma\mathcal{UT}(\mathit{elim})$ acts upon. We show that for an arbitrary $\Sigma$-unifier $\theta$ of $\mathcal{E}$, the formula $\lambda\overline{X}.\mathbf{A}$ is more general than $\theta(F)$. So let $\theta$ be an arbitrary $\Sigma$-unifier of $\mathcal{E}$ such that $\Xi \vdash_\Sigma \theta{::}\Gamma^+$, then

$$\Xi \vdash_\Sigma \theta(F) =_{\beta\eta} \theta(\lambda\overline{X}.F\overline{X}) =_{\beta\eta} \lambda\overline{X}.\theta(F\overline{X}) =_{\beta\eta} \lambda\overline{X}.\theta(\mathbf{A}) =_{\beta\eta} \theta(\lambda\overline{X}.\mathbf{A})$$

since the $X^0_i$ are not in $\mathbf{Dom}(\theta)$. This is just the claim with $\theta$ as the instantiating substitution. Now we obtain the assertion by 4.4.15. □

**Theorem 4.5.11 (Soundness of $\Sigma\mathcal{UT}$)** *If $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}' \vdash_{\Sigma\mathcal{UT}} \mathcal{F}$ such that $\mathcal{F}$ is in Σ-solved form, then the substitution $\sigma_{\mathcal{F}}|_{\mathbf{Dom}(\Gamma^+)} \in \mathbf{wsU}(\Sigma,\mathcal{E})$.*

**Proof:** We prove $\sigma_{\mathcal{F}} \in \mathbf{wsU}(\Sigma,\mathcal{E})$ by induction on the length of the transformation sequence using the above lemmata in the induction step. The restriction of $\sigma_{\mathcal{F}}$ does not affect the fact that $\sigma_{\mathcal{F}}|_{\mathbf{Dom}(\Gamma^+)}$ still Σ-unifies $\mathcal{E}$. □

So if the algorithm $\Sigma\mathcal{UT}$ returns a substitution $\theta$ for an initial system $\mathcal{E}$, then $\theta$ is indeed a Σ-unifier for $\mathcal{E}$. The main result of this section is the converse, namely, that given an initial Σ-unification problem $\mathcal{E}$ and a Σ-unifier $\theta$, the algorithm $\Sigma\mathcal{UT}$ can compute a Σ-unifier $\sigma$ of $\mathcal{E}$, which is more general than $\theta$.

As higher-order unification is undecidable [Gol81, Hue76, Luc72], our set of transformations cannot be terminating in general. We will prove, that $\Sigma\mathcal{UT}$ is a complete Σ-unification procedure, that is, if for any given $\theta \in \mathbf{wsU}(\Sigma,\mathcal{E})$ there is a $\Sigma\mathcal{UT}$-derivation $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{F}$ such that $\mathcal{F}$ is a Σ-unification problem in Σ-solved form, and $\sigma_{\mathcal{F}}$ is more general than $\theta$. For this we only need termination for $\Sigma\mathcal{UT}$ inference rules that approximate $\theta$.

The following measure provides the basis for defining the approximating rule applications and for proving their termination.

**Definition 4.5.12** Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}'$ be a Σ-unification problem and $\theta$ be an $\mathcal{R}_\Gamma$-substitution, then

$$\mu(\mathcal{E},\theta) := (\mu_1(\mathcal{E},\theta), \mu_2(\mathcal{E}))$$

is called a **measure for $\mathcal{E}$ and $\theta$**, iff $\mu_1(\mathcal{E},\theta)$ is a multiset of depths of semi-structural ΣΛ-derivations $\mathcal{A}_X\colon\Gamma \vdash_\Sigma \theta(x)::\Gamma^+(X)$, where $X^+ \in \mathbf{Dom}(\theta)$ is unsolved in $\mathcal{E}$ and $\mu_2(\mathcal{E})$ is the multiset of depths of formulae in $\mathcal{E}$. Furthermore, let $\prec$ be the strict lexicographic ordering for the obvious component orderings.

**Lemma 4.5.13** *Let $\mathcal{E}$ be a Σ-unification problem in $\mathcal{SIM}$-normal form, but not in Σ-solved form, $\theta \in \mathbf{wsU}(\Sigma,\mathcal{E})$, and $\mu(\mathcal{E},\theta)$ a measure for $\mathcal{E}$ and $\theta$, then there exists a Σ-unification problem $\mathcal{E}'$, an $\mathcal{E}'$-substitution $\theta'$, and a measure $\mu(\mathcal{E}',\theta')$ for $\mathcal{E}'$ and $\theta'$ such that $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{E}'$, and*

1. $\theta \doteq \theta'[\mathcal{E}]$,
2. $\theta' \in \mathbf{wsU}(\Sigma,\mathcal{E}')$,
3. $\mu(\mathcal{E}',\theta') \prec \mu(\mathcal{E},\theta)$.

**Proof:** Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{F}, \Xi \vdash_\Sigma \theta::\Gamma^+$, and $\mathbf{F}\overline{\mathbf{U}} =^? \mathbf{G}\overline{\mathbf{V}}$ be a pair in $\mathcal{F}$, that is not Σ-solved. We observe that $\mathbf{F}$ and $\mathbf{G}$ must be atoms and cannot be equal constants, and moreover we cannot have $\Gamma \vdash_\Sigma \mathbf{A}{=_{\beta\eta}}\mathbf{B}$, since $\mathcal{E}$ is in $\mathcal{SIM}$-normal form.

If $\mathbf{F} = \mathbf{G}$ is a variable not in $\mathbf{supp}(\theta)$, then $\Sigma\mathcal{UT}(dec)$ applies. By 4.5.8 we have $\theta \in \mathbf{wsU}(\Sigma,\mathcal{E}')$ and $\mu(\mathcal{E}',\theta) \prec \mu(\mathcal{E},\theta)$, since $\mu_1(\mathcal{E}',\theta) \preceq \mu_1(\mathcal{E},\theta)$ and $\mu_2(\mathcal{E}') \prec \mu(\mathcal{E})$.

Otherwise either $\mathbf{F} \neq \mathbf{G}$ or $\mathbf{F} = \mathbf{G} \in \mathbf{supp}(\theta)$. In both cases, since $\mathcal{E}$ is Σ-unifiable, either $\mathbf{F}$ or $\mathbf{G}$ is an unsolved variable $F \in \mathbf{supp}(\theta)$ with $\Gamma^+(F) = \mathbb{A}$ at the head. Without loss of generality we assume that $F = \mathbf{F}$. Now, since $\theta$ is an $\mathcal{R}_\Gamma$-substitution we have $\Xi,\Gamma \vdash_\Sigma \theta(F)::\mathbb{A}$ and $\Xi,\Gamma \vdash_\Sigma \overline{\mathcal{R}}(F,\theta(F))$. By the general binding theorem 4.2.4 there exists a general binding $\mathbf{G} \in \mathcal{A}^h_{\mathbb{A}}(\Sigma,(\Xi,\Gamma^+),\mathcal{C})$ of sort $\mathbb{A}$ and a Σ-substitution $\rho$, such that $\mathbf{supp}(\rho) = \mathbf{Dom}(\mathcal{C})$ and $\mathcal{C},\Gamma \vdash_\Sigma \rho(\mathbf{G}){=_{\beta\eta}}\mathbf{A}$. Since the variables in $\mathbf{Dom}(\mathcal{C})$ are new, we also have $\overline{\mathcal{R}}(F^+,\mathbf{G})$. Therefore,

- if $\mathbf{head}(\mathbf{G}) \notin \mathbf{supp}(\theta)$, then $\Sigma\mathcal{UT}(\mathit{flex/orig})$ applies.
- if $\mathbf{head}(\mathbf{G}) \in \mathbf{supp}(\theta)$ then $\Sigma\mathcal{UT}(\mathit{guess})$ applies.

In all these cases we set $\theta' := \theta \cup \rho$ and have $\theta = \theta'[\mathcal{E}]$, since $\mathbf{supp}(\rho) \cap \mathbf{Dom}(\Gamma^+) = \mathbf{Dom}(\mathcal{C}) \cap \mathbf{Dom}(\Gamma^+) = \emptyset$ and $\theta' \in \mathbf{wsU}(\Sigma, \mathcal{E}')$ by 4.5.11. To see that $\rho$ is a $\overline{\mathcal{R}[\mathbf{G}/X]}$-substitution, we have to convince ourselves that $Y \notin \mathbf{Free}(\rho(X))$ for all $X \in \mathbf{Dom}(\rho) = \mathbf{Dom}(\mathcal{C})$ such that $(X,Y) \in \mathcal{R}[\mathbf{G}/F]$. By definition of $\mathcal{R}[\mathbf{G}/F]$ this is the case, if $Y \notin \mathbf{Free}(\rho(X))$ for all $X \in \mathbf{Dom}(\mathcal{C})$ such that $(F,Y) \in \mathcal{R}$. If this were the case, then $Y$ would be free in $\rho(\mathbf{G}) = \theta(F)$, which would contradict the assumption that $\theta$ is an $\mathcal{R}$-substitution.

If $\mathcal{A}{:}\, \Xi, \Gamma \vdash_\Sigma \theta(F)::\mathbb{A}$ is the semi-structural $\Sigma\Lambda$-derivation that contributes to $\mu_1(\mathcal{E}, \theta)$, then the general binding theorem guarantees the existence of semi-structural $\Sigma\Lambda$-derivations $\mathcal{R}^i{:}\, \Xi, \Gamma \vdash_\Sigma \rho(X)::\mathcal{C}(X)$ for all $X \in \mathbf{Dom}(\theta)$. Since $\Sigma\mathcal{UT}(\mathit{guess})$ and $\Sigma\mathcal{UT}(\mathit{flex/orig})$ remove the variable $F$ from the set of variables in $\mathbf{supp}(\theta)$ that are not $\Sigma$-solved in $\mathcal{E}$, and since they replace it with the set $\mathbf{Dom}(\mathcal{C}) = \mathbf{supp}(\rho)$, we have $\mu_1(\mathcal{E}', \theta') \prec \mu_1(\mathcal{E}, \theta)$. Thus we have $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$. □

If we call such a transformation **$\mu$-prescribed**, then each application of a $\mu$-prescribed transformation decreases the well-founded measure $\mu$. Thus any sequence of $\mu$-prescribed transformations must terminate. The previous lemma also guarantees that any system obtained by exhaustively applying $\mu$-prescribed transformations to a $\Sigma$-unifiable system must be $\Sigma$-solved, since otherwise it guarantees another $\mu$-prescribed transformation.

**Corollary 4.5.14** *If $\mathcal{E}$ is a $\Sigma$-unifiable unification problem such that no $\mu$-prescribed transformation rule from $\Sigma\mathcal{UT}$ is applicable, then $\mathcal{E}$ is in $\Sigma$-solved form.*

**Theorem 4.5.15 (Completeness Theorem for $\Sigma\mathcal{UT}$)** *For any $\Sigma$-unification problem $\mathcal{E}$ and any $\Sigma$-substitution $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$, there is a $\Sigma\mathcal{UT}$-derivation $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{F}$ such that $\mathcal{F}$ is in $\Sigma$-solved form and $\Gamma \vdash_\Sigma \sigma_\mathcal{F} \preceq_{\beta\eta} \theta[\mathcal{E}]$.*

**Proof:** Let $\mathcal{E} = \langle \Gamma{:}\,\mathcal{R}\rangle.\mathcal{G}$ and $\mathcal{D}{:}\, \mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{F}$ be a maximal $\mu$-prescribed $\Sigma\mathcal{UT}$-derivation out of $\mathcal{E}$. By 4.5.13 this is alway finite, so we can prove the assertion by induction on the number $n$ of nodes in $\mathcal{D}$. If $n = 0$, then $\mathcal{E}$ is in $\Sigma$-solved form and $\sigma_\mathcal{E}$ is a most general $\Sigma$-unifier for $\mathcal{E}$. In particular, we have $\Gamma \vdash_\Sigma \sigma_\mathcal{E} \preceq_{\beta\eta} \theta[\mathcal{E}]$.

If $n > 0$, then there is a $\mu$-prescribed inference $\mathcal{E} \vdash \mathcal{E}'$ and a $\Sigma$-substitution $\theta'$ satisfying 4.5.13. By inductive hypothesis there is a $\Sigma\mathcal{UT}$-derivation $\mathcal{E}' \vdash_{\Sigma\mathcal{UT}} \mathcal{F}$ such that $\Gamma \vdash_\Sigma \sigma_\mathcal{F}\theta'[\mathcal{E}']$. By 4.5.11 we have $\sigma_\mathcal{F} \in \mathbf{wsU}(\Sigma, \mathcal{E}') \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$. Furthermore, by inspection of the inference rules we see that $\Sigma\mathcal{UT}$ rules only expand the set of positive variables in $\Gamma$, so $\Gamma \vdash_\Sigma \sigma_\mathcal{F} \preceq_{\beta\eta} \theta'[\mathcal{E}']$ implies $\Gamma \vdash_\Sigma \sigma_\mathcal{F} \preceq_{\beta\eta} \theta'[\mathcal{E}]$, which in turn yields the assertion with the conclusion $\theta' = \theta[\mathcal{E}]$ of 4.5.13. □

If we combine the soundness results theorem 4.5.11 with the completeness result from theorem 4.5.15, we can characterize the set of solutions found by the algorithm $\Sigma\mathcal{UT}$ by the following corollary.

**Corollary 4.5.16** *For any $\Sigma$-unification problem $\mathcal{E}$ the set*

$$\Sigma\mathcal{UT}(\mathcal{E}) := \{\sigma_\mathcal{F} \mid \mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{F} \text{ and } \mathcal{F} \text{ is in } \Sigma\text{-solved form}\}$$

*is a complete set of $\Sigma$-unifiers for $\mathcal{E}$.*

**Example 4.5.17** Let $\mathbb{R}, \mathbb{R}^+$ be sorts of type $\iota$ with the intended meanings of real numbers and non-negative real numbers. Furthermore, let $\mathbb{M}, \mathbb{P}, \mathbb{D}, \mathbb{C}$ sorts of type $\iota \to \iota$ such that all have domain and codomain sorts $\mathbb{R}$. These have the intended meanings of monomials, polynomials, differentiable and continuous functions on the reals. Finally, let $\Sigma$ be the signature with the following term declarations:

$$\begin{array}{l}
[+{::}\mathbb{R} \to \mathbb{R} \to \mathbb{R}], [*{::}\mathbb{R} \to \mathbb{R} \to \mathbb{R}], [\forall[X{::}\mathbb{R}]. * XX{::}\mathbb{R}^+],\\
[\mathbb{R}^+ \leq \mathbb{R}], [\mathbb{M} \leq \mathbb{P}], [\mathbb{P} \leq \mathbb{D}], [\mathbb{D} \leq \mathbb{C}]\\
[\lambda X_{\mathbb{R}}.X{::}\mathbb{M}], [\forall[Y{::}\mathbb{R}].(\lambda X_{\mathbb{R}}.Y){::}\mathbb{M}],\\
[\forall[F,G{::}\mathbb{M}].(\lambda X_{\mathbb{R}}. * (FX)(GX)){::}\mathbb{M}], [\forall[F,G{::}\mathbb{P}].(\lambda X_{\mathbb{R}}. + (FX)(GX)){::}\mathbb{P}],\\
[\partial{::}\mathbb{D} \to \mathbb{C}], [\partial|{:}\mathbb{P} \to \mathbb{P}], [\partial|{:}\mathbb{M} \to \mathbb{M}].
\end{array}$$

Thus $\Sigma$ formalizes a small fragment of elementary calculus. In this setting we can answer the question whether there are differentiable functions that are non-negative by solving the following unification problem:

$$\langle [F{::}\mathbb{D}], [G{::}\mathbb{R} \to \mathbb{R}^+]{:}\,\emptyset\rangle.F =^? G$$

This is a flex-flex problem, so we can use *Σ𝒰𝒯(guess)* with the general imitation binding $(\lambda X_{\mathbb{R}}. * (H^1X)(H^1X))$ induced by the term declaration $[\forall[X{::}\mathbb{R}]. * XX{::}\mathbb{R}^+] \in \Sigma$, thus we obtain

$$\langle [F{::}\mathbb{D}], [G{::}\mathbb{R} \to \mathbb{R}^+], [H^1{::}\mathbb{R} \to \mathbb{R}]{:}\,\emptyset\rangle.F =^? \lambda X_{\mathbb{R}}.*(H^1X)(H^1X) \wedge G =^? \lambda X_{\mathbb{R}}.*(H^1X)(H^1X)$$

With the weakening bindings $F =^? H^3_{\mathbb{P}}$ and $H^3 =^? H^4_{\mathbb{M}}$ from the subsort declarations $[\mathbb{P} \leq \mathbb{D}], [\mathbb{M} \leq \mathbb{P}] \in \Sigma$ we have

$$H^4_{\mathbb{M}} =^? \lambda X_{\mathbb{R}}. * (H^1X)(H^1X)$$

To make the Σ𝒰𝒯-derivation more legible, we drop the declaration and delete solved pairs in Σ-unification problems in this example. To keep the sort information complete we indicate the sorts of variables in the subscript. We continue our Σ𝒰𝒯-derivation by applying *Σ𝒰𝒯(flex/orig)* with the binding $H^4_{\mathbb{M}} =^? \lambda X_{\mathbb{R}}. * (H^5_{\mathbb{M}}X)(H^6_{\mathbb{M}}X)$ from the declaration $[\forall[F,G{::}\mathbb{M}].\lambda X_{\mathbb{R}} * (FX)(GX){::}\mathbb{P}] \in \Sigma$ we obtain

$$H^5_{\mathbb{M}}Y =^? H^1_{\mathbb{R}\to\mathbb{R}}Y \wedge H^6_{\mathbb{M}}Y =^? H^1_{\mathbb{R}\to\mathbb{R}}Y$$

The first pair can be solved with the imitation binding $H^1_{\mathbb{R}\to\mathbb{R}} =^? (\lambda X_{\mathbb{R}}.H^5_{\mathbb{M}}(H^7_{\mathbb{R}\to\mathbb{R}}Y)$ and the subsequent projection using $H^7_{\mathbb{R}\to\mathbb{R}} =^? (\lambda X_{\mathbb{R}}.X)$. This leaves us with the problem

$$H^6_{\mathbb{M}}Y =^? H^5_{\mathbb{M}}Y$$

which can be solved with the bindings $H^6_{\mathbb{M}} =^? (\lambda X_{\mathbb{R}}.H^5_{\mathbb{M}}(H^8_{\mathbb{M}}Y)$ and $H^8_{\mathbb{M}} =^? (\lambda X_{\mathbb{R}}.X)$. Note that the last general binding comes from the term declaration $[\lambda X_{\mathbb{R}}.X{::}\mathbb{M}] \in \Sigma$.

Collecting all partial solutions obtained in this Σ𝒰𝒯-derivation gives us the Σ-unifier $\sigma := [\mathbf{D}/F], [\mathbf{D}/G]$ where $\mathbf{D} = (\lambda X_{\mathbb{R}}. * (H^5_{\mathbb{M}}X)(H^5_{\mathbb{M}}X))$, which is just the most general expression for a monomial with even degree.

### 4.6 Pre-$\Sigma$-Unification ($\Sigma\mathcal{PT}$)

Just as in the case of unification for $\Lambda$, the rule $\Sigma\mathcal{UT}(guess)$ gives rise to a serious explosion of the search space for unifiers (cf. 4.5.5), which makes general higher-order unification in this form impractical. Huet's solution to this problem was to redefine the higher order unification problem to a form sufficient for refutation purposes: for the pre-unification problem flex-flex pairs are considered already solved, since they can always be trivially solved by binding the head variables to special constant functions that identify the formulae by absorbing their arguments.

We give a generalization of Huet's pre-unification procedure to $\Sigma\Lambda$. However in $\Sigma\Lambda$ the solution to the flex-flex problem is not as simple as in the unsorted case, since the heads of flex-flex pairs can be variables of functional base sorts $\mathbb{A}$. In this case flex-flex-pairs, are not solvable independently of their arguments, since in general the constant functions needed for absorbing the arguments are not of sort $\mathbb{A}$. Our solution to this problem is to modify the definition of pre-solved pairs and to keep the guess rule, but to restrict its application to the functional flex-flex case. Furthermore, pre-$\Sigma$-unification only makes sense for regular signatures (cf. 3.6.16), as the following example shows. Therefore we will only consider regular signatures in the following.

**Example 4.6.1 (Non-Regular Pre-$\Sigma$-Unification)** We consider the non-regular signature given by $\mathcal{S} := \{\mathbb{A}, \mathbb{B}\}$, $\tau(\mathbb{A}) = \tau(\mathbb{B}) = \alpha$, and $\Sigma := \{[c{::}\mathbb{A}], [c{::}\mathbb{B}]\}$. The $\Sigma$-substitution $[c/X], [c/Y]$ is the only $\Sigma$-unifier of the unification problem $\langle [X^+{::}\mathbb{A}], [Y^+{::}\mathbb{B}]{:}\,\emptyset\rangle.X =^? Y$, but it can only be found by applying some kind of $\Sigma\mathcal{UT}(guess)$ transformation.

**Lemma 4.6.2** *The problem of deciding whether a given signature $\Sigma$ is regular or not, can be reduced to the $\Sigma$-unification problem. Thus it is undecidable.*

**Proof:** Let $\Sigma$ be a regular signature, $\mathbb{A}, \mathbb{B}$, and $\mathbb{D}$ new base sorts of base type $\alpha$, and $\Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{C}$ and $\Delta \vdash_\Sigma \mathbf{B}{::}\mathbb{C}$ where $\tau(\mathbb{C}) = \gamma$. Furthermore, let $\overline{\Sigma}' := \overline{\Sigma} \cup \{h_{\gamma\to\alpha}\}$ and

$$\Sigma' := \Sigma \cup \{[h{::}\mathbb{C} \to \mathbb{D}], [\forall\Delta.h\mathbf{A}{::}\mathbb{A}], [\forall\Delta.h\mathbf{B}{::}\mathbb{B}]\}$$

Clearly $\Sigma'$ is a valid signature, since $\mathbb{A}, \mathbb{B}, \mathbb{D}$ are of base type and therefore $\mathbb{A}$ **Rdom** $\mathbb{B}$ and $\mathbb{B}$ **Rdom** $\mathbb{D}$. If there is a $\Sigma$-unifier $\theta \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$ of $\langle\Delta{:}\,\emptyset\rangle.\mathbf{A} =^? \mathbf{B}$, then $\Gamma \vdash_\Sigma h\theta(\mathbf{A}){::}\mathbb{A}$ and $\Gamma \vdash_\Sigma h\theta(\mathbf{B}){::}\mathbb{B}$ by *ws:td* and *ws:subst*. Furthermore, we have $\Gamma \vdash_\Sigma h\theta(\mathbf{A}){::}\mathbb{B}$ by 3.5.1, since $\Gamma \vdash_\Sigma \theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$, and therefore $\Gamma \vdash_\Sigma h\theta(\mathbf{A}) =_{\beta\eta} h\theta(\mathbf{B})$ by *tr:app:arg*. In particular, we have found a formula with two least sorts. Thus $\Sigma'$ is regular, if $\mathbf{A}$ and $\mathbf{B}$ are not $\Sigma$-unifiable. $\square$

**Remark 4.6.3** Note that the previous undecidability result is independent of the decidability of the sort computation problem.

**Definition 4.6.4 (Pre-Equality)** Let $\Gamma \vdash_\Sigma \mathbf{A} =^p{}_{\beta\eta} \mathbf{B}$ be the **pre-equality** judgment defined by the inference system for sorted $\beta\eta$-equality augmented by the following inference rule

$$\frac{\mathfrak{r}^n(\Gamma(F)) = \mathfrak{r}^k(\Gamma(G)) \quad \mathbf{ln}(\Gamma(F)) \geq n \quad \mathbf{ln}(\Gamma(G)) \geq k}{\vdash_\Sigma F\overline{\mathbf{U}^n} =^p G\overline{\mathbf{V}^k}}\ \textit{sort:pre:top}$$

Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}'$ be a Σ-unification problem, then we call an $\mathcal{R}_\Gamma$-substitution $\sigma$ a **pre-Σ-unifier** of the pair $\mathbf{A} =^? \mathbf{B} \in \mathcal{E}'$, iff $\Delta, \Gamma^- \vdash_\Sigma \sigma(\mathbf{A}) =^p{}_{\beta\eta} \sigma(\mathbf{B})$. We denote the set of pre-Σ-unifiers by $\mathbf{wsPU}(\Sigma, \mathcal{E})$.

**Definition 4.6.5 (Pre-Σ-Solved Form)** Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}'$ be a Σ-unification problem, then we call a pair $F^+\overline{\mathbf{U}^k} =^? G^+\overline{\mathbf{V}^n}$ **pre-Σ-solved** in $\mathcal{E}$, iff $\Gamma^+(F) = \overline{\mathbb{A}^k} \to \mathbb{B}$ and $\Gamma^+(G) = \overline{\mathbb{C}^n} \to \mathbb{B}$. A Σ-unification problem $\mathcal{E}'$ is in **pre-Σ-solved form**, iff all pairs in $\mathcal{E}'$ are Σ-solved or pre-Σ-solved.

If $\mathcal{E}$ is a Σ-unification problem in pre-Σ-solved form, then we can write $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}_\sigma \wedge \mathcal{E}_p$, where the pairs in $\mathcal{E}_p$ are pre-Σ-solved, but Σ-unsolved.

**Remark 4.6.6** Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{F}$ be a Σ-unification problem in pre-Σ-solved form such that $\mathcal{G} := F\overline{\mathbf{U}^n} =^? G\overline{\mathbf{V}^m}$ is a pre-Σ-solved pair in $\mathcal{F}$ with $\Delta(F) = \overline{\mathbb{A}^n} \to \mathbb{B}$, $\Gamma(G) = \overline{\mathbb{C}^m} \to \mathbb{D}$, and let $H$ be a variable of sort $\mathbb{B}$ not in $\mathbf{Dom}(\Gamma)$, then

$$\sigma_\mathcal{F} := [\lambda X^1_{\mathbb{A}^1} \ldots X^n_{\mathbb{A}^n}.H/F], [\lambda X^1_{\mathbb{C}^1} \ldots X^m_{\mathbb{C}^m}.H/G]$$

is a Σ-unifier for $\mathcal{E}$, since $\Gamma \vdash_\Sigma \sigma_\mathcal{F}(F\overline{\mathbf{U}^n}) =_{\beta\eta} H =_{\beta\eta} \sigma_\mathcal{F}(G\overline{\mathbf{V}^m})$. Clearly the sort conditions on $F$ and $G$ ensure that $\sigma_\mathcal{F}$ is an $\mathcal{R}_\Gamma$-substitution.

Thus pre-Σ-unifiers can always be extended to Σ-unifiers by finding trivial Σ-unifiers for the pre-Σ-solved pairs. Let $\mathcal{E} = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}_\sigma \wedge \mathcal{E}_p$ be a Σ-unification problem in pre-Σ-solved form, then we can construct a Σ-unifier $\theta$ for $\mathcal{E}_p$ as above, and see that $\theta \cup \sigma$ is a Σ-unifier of $\mathcal{E}$. Therefore a Σ-unification problem $\mathcal{E}$ is pre-Σ-unifiable, iff it is Σ-unifiable.

**Definition 4.6.7 ($\Sigma\mathcal{PT}$: Transformations for Pre-Σ-Unification)** We define the set $\Sigma\mathcal{PT}$ of **transformations for pre-Σ-unification** by modifying $\Sigma\mathcal{UT}$ (4.5.4): the inference rules of $\Sigma\mathcal{PT}$ are obtained from their $\Sigma\mathcal{UT}$ counterparts by requiring that they may not be performed on a pre-Σ-solved pair. Thus we have the following set of inference rules:

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.F\overline{\mathbf{U}^n} =^? F\overline{\mathbf{V}^n} \wedge \mathcal{E}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}} \; \Sigma\mathcal{PT}(dec)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.F\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F^+, \mathbf{G})}{\langle\Gamma, \mathcal{C}\colon\mathcal{R}[\mathbf{G}/F]\rangle.F =^? \mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E})} \; \Sigma\mathcal{PT}(flex-rig)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C} \vee F\overline{\mathbf{U}^n} =^? G\overline{\mathbf{V}^m} \wedge \mathcal{E} \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F, \mathbf{G})}{\langle\Gamma, \mathcal{C}\colon\mathcal{R}[\mathbf{G}/F]\rangle.F =^? \mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}} =^? G\overline{\mathbf{V}} \wedge \mathcal{E})} \; \Sigma\mathcal{PT}(guess)$$

where $\Gamma(F) = \mathbb{A}$, $\Gamma(G) = \mathbb{B}$, $\mathbf{ln}(\mathbb{A}) \geq n$, $\mathbf{ln}(\mathbb{B}) \geq m$, and $\mathbf{G} \in \mathcal{A}^h(\Sigma, \Delta, \mathcal{C})$ is a general binding of sort $\mathbb{A}$ that approximates some head $h$.

With this definition we immediately obtain the soundness of $\Sigma\mathcal{PT}$.

**Theorem 4.6.8 (Soundness of $\Sigma\mathcal{PT}$)** *Let $\mathcal{D}\colon \mathcal{E} \vdash_{\Sigma\mathcal{PT}} \mathcal{E}'$, if $\mathcal{E}' = \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}_\sigma \wedge \mathcal{E}_p$ is in pre-Σ-solved form, then $\sigma \in \mathbf{wsPU}(\Sigma, \mathcal{E})$.*

**Proof sketch:** All $\Sigma\mathcal{PT}$-transformations are special cases of the $\Sigma\mathcal{UT}$-transformations, so the assertion can be obtained with the same methods as 4.5.11. $\square$

For the completeness result we need an analogue of the termination lemma (4.5.13).

**Lemma 4.6.9** *Let $\mathcal{E}$ be a $\Sigma$-unification problem in $\mathcal{SIM}$-normal form, but not pre-$\Sigma$-solved and $\theta \in \mathbf{wsPU}(\Sigma, \mathcal{E})$, then there exists a $\Sigma$-unification problem $\mathcal{E}'$, an $\mathcal{E}'$-substitution $\theta'$, and a measure $\mu(\mathcal{E}', \theta')$ for $\mathcal{E}'$ and $\theta'$ such that $\mathcal{E} \vdash_{\Sigma\mathcal{PT}} \mathcal{E}'$, $\theta \doteq \theta'[\mathcal{E}]$, $\theta' \in \mathbf{wsPU}(\Sigma, \mathcal{E}')$, and $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$.*

**Proof sketch:** In the proof of 4.5.13 we observe that the restriction of the $\Sigma\mathcal{UT}$ inference rules to $\Sigma\mathcal{PT}$ inference rules by forbidding applications of *$\Sigma\mathcal{UT}$(flex/orig)* to flex-flex-pairs does not affect the result, since the forbidden applications to unsolved flex-flex pairs can be simulated by a *$\Sigma\mathcal{PT}$(guess)*. $\square$

**Theorem 4.6.10 (Completeness Theorem for $\Sigma\mathcal{PT}$)** *For any $\Sigma$-unification problem $\mathcal{E}$ and any $\theta \in \mathbf{wsPU}(\Sigma, \mathcal{E})$, there is a $\Sigma\mathcal{PT}$-derivation $\mathcal{E} \vdash_{\Sigma\mathcal{PT}} \mathcal{F}$ such that $\mathcal{F}$ is in pre-$\Sigma$-solved form and $\sigma_{\mathcal{F}} \preceq^{p}_{\beta\eta} \theta[\mathcal{E}]$.*

**Proof sketch:** The proof for the completeness of $\Sigma\mathcal{PT}$ is analogous to that of 4.5.15, using 4.6.9 instead of 4.5.13. $\square$

If we combine the soundness and completeness results 4.6.8 and 4.6.10, we can characterize the set of solutions found by the algorithm $\Sigma\mathcal{PT}$ by the following corollary.

**Corollary 4.6.11** *$\Sigma\mathcal{PT}$ is a complete pre-$\Sigma$-unification procedure. Moreover, if $\mathcal{E}$ is a $\Sigma$-unification problem, then the set $\Sigma\mathcal{PT}(\mathcal{E}) := \{\sigma \mid \mathcal{E} \vdash_{\Sigma\mathcal{PT}} \mathcal{E}_\sigma\}$ is a complete set of pre-$\Sigma$-unifiers for $\mathcal{E}$.*

**Example 4.6.12** Let us reconsider the example 4.5.17. Since $\mathbb{M}$ and $\mathbb{P}$ are base sorts, and therefore have length one, the $\Sigma\mathcal{UT}$-derivation presented there is also a $\Sigma\mathcal{PT}$-derivation, thus $\sigma$ is also a pre-$\Sigma$-unifier of $\langle [F{::}\mathbb{D}], [G{::}\mathbb{R} \to \mathbb{N}] \colon \emptyset \rangle . F =^? G$.

# 5 $\Sigma\mathcal{HOL}$: A Sorted Higher-Order Logic

In this section we will present a formulation $\Sigma\mathcal{HOL}$ of higher-order logic by giving the sorted $\lambda$-calculus $\Sigma\Lambda$ developed so far a logical interpretation. For this we specialize the sort system and assume the existence of certain logical constants with a fixed interpretation. In the trivially sorted case $\Sigma\mathcal{HOL}$ is a simply typed system of higher-order logic, which is essentially the Andrews/Henkin version [Hen50, And71, And72] of simple type theory. We give three notions of algebraic semantics for $\Sigma\mathcal{HOL}$: we review the notions of standard and general $\Sigma$-models and introduce the concept of $\Sigma$-model structures, which generalize Andrews' $\upsilon$-complexes [And71], and serve as a semantics for non-extensional higher-order logics.

## 5.1 The System $\Sigma\mathcal{HOL}$

**General Assumption 5.1.1** For the sorted higher-order logic $\Sigma\mathcal{HOL}$ we assume that the set $\mathcal{BT}$ of base types is $\{o, \iota\}$ where the base type $\iota$ stands for the set of **individuals** and the type $o$ for the **truth values**. A well-formed formula of type $o$ is called a **proposition**, and a closed proposition a **sentence**.

Furthermore, we assume that for any sort system $(\mathcal{S}, \mathcal{BS}, \mathfrak{d}, \mathfrak{r})$ we have a sort $\mathbb{O}$ with $\tau(\mathbb{O}) = o$. We need this sort of truth values, since types are not first-class objects of $\Sigma\mathcal{HOL}$, and we want to be able to characterize sentences and propositions by their sort.

**General Assumption 5.1.2 ($\mathbb{O}$ is Top Sort)** We assume that whenever $\mathbb{A} \in \mathcal{BS}_o$, then $[\mathbb{A} \leq \mathbb{O}] \in \Sigma$, since we want $\mathbb{O}$ to be the top sort of type $o$. Otherwise it would be possible to specify signatures that severely distort the intended semantics of the universe $\overline{\mathcal{D}}_o$, which we want to correspond to the truth values.

We could have achieved the same goal by prohibiting any other sorts of type $o$, but as we will see our more general solution provides a powerful means for specifying the semantics of predicates (5.1.6), and even allows alternate, specialized formalizations of logical calculi (5.3.15).

**Remark 5.1.3** For unsorted higher-order logics the underlying $\lambda$-calculus is usually specialized by assuming the existence of logical constants $\{q^\alpha_{\alpha\to\alpha\to o} \mid \alpha \in \mathcal{T}\} \subseteq \Omega$ for equality and $\{\neg_{o\to o}, \wedge_{o\to o\to o}, \Pi^\alpha_{(\alpha\to o)\to o} \mid \alpha \in \mathcal{T}\}$ for the connectives and quantors.

In our sorted setting we have a problem, when naively a priori assuming the existence of the logical constants, since we need infinitely many declarations of the form $[p^{\mathbb{A}}{::}\mathbb{A} \to \mathbb{A} \to \mathbb{O}]$ for logical constants $q^{\mathbb{A}}_{\alpha\to\alpha\to o}$, because there are infinitely many sorts $\mathbb{A}$. As a solution we have the alternative either to extend the definition of valid signature to encompass infinite signatures by allowing infinite $\Sigma\mathcal{HOL}$-derivations or to use the mechanism of negative variables, which is equivalent.

**Definition 5.1.4 (Logical Constants)** Let

$$\Sigma^{\Sigma\mathcal{HOL}} := \{[q^{\mathbb{A}}{::}\mathbb{A} \to \mathbb{A} \to \mathbb{O}], [\neg{::}\mathbb{O} \to \mathbb{O}], [\wedge{::}\mathbb{O} \to \mathbb{O} \to \mathbb{O}], [\Pi^{\mathbb{A}}{::}(\mathbb{A} \to \mathbb{O}) \to \mathbb{O}] \mid \mathbb{A} \in \mathcal{S}\}$$

We see that $\Sigma^{\Sigma\mathcal{HOL}}$ is a valid signature, since all these constants are distinct and can therefore be added with the rule $sig{-}const$. Moreover we can always assume our signatures to be supersets of $\Sigma^{\Sigma\mathcal{HOL}}$ by replacing the rule *sig:empty* with

$$\frac{}{\vdash_{sig} \Sigma^{\Sigma\mathcal{HOL}}}\ sig{:}empty'$$

We call the constants $q^{\mathbb{A}}, \neg, \wedge, \Pi^{\mathbb{A}}$ **logical constants**. More specifically $\neg, \wedge$ are called **connectives** and $\Pi^{\mathbb{A}}$ a **quantor** in order to distinguish them from the **equality constant** $q^{\mathbb{A}}$. Non-logical constants are called **parameters**, since the choice of parameters determines the particular formulation of the logical system $\Sigma\mathcal{HOL}$.

**Notation 5.1.5** Since the constants $q^{\mathbb{A}}$ are intended to denote the equality relation, we use $\mathbf{A} =^{\mathbb{A}} \mathbf{B}$ or even $\mathbf{A} = \mathbf{B}$ as an abbreviation for $(q^{\mathbb{A}}\mathbf{AB})$. Furthermore, we can obtain the connectives $\vee, \Rightarrow, \Leftrightarrow$ from the connectives defined so far, for instance, we take $\mathbf{A} \Rightarrow \mathbf{B}$ as an abbreviation for $\neg(\mathbf{A} \wedge \neg\mathbf{B})$. Finally, we use the infix notation for connectives, and write $\forall X_{\mathbb{A}}.\mathbf{A}$ as an abbreviation for $\Pi^{\mathbb{A}}(\lambda X_{\mathbb{A}}.\mathbf{A})$

**Example 5.1.6 (Subsorts of $\mathbb{O}$)** Let $\mathbb{T}, \mathbb{F}, \mathbb{P}, \mathbb{N}, sortz$, and $\mathbb{R}$ be sorts and $\Sigma$ be the set of the following term declarations[13]

$$[\mathbb{T} \leq \mathbb{O}], [\mathbb{F} \leq \mathbb{O}], [\neg{::}\mathbb{O} \to \mathbb{O}], [\neg|{:}\mathbb{T} \to \mathbb{F}], [\neg|{:}\mathbb{F} \to \mathbb{T}],$$
$$[\wedge{::}\mathbb{O} \to \mathbb{O} \to \mathbb{O}], [\wedge|{:}\mathbb{T} \to \mathbb{T} \to \mathbb{T}], [\wedge|{:}\mathbb{T} \to \mathbb{F} \to \mathbb{F}], [\wedge|{:}\mathbb{F} \to \mathbb{T} \to \mathbb{F}], [\wedge|{:}\mathbb{F} \to \mathbb{F} \to \mathbb{F}]$$

together with the declarations $[\Pi^{\mathbb{A}}{::}(\mathbb{A} \to \mathbb{O}) \to \mathbb{O}], [\Pi^{\mathbb{A}}{::}(\mathbb{A} \to \mathbb{T}) \to \mathbb{T}]$ for all $\mathbb{A} \in \mathcal{S}$.

Clearly $\Sigma$ gives complete semantic information for the connectives and partial information for the quantor $\Pi^{\mathbb{A}}$, if we interpret $\mathbb{T}$ as the subset $\{\mathtt{T}\} \subseteq \mathcal{D}_{\mathbb{O}}$ and $\mathbb{F}$ as $\{\mathtt{F}\} \subseteq \mathcal{D}_{\mathbb{O}}$. If the signature $\Sigma$ is augmented with term declarations

$$[\mathbb{P} \leq \mathbb{R}], [\mathbb{N} \leq \mathbb{R}], [\mathbb{Z} \leq \mathbb{R}], [1{::}\mathbb{P}], [3{::}\mathbb{P}], [-5{::}\mathbb{N}], [0{::}\mathbb{Z}],$$
$$[\leq{::}\mathbb{R} \to \mathbb{R} \to \mathbb{O}], [\forall[X{::}\mathbb{R}].X \leq X{::}\mathbb{T}],$$
$$[\leq|{:}\mathbb{P} \to \mathbb{N} \to \mathbb{F}], [\leq|{:}\mathbb{N} \to \mathbb{P} \to \mathbb{T}], [\leq|{:}\mathbb{P} \to \mathbb{Z} \to \mathbb{F}],$$
$$[+{::}\mathbb{R} \to \mathbb{R} \to \mathbb{R}], [+|{:}\mathbb{P} \to \mathbb{P} \to \mathbb{P}], [+|{:}\mathbb{N} \to \mathbb{N} \to \mathbb{N}], [+|{:}\mathbb{Z} \to \mathbb{P} \to \mathbb{P}] \ldots\}$$

for various predicates and functions on the positive ($\mathbb{P}$), negative ($\mathbb{N}$), real ($\mathbb{R}$) numbers, and zero ($\mathbb{Z}$), then formulae like $-5 + 0 \leq 3$ and $\forall X_{\mathbb{P}}. -5 \leq 3 + X$ can be seen to be of sort $\mathbb{T}$, while $\neg 0 \leq 0$ and $0 + 1 \leq 0$ have sort $\mathbb{F}$. Moreover we have $\vdash_{\Sigma} \forall P_{\mathbb{T}}.P \Rightarrow P$ and $\vdash_{\Sigma} \forall P_{\mathbb{F}}.P \Rightarrow P$. But we cannot infer $\vdash_{\Sigma} \forall P_{\mathbb{O}}.P \Rightarrow P$ in $\Sigma\mathcal{HOL}$, since the sort mechanism of $\Sigma\mathcal{HOL}$ cannot manipulate sort information that formalizes that $\mathbb{O}$ is the union of $\mathbb{T}$ and $\mathbb{F}$.

## 5.2 Σ-Model Structures

The standard example and intuitive semantic notion for sorted higher-order logic is that of a general Σ-model. This we can define from Σ-algebras by insisting on a suitable semantics for the sort $\mathbb{O}$ of truth values. Since we also want to have a semantics where the extensionality

[13]We want to remind the reader that the notation $[\mathbf{A}|{:}\mathbb{A}]$ is an abbreviation for the term declaration $[(\lambda X_{\mathfrak{d}(\mathbb{A})}.\mathbf{A}X){::}\mathbb{A}]$.

axioms can fail to be valid, we generalize the notion of a general $\Sigma$-model to the notion of a $\Sigma$-model structure. The definitions are fairly simple, since we can make use of the work we have invested in the construction of algebraic models for $\Sigma\Lambda$ and $\Lambda$.

**Definition 5.2.1 ($\Sigma$-Valuation)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a $\Sigma$-structure, then a surjective total function $\upsilon: \mathcal{D}_{\mathbb{O}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ such that

1. $\upsilon(\mathcal{I}(q^{\mathbb{A}})@a@b) = \mathtt{T}$, iff $a = b$,
2. $\upsilon(\mathcal{I}(\neg)@a) = \mathtt{T}$, iff $\upsilon(a) = \mathtt{F}$,
3. $\upsilon(\mathcal{I}(\vee)@a@b) = \mathtt{T}$, iff $\upsilon(a) = \mathtt{T}$ or $\upsilon(b) = \mathtt{T}$,
4. $\upsilon(\mathcal{I}(\Pi^{\mathbb{A}})@f) = \mathtt{T}$, iff $\upsilon(f@a) = \mathtt{T}$ for each $a \in \mathcal{D}_{\mathbb{A}}$

is called a **$\Sigma$-valuation for $\mathcal{A}$.**

The notion of $\Sigma$-valuation intuitively gives a truth-value interpretation to the domain $\mathcal{D}_{\mathbb{O}}$ of a $\Sigma$-structure, which is consistent with the intuitive interpretations of the logical constants. Since models are semantic entities that are constructed to make statements about truth and falsity of formulae, the requirement that there exists a $\Sigma$-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 $\Sigma$-structures that have $\Sigma$-valuations.

**Definition 5.2.2 ($\Sigma$-Model Structure)** Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a $\Sigma$-structure and $\upsilon$ be a $\Sigma$-valuation for $(\mathcal{D}, @, \mathcal{I})$, then we call the quadruple $\mathcal{M} := (\mathcal{D}, @, \mathcal{I}, \upsilon)$ a **$\Sigma$-model structure**. Let $\Gamma$ be a variable context, and $\varphi$ be a $\Gamma$-assignment into $\mathcal{A}$, then we call the function $\mathfrak{V}_\varphi = \upsilon \circ \mathcal{I}_\varphi: \mathit{wsf}_{\mathbb{O}}(\Sigma, \Gamma) \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ the **value function for $\mathcal{M}$ and $\varphi$**. If $\Gamma$ does not contain positive variables or the formulae in question are closed, then the value does not depend on the assignment. In these cases we drop the reference from $\mathfrak{V}_\varphi$ and call $\mathfrak{V}$ the **value function** for $\mathcal{M}$.

**Remark 5.2.3** Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon)$ be a $\Sigma$-model structure, then $\mathcal{D}_{\mathbb{O}}$ has at least two elements, since we have assumed $\upsilon: \mathcal{D}_{\mathbb{O}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ to be surjective. Furthermore, we have assumed that $\mathbb{O}$ is the top sort of type $o$, thus for all $\mathbb{A}$ of type $o$ we have $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$ and thus $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}_{\mathbb{O}}$. This guarantees that $\overline{\mathcal{D}}_o = \bigcup_{\mathbb{A} \in \mathcal{BS}} \mathcal{D}_{\mathbb{A}} = \mathcal{D}_{\mathbb{O}}$.

**General Assumption 5.2.4** For each $\mathbb{A} \in \mathcal{S}$ we assume the existence of a closed formula $\mathbf{G}^{\mathbb{A}} \in \mathit{wsf}_{\mathbb{A}}(\Sigma, \emptyset)$. This guarantees that sorts are not empty, i.e. that $\mathcal{D}_{\mathbb{A}} \neq \emptyset$ in any $\Sigma$-model structure $\mathcal{A} = (\mathcal{D}, @, \mathcal{I}, \upsilon)$, since $\mathcal{I}(\mathbf{G}^{\mathbb{A}}) \in \mathcal{D}_{\mathbb{A}}$ by definition of $\Sigma$-structures (3.3.4).

Automated deduction systems based on unification usually work with this implicit assumption, since otherwise the calculus becomes unsound: if the sort $\mathbb{A}$ is empty, the formula $[pX_{\mathbb{A}}] \wedge \neg[pX_{\mathbb{A}}]$ is contradictory but at the same time satisfiable.

Note that it is sufficient to assume a closed formula $\mathbf{G}^{\mathbb{A}}$ for all $\mathbb{A} \in \mathcal{BS}$, since we can choose $\mathbf{G}^{\mathbb{B} \to \mathbb{A}} := (\lambda X_{\mathbb{B}}.\mathbf{G}^{\mathbb{A}})$. Therefore each well-sorted formula $\mathbf{A} \in \mathit{wsf}_{\mathbb{A}}(\Sigma, \Gamma)$ has a closed $\Sigma$-instance, that is, there exists a $\Sigma$-substitution $\sigma \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$ such that $\sigma(\mathbf{A})$ is a closed formula. In this case the contradiction above really is unsatisfiable.

**Lemma 5.2.5** *Let $\mathcal{M}$ be a Σ-model structure and $\mathfrak{V}$ the value function for $\mathcal{M}$, furthermore, let $\top_o := (q^{\mathbb{A}} = q^{\mathbb{A}})$ and $\bot_o := q^{\mathbb{O}\to\mathbb{O}}(\lambda X_{\mathbb{O}}.\top_o)(\lambda X_{\mathbb{O}}.X)$, then $\mathfrak{V}(\top_o) = \mathtt{T}$ and $\mathfrak{V}(\bot_o) = \mathtt{F}$.*

**Proof:** Note that $\mathcal{I}(\top_o) = \mathcal{I}(q^{\mathbb{A}\to\mathbb{A}\to\mathbb{O}}q^{\mathbb{A}}q^{\mathbb{A}}) = \mathcal{I}(q)@\mathcal{I}(q)@\mathcal{I}(q)$, so $\mathfrak{V}(\top_o) = \upsilon(\mathcal{I}(\top_o)) = \mathtt{T}$ by definition. Furthermore, $\mathfrak{V}(\bot_o) = \upsilon(\mathcal{I}_\varphi(q(\lambda X.\top_o)(\lambda X.X))) = \mathtt{T}$, iff $\mathcal{I}_\varphi(\lambda X.\top_o) = \mathcal{I}_\varphi(\lambda X.X)$, since $\upsilon$ is a Σ-valuation. This is the case exactly, iff $\mathcal{I}(\lambda X.\top_o)@a = \mathcal{I}(\lambda X.X)@a$ for all $a \in \mathcal{D}_{\mathbb{A}}$, since $\mathcal{M}$ is functional. Finally, this is equivalent to $\mathcal{I}(\top_o) = \mathcal{I}_{[a/X]}(X) = a$, which is obviously not the case, since $\mathcal{D}_{\mathbb{O}}$ has at least two elements by 5.2.3. □

Thus we can define expressions for truth ($\top_o$) and falsity ($\bot_o$) in ΣHOL from $q^{\mathbb{A}}$ that obtain the intended meaning in all Σ-model structures. We use them just like logical constants in the following. Similarly we can define equality from the connectives and quantors.

**Definition 5.2.6 (Leibniz' Formulation for Equality)** We define the **Leibniz formula** for equality by

$$\mathbf{Q}^{\mathbb{A}} := (\lambda X_{\mathbb{A}} Y_{\mathbb{A}}.\forall P_{\mathbb{A}\to\mathbb{O}}.PX \Rightarrow PY)$$

With this definition the formula $(\mathbf{A} = \mathbf{B}) \doteq \mathbf{Q}^{\mathbb{A}}\mathbf{A}\mathbf{B}$ $\beta$-reduces to $\forall P_{\mathbb{A}\to\mathbb{O}}.(P\mathbf{A}) \Rightarrow (P\mathbf{B})$, which can be read as: formulae $\mathbf{A}$ and $\mathbf{B}$ are not equal, iff there exists a discerning property $P$. In other words, $\mathbf{A}$ and $\mathbf{B}$ are equal, if they are indiscernible. We semantically justify this definition by 5.2.7.

**Lemma 5.2.7** *Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon)$ be a Σ-model structure, and let $\mathbf{Q}^{\mathbb{A}}$ be defined as in* 5.2.6*, then $\mathfrak{V}_\varphi(\mathbf{Q}^{\mathbb{A}}\mathbf{A}\mathbf{B}) = \mathtt{T}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$.*

**Proof:** Let $a, b \in \mathcal{D}_{\mathbb{A}}$, we show that $\upsilon(\mathcal{I}_\varphi(\mathbf{Q})@a@b) = \mathtt{T}$, iff $a = b$, which entails the assertion. We have $\mathcal{I}_\varphi(\mathbf{Q}) = \mathcal{I}_\varphi(\lambda X.\lambda Y.\forall P.PX \Rightarrow PY)$ by definition 5.2.6, and thus $\mathcal{I}_\varphi(\mathbf{Q}^{\mathbb{A}})@a@b = \mathcal{I}_\psi(\forall P.PX \Rightarrow PY)$, if $\psi = \varphi, [a/X], [b/Y]$. Now let $r \in \mathcal{D}_{\mathbb{A}\to\mathbb{O}}$, then

$$\upsilon(\mathcal{I}_{\psi,[r/P]}(PX)) = r@a = \mathtt{F} \qquad \text{or} \qquad \upsilon(\mathcal{I}_{\psi,[r/P]}(PY)) = r@a = \mathtt{T}\ ,$$

since $\upsilon$ is total. So we see that $\upsilon(\mathcal{I}_\varphi(\mathbf{Q})@a@a) = \upsilon(\mathcal{I}_{\psi,[r/P]}(PX \Rightarrow PY)) = \mathtt{T}$ for all $r \in \mathcal{D}_{\mathbb{A}\to\mathbb{O}}$, which yields the assertion.

Now let $a \neq b \in \mathcal{D}_{\mathbb{A}}$ and $r = \mathcal{I}_{\varphi,[a/X]}(\lambda Y_{\mathbb{A}}.q^{\mathbb{A}}XY)$, then

$$\upsilon(r@a) = \mathcal{I}_{\varphi,[a/X],[a/Y]}(q^{\mathbb{A}}XY) = \mathtt{T} \qquad \text{and} \qquad \upsilon(r@b) = \mathcal{I}_\psi(q^{\mathbb{A}}XY) = \mathtt{F}\ .$$

Thus $\upsilon(\mathcal{I}_\varphi(\mathbf{Q})@a@b) = \upsilon(\mathcal{I}_\psi(\forall P.PX \Rightarrow PY)) = \mathtt{F}$, since $\upsilon(\mathcal{I}_{\psi,[r/P]}(PX \Rightarrow PY)) = \mathtt{F}$, as $\upsilon(\mathcal{I}_{\psi,[r/P]}(PX)) = r@a\mathtt{T}$ and $\upsilon(\mathcal{I}_{\psi,[r/P]}(PY)) = r@b = \mathtt{F}$. □

**Remark 5.2.8** The previous lemma shows that in definition 5.1.4 we can indeed use the Leibniz property to treat equality as a defined notion, if we take care to ensure that our Σ-model structures contain the identity relation for each sort. Thus we would principally not have needed to assume the constants $q^{\mathbb{A}}$ in our signature. The critical part in this choice is that for ensuring the correct meaning for $\mathbf{Q}^{\mathbb{A}}$ we have to require the existence of the identity relation for each sort in each Σ-model structure (see [And72] for a discussion in the context of unsorted general models). This requirement is automatically met, if we have constants $q^{\mathbb{A}} \in \overline{\Sigma}$, so it seems natural to treat equality as primitive.

We now present two special classes of $\Sigma$-model structures, which model the intended understanding of $\Sigma\mathcal{HOL}$. The class of standard $\Sigma$-models is in some way the most natural notion of semantics for $\Sigma\mathcal{HOL}$, however, with the notion of completeness induced with this semantics there cannot be complete calculi, a fact that makes it virtually useless for our purposes. The class of general $\Sigma$-models allows complete calculi and, in fact, we will exhibit one later in this section (see subsections 5.3 and 5.5). Unfortunately, it will turn out that our resolution calculus is not complete even with respect to general $\Sigma$-models, therefore we cannot restrict our presentation to this semantics, but have to take the more general notion of $\Sigma$-model structures.

**Definition 5.2.9 (General $\Sigma$-Model)** Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon)$ be a $\Sigma$-model structure such that $\mathcal{A} = (\mathcal{D}, \mathcal{I})$ is a $\Sigma$-algebra, then $\mathcal{M}$ is called a **general $\Sigma$-model**, iff $\mathcal{D}_{\mathbb{O}}$ is the set $\{\mathtt{T}, \mathtt{F}\}$ of truth values. Note that with this definition $\upsilon$ must be the identity function on $\mathcal{D}$, moreover @ is just function application. Thus @ and $\upsilon$ are fixed in general $\Sigma$-models, and we can fully describe $\mathcal{M}$ by its carrier set $\mathcal{D}$ and its interpretation $\mathcal{I}$. We are striving for a general notion of algebraic model, so we only require $\mathcal{M}$ to be comprehension-closed ($\mathcal{A}$ is a $\Sigma$-algebra) and do not require $\mathcal{M}$ to be full. A full general $\Sigma$-model is called a **standard $\Sigma$-model**.

Now we define the usual notions of satisfiability and validity. However, since we have more than one notion of semantics, we have to take care to specify the intended semantics.

**Definition 5.2.10** Let $\mathcal{K}$ be a class of $\Sigma$-model structures, $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon) \in \mathcal{K}$, $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{O}$, and $\varphi$ a $\Gamma$-assignment into $\mathcal{M}$. We say that

1. $\varphi$ **satisfies A in** $\mathcal{M}$ ($\mathcal{M} \models_\varphi \mathbf{A}$), iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathtt{T}$.

2. $\mathbf{A}$ is **satisfiable in** $\mathcal{M}$, iff there is an assignment $\varphi$ that satisfies $\mathbf{A}$ in $\mathcal{M}$.

3. $\mathbf{A}$ is **satisfiable in** $\mathcal{K}$, iff there is a $\Sigma$-model structure $\mathcal{M} \in \mathcal{K}$ such that $\mathbf{A}$ is satisfiable in $\mathcal{M}$.

4. $\mathbf{A}$ is **valid in** $\mathcal{M}$ ($\mathcal{M} \models \mathbf{A}$), iff all assignments into $\mathcal{M}$ satisfy $\mathbf{A}$ in $\mathcal{M}$.

5. $\mathbf{A}$ is **valid in** $\mathcal{K}$ ($\models_\mathcal{K} \mathbf{A}$), iff $\mathbf{A}$ is valid in all $\mathcal{M} \in \mathcal{K}$.

We say a proposition $\mathbf{A}$ **entails** a proposition $\mathbf{B}$ **in** $\mathcal{K}$ ($\mathbf{A} \models_\mathcal{K} \mathbf{B}$), iff for all $\mathcal{M} \in \mathcal{K}$ we have that $\mathcal{M} \models \mathbf{A}$ implies $\mathcal{M} \models \mathbf{B}$.

**Remark 5.2.11** If we were only interested in analyzing $\Sigma\mathcal{HOL}$ with respect to the general $\Sigma$-model semantics, we could have simplified the presentation of the theory by only assuming the equality constants, defining the other logical constants by the following definitions, and treating them as defined formulae: $\neg := (q^{\mathbb{O}} \bot_o)$, $\Pi^{\mathbb{A}} := (q^{\mathbb{A}\to\mathbb{O}}(\lambda X_{\mathbb{A}}.\top_o))$ and $\wedge := (\lambda X_{\mathbb{O}} Y_{\mathbb{O}}.(\lambda G.G\top_o\top_o) = (\lambda G.GXY))$.

Furthermore, we could have weakened the conditions for $\upsilon$ to be a $\Sigma$-valuation by only requiring that $\upsilon \circ \mathcal{I}(q^{\mathbb{A}})$ is the identity relation on $\mathcal{D}_{\mathbb{A}}$, since the definitions above entail that $\upsilon := \mathrm{Id}_{\mathcal{D}_{\mathbb{O}}}$ is a $\Sigma$-valuation. Let us consider the cases of 5.2.1.

1. This case is trivial, since $\mathcal{I}(q^{\mathbb{A}})$ is the identity relation on $\mathcal{D}_{\mathbb{A}}$ by definition.

2. We have $\mathcal{I}(\neg)@a = \mathcal{I}(q^{\mathbb{O}}\top_o)@a = \mathcal{I}(q^o)@\mathtt{F}@a = \mathtt{T}$, iff $a = \mathtt{F}$, since $\mathcal{D}_{\mathbb{O}} = \{\mathtt{T}, \mathtt{F}\}$.

3. $\mathcal{I}(\Pi^{\mathbb{A}})@f = \mathcal{I}(q^{\mathbb{A}\to\mathbb{O}}(\lambda X_{\mathbb{A}}.\top_o)@f = \mathtt{T}$, iff $\mathcal{I}(\lambda X_{\mathbb{A}}.\top_o) = f$, iff $\mathtt{T} = \mathcal{I}(\top_o) = \mathcal{I}(\lambda X.\top_o)@a = f@a$ for all $a \in \mathcal{D}_{\mathbb{A}}$, since $\mathcal{M}$ is functional.

4. Let $a, b \in \mathcal{D}_{\mathbb{O}}$ and $\varphi = [a/X], [b/Y]$, then $\mathcal{I}(\wedge)@a@b = \mathtt{T}$, iff $\mathcal{I}_\varphi(\lambda G.G\top_o\top_o) = \mathcal{I}_\varphi(\lambda G.GXY)$, which is the case, iff for all $g \in \mathcal{D}_{\mathbb{O}\to\mathbb{O}\to\mathbb{O}}$ we have $g@\mathtt{T}@\mathtt{T} = g@a@b$, since $\mathcal{M}$ is functional. This is clearly equivalent to the condition that $a = \mathtt{T} = b$.

Note that a construction like the one above is not possible in the case of Σ-model structures, since the proof of the condition for $\neg$ requires that $\mathcal{D}_{\mathbb{O}}$ has exactly the elements $\mathtt{T}$ and $\mathtt{F}$, as we take all elements that are not $\mathtt{T}$ to be false.

**Remark 5.2.12** Let $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ and $\mathcal{N} = (\mathcal{E}, \mathcal{J})$ be general Σ-models, and let $\kappa: \mathcal{M} \longrightarrow \mathcal{N}$ be a Σ-homomorphism, then by 5.2.5 and 2.1.11(2) we have $\kappa(\mathtt{T}) = \kappa(\mathcal{I}_\varphi(\top_o)) = \mathcal{J}_{\kappa\circ\varphi}(\top_o) = \mathtt{T}$ and similarly $\kappa(\mathtt{F}) = \mathtt{F}$.

**Remark 5.2.13** Note that the class of general Σ-models defined above is rich in non-standard models[14], since we do not require it to contain a description function. In this detail we differ from most systems of higher-order logic (cf. [Rus08, Chu40, Hen50, And71, And86]), which do require the existence of a constant $\iota^{\mathbb{A}} \in \Sigma_{(\mathbb{A}\to\mathbb{O})\to\mathbb{A}}$ for each type $\mathbb{A}$. Correspondingly these approaches require that this constant denotes the function that maps each singleton set to its unique member in (general) Σ-models by requiring an axiom

$$\forall P_{\mathbb{A}\to\mathbb{O}}.(\exists X_{\mathbb{A}}.PX) \wedge (\forall Y_{\mathfrak{d}(\mathbb{A})}.PX \Rightarrow .X = Y) \Rightarrow P(\iota P)$$

Even though our's may not be the most interesting notion of general Σ-model, we choose not to deal with descriptions in this thesis, which focuses on treating sorted methods in a resolution context.

**Definition 5.2.14 (Full Extensionality)** We call the following formula schemata

$$\begin{array}{lcl} \mathbf{Ext}^{\mathbb{A}} & \doteq & \forall F_{\mathbb{A}}.\forall G_{\mathbb{A}}(\forall X_{\mathfrak{d}(\mathbb{A})}.FX = GX) \Rightarrow F = G \\ \mathbf{Ext}^{\mathbb{O}} & \doteq & \forall F_{\mathbb{O}}.\forall G_{\mathbb{O}}.(F \Leftrightarrow G) \Leftrightarrow F = G \end{array}$$

the **axioms of full extensionality** and we specifically refer to the latter formula as the **extensionality axiom for truth values**.

We now analyze the validity of the extensionality axioms in our notions of semantics.

**Lemma 5.2.15** *Let $\mathcal{M}$ be a Σ-model structure and $\mathbb{A} \in \mathcal{S}^{nf}$, then $\mathcal{M} \models \mathbf{Ext}^{\mathbb{A}}$.*

**Proof:** The validity of $\mathbf{Ext}^{\mathbb{A}}$ is a consequence of the functionality we have assumed for Σ-algebras. □

**Lemma 5.2.16** *The axiom $\mathbf{Ext}^{\mathbb{O}}$ is not valid in the class of Σ-model structures.*

[14] In our notion of general Σ-model we cannot guarantee the existence of, for instance, step functions, i.e. functions $f \in \mathcal{F}(\mathcal{D}_{\mathbb{A}}; \mathcal{D}_{\mathbb{B}})$ that are constant on sets in $\mathcal{D}_{\mathbb{A}\to\mathbb{O}}$.

**Proof:** Let $(\mathcal{D}, @, \mathcal{I})$ be any standard $\Sigma$-algebra with $\mathcal{D}_{\mathbb{O}} = \{a, b, c\}$, and let $\upsilon(a) = \upsilon(b) =$ T and $\upsilon(c) =$ F. Furthermore, let the interpretation function $\mathcal{I}$ behave on connectives and quantors as indicated by the following schemata:

| $\mathcal{I}(\neg)$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| | $c$ | $c$ | $a$ |

| $\mathcal{I}(\wedge)$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $a$ | $a$ | $a$ | $c$ |
| $b$ | $a$ | $a$ | $c$ |
| $c$ | $c$ | $c$ | $c$ |

$$\mathcal{I}(\Pi^{\mathbb{A}})@f = \begin{cases} a, & \text{if } f@g \in \{a,b\} \text{ for all } g \in \mathcal{D}_{\mathbb{A}} \\ b, & \text{if } f@g = c \text{ for some } g \in \mathcal{D}_{\mathbb{A}} \end{cases}$$

Let $\varphi(X) = a$ and $\varphi(Y) = b$, then we can see that $\mathcal{I}_\varphi(X \Rightarrow Y) = a$ and $\mathcal{I}_\varphi(X \Leftarrow Y) = a$, thus $\mathcal{I}_\varphi(X \Leftrightarrow Y) = a$, but $\mathcal{I}(X = Y) = c$. So we have $\mathcal{I}_\varphi(\forall X, Y_{\mathbb{O}}.X \Leftrightarrow Y.X = Y) = c$, and thus $\mathfrak{V}_\varphi(\forall X, Y_{\mathbb{O}}.X \Leftrightarrow Y.X = Y) =$ F. □

**Lemma 5.2.17** *Let $\mathcal{M}$ be a general $\Sigma$-model, then $\mathcal{M} \models$ $\mathbf{Ext}^{\mathbb{O}}$.*

**Proof:** The validity of the axiom for truth values is a consequence of 5.2.5 and the definition of $\Sigma$-valuation. □

As we have seen, full extensionality is valid in general $\Sigma$-models, unfortunately, it will turn out that our resolution calculus will not be able to handle full extensionality. This is not a problem special to our sorted system, since this is also a problem for the unsorted calculus of higher-order resolution [Hue72] (or for higher-order mating-search [And89]). These papers give relative completeness results like our theorem 6.4.6, we additionally give an algebraic semantics that describes the derivational power of our calculi.

## 5.3 Calculi

In this section we introduce the syntactic counterparts of the entailment relation and fix the formalism for the calculi $\Sigma\mathfrak{T}$, $\Sigma\mathfrak{T}\eta$, and $\Sigma\mathfrak{T}\mathfrak{E}$, which are simple and intuitive generalizations of a commonly used Hilbert-style calculus for first-order logic, and characterize their deductive power in terms of our semantics. In the literature [And71, Hue72, Mil83] the deductive power of machine-oriented calculi is determined relative to that of the unsorted versions of these calculi. Analogously we compare the deductive power of the sorted versions with that of our sorted resolution calculus $\Sigma\mathcal{HR}$, and we obtain the analogous results.

**Definition 5.3.1 (Calculus)** A **calculus** $\mathcal{C}$ is an inference system for the judgment $\Vdash \mathbf{A}$, i.e. $\mathbf{A}$ is **provable** in $\mathcal{C}$. Since $\Vdash$ is a unary relation, it is customary to drop it, and to consider inference rules of a calculus $\mathcal{C}$ just as relations over propositions.

Let $\mathbf{A}$ be a proposition and $\Phi$ be a set of sentences. Using the nomenclature for inference systems we call a $\mathcal{C}$-derivation $\mathcal{D}$ a **$\mathcal{C}$-proof of A from the set $\Phi$ of hypotheses**, if $\mathbf{A}$ is the assertion of the root of $\mathcal{D}$, and the supports of the leaves of $\mathcal{D}$ are subsets of $\Phi$. If there exists a $\mathcal{C}$-derivation of $\mathbf{A}$ from $\Phi$, then we write $\Phi \Vdash_{\mathcal{C}} \mathbf{A}$. Let $\mathcal{C}$ be a calculus, then a proposition $\mathbf{A}$ is called a **theorem of $\mathcal{C}$**, iff there exists a $\mathcal{C}$-derivation of $\mathbf{A}$ from the empty set of hypotheses.

**Definition 5.3.2 ($\mathcal{C}$-Consistent)** Let $\mathcal{C}$ be a calculus, then a set $\Phi$ of propositions is called **$\mathcal{C}$-inconsistent**, iff $\Phi \Vdash_{\mathcal{C}} \bot_o$, and **$\mathcal{C}$-consistent** otherwise. We call a set $\Psi$ **$\mathcal{C}$-consistent with** a set $\Phi$, iff $\Phi \cup \Psi$ is $\mathcal{C}$-consistent.

**Lemma 5.3.3** *If $\mathcal{C}$ is a calculus, and if $\Phi$ is a $\mathcal{C}$-inconsistent set of propositions, then there exists a finite $\mathcal{C}$-inconsistent subset of $\Phi$.*

**Proof:** Let $\mathcal{D}$ be a $\mathcal{C}$-derivation of $\bot_o$ from $\Phi$. As $\mathcal{D}$ is a finite tree, the set $\Psi \subseteq \Phi$ of labels of the leaves of $\mathcal{D}$ is finite. Thus $\Psi$ is a $\mathcal{C}$-inconsistent and finite subset of $\Psi$. □

**Definition 5.3.4 (Sound, Complete, Saturated)** Let $\mathcal{C}$ be a calculus and $\mathcal{K}$ a class of $\Sigma$-model structures, then

1. $\mathcal{C}$ is called **sound with respect to $\mathcal{K}$**, iff each theorem of $\mathcal{C}$ is valid in $\mathcal{K}$.
2. $\mathcal{C}$ is called **complete with respect to $\mathcal{K}$**, iff each valid sentence in $\mathcal{K}$ is a theorem of $\mathcal{C}$.
3. $\mathcal{C}$ is called **refutation complete with respect to $\mathcal{K}$**, iff each $\mathcal{K}$-unsatisfiable sentence $\mathbf{A}$ can be refuted in $\mathcal{C}$, i.e. there is a $\mathcal{C}$-derivation of an elementary contradiction from $\mathbf{A}$.
4. $\mathcal{C}$ is called **saturated**, iff for all $\mathcal{C}$-consistent sets $\Phi$ of propositions we have $\Phi * \mathbf{A}$ is $\mathcal{C}$-consistent or $\Phi * \neg\mathbf{A}$ is $\mathcal{C}$-consistent.

**Definition 5.3.5 (The Calculi $\Sigma\mathfrak{T}$, $\Sigma\mathfrak{T}\eta$ and $\Sigma\mathfrak{TE}$)** The calculus $\Sigma\mathfrak{T}$ consists of the following **propositional axiom** schemata:

$$\frac{\Gamma(P) = \mathbb{O}}{\Gamma \Vdash_\Sigma (P \vee P) \Rightarrow P} \qquad \frac{\Gamma(P) = \mathbb{O}}{\Gamma \Vdash_\Sigma (P \Rightarrow (P \Rightarrow P)}$$

$$\frac{\Gamma(P) = \Gamma(Q) = \mathbb{O}}{\Gamma \Vdash_\Sigma (P \vee Q) \Rightarrow (Q \vee P)} \qquad \frac{\Gamma(P) = \Gamma(Q) = \Gamma(R) = \mathbb{O}}{(P \Rightarrow Q) \Rightarrow .(R \vee P) \Rightarrow (R \vee Q)}$$

$$\frac{\Gamma(F) = \mathbb{A} \to \mathbb{O} \quad \Gamma(X) = \mathfrak{d}(\mathbb{A})}{\Pi^{\mathfrak{d}(\mathbb{A})} F \Rightarrow FX} \qquad \frac{\Gamma(P) = \mathbb{O} \quad \Gamma(F) = \mathbb{A} \to \mathbb{O}}{\forall X_{\mathbb{A}} (P \vee FX) \Rightarrow .P \vee \Pi^{\mathbb{A}} F}$$

The **$\beta$-conversion** ($\Sigma\mathfrak{T}(\beta)$), **modus ponens** ($\Sigma\mathfrak{T}(MP)$), **$\Sigma$-substitution** ($\Sigma\mathfrak{T}(Subst)$), and **universal generalization** ($\Sigma\mathfrak{T}(UG)$) inference rules

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} {=}_\beta \mathbf{B} \quad \Gamma \Vdash_\Sigma \mathbf{A}}{\Gamma \Vdash_\Sigma \mathbf{B}} \Sigma\mathfrak{T}(\beta) \qquad \frac{\Gamma \Vdash_\Sigma \mathbf{A} \Rightarrow \mathbf{B} \quad \Gamma \Vdash_\Sigma \mathbf{A}}{\Gamma \Vdash_\Sigma \mathbf{B}} \Sigma\mathfrak{T}(MP)$$

$$\frac{\Gamma, [X :: \mathbb{A}] \Vdash_\Sigma \mathbf{A}X \quad \Gamma \vdash_\Sigma \mathbf{B} :: \mathbb{A}}{\Gamma \Vdash_\Sigma \mathbf{AB}} \Sigma\mathfrak{T}(Subst) \qquad \frac{\Gamma, [X :: \mathbb{A}] \Vdash_\Sigma \mathbf{A}X}{\Gamma \Vdash_\Sigma \Pi^{\mathbb{A}} \mathbf{A}} \Sigma\mathfrak{T}(UG)$$

We obtain the calculus $\Sigma\mathfrak{T}\eta$ by adding the $\eta$-conversion rule

$$\frac{\Gamma \vdash_{\Sigma} \mathbf{A} =_{\eta} \mathbf{B} \quad \Gamma \Vdash_{\Sigma} \mathbf{A}}{\Gamma \Vdash_{\Sigma} \mathbf{B}} \, \Sigma\mathfrak{T}(\eta)$$

and, finally, $\Sigma\mathfrak{T}\mathfrak{E}$ with the following rule for full extensionality

$$\frac{\Gamma \Vdash_{\Sigma} \mathbf{A} \Leftrightarrow \mathbf{B}}{\Gamma \Vdash_{\Sigma} \mathbf{A} = \mathbf{B}} \, \Sigma\mathfrak{T}\mathfrak{E}(TW)$$

These rules correspond to the $\mathbf{Ext}^{\mathbb{A}}$ and $\mathbf{Ext}^{\mathbb{O}}$ axioms. We collectively denote these calculi with $\Sigma\mathfrak{T}*$. Note that if $\Gamma \Vdash_{\Sigma} \mathbf{A}$, then we have $\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{O}$, since we have taken care to require this for the propositional axioms, and the proper inference rules conserve this property.

**Theorem 5.3.6 (Soundness)** *$\Sigma\mathfrak{T}$ and $\Sigma\mathfrak{T}\eta$ are sound with respect to the class of $\Sigma$-model structures. Moreover, $\Sigma\mathfrak{T}*$ are sound with respect to the classes of general $\Sigma$-models and standard $\Sigma$-models.*

**Proof:** The inference rules $\Sigma\mathfrak{T}(\beta)$ and $\Sigma\mathfrak{T}(\eta)$ are sound in all $\Sigma$-structures and therefore in all $\Sigma$-model structures. The validity of the propositional axioms $\Sigma\mathfrak{T}(MP)$ and $\Sigma\mathfrak{T}(UG)$ is an immediate consequence of the definition of $\Sigma$-valuations. For the $\Sigma\mathfrak{T}(Subst)$ rule let $\Gamma, [X{::}\mathbb{A}] \vdash_{\Sigma} \mathbf{A}{::}\mathbb{O}$, $\Gamma' \vdash_{\Sigma} \mathbf{B}{::}\mathbb{A}$ and $\mathcal{M} \models \mathbf{A}$, then by the substitution value theorem (3.3.8) we have $v(\mathcal{I}_{\varphi}([\mathbf{B}/X]\mathbf{A})) = v(\mathcal{I}_{\varphi,[\mathcal{I}_{\varphi}(\mathbf{B})/X]}(\mathbf{A}) = \mathtt{T}$ for all $\Gamma$-assignments $\varphi$. Thus all inference rules are sound.

We obtain the second assertion by 5.2.17 and the fact that the classes of general and standard $\Sigma$-models are subclasses of that of $\Sigma$-model structures. $\square$

We now present two results on provability, which simplifies the analysis of the $\Sigma\mathfrak{T}*$ calculi.

**Definition 5.3.7** We call a proposition $\mathbf{A}$ a **tautology**, iff it is a substitution instance of a proposition $\mathbf{P}$ that only contains logical connectives and variables of sort $\mathbb{O}$, and is valid in all $\Sigma$-model structures. Note that the validity of $\mathbf{P}$ only depends on the assignment for propositional variables in $\mathbf{P}$.

**Lemma 5.3.8 (Rule P)** *If $\mathbf{A}$ is a tautology, then $\Vdash_{\mathcal{C}} \mathbf{A}$ for any $\Sigma\mathfrak{T}*$-calculus $\mathcal{C}$.*

**Proof:** Let $\mathbf{P}$ be the proposition such that $\mathbf{A} = \sigma(\mathbf{P})$, and $\mathbf{P}$ only contains propositional variables and connectives. It is well-known that the propositional part of $\Sigma\mathfrak{T}*$ is complete (see for instance [And86]), so there is a $\Sigma\mathfrak{T}*$-proof $\mathcal{D}$: $\Vdash_{\Sigma\mathfrak{T}*} \mathbf{P}$. One application of the substitution rule now gives the assertion. $\square$

**Theorem 5.3.9 (Deduction Theorem)** *If $\mathcal{H}, \mathbf{A} \Vdash_{\mathcal{C}} \mathbf{B}$, then $\mathcal{H} \Vdash_{\mathcal{C}} \mathbf{A} \Rightarrow \mathbf{B}$ where $\mathcal{C} \in \Sigma\mathfrak{T}*$.*

**Proof:** We refer to lemma 5240 in [And86] and to [Hen50]. $\square$

**Notation 5.3.10** For reasons of legibility we will write $S * a$ for $S \cup \{a\}$, where $S$ is a set. We will use this notation with the convention that $*$ associates to the left.

**Lemma 5.3.11** *All $\Sigma\mathfrak{T}*$ calculi are saturated.*

**Proof:** Let $\mathcal{C}$ be a $\Sigma\mathfrak{T}*$ calculus. To see that $\mathcal{C}$ is saturated, let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathcal{C}$-inconsistent, then we show that $\Phi$ is $\mathcal{C}$-inconsistent, and obtain the assertion. By definition $\Phi * \mathbf{A} \Vdash \perp_o$, and by the deduction theorem $\Phi \Vdash_{\mathcal{C}} \mathbf{A} \Rightarrow \perp_o$, and thus $\Phi \Vdash_{\mathcal{C}} \neg\mathbf{A}$ by rule $P$, similarly $\Phi \Vdash_{\mathcal{C}} \mathbf{A}$ and thus $\Phi$ is $\mathcal{C}$-inconsistent. $\square$

**Remark 5.3.12** If we look more closely at the proof, we see that we only need the deduction theorem and rule $P$, thus any calculus $\mathcal{C}$ that admits the deduction theorem and rule $P$ is saturated. Therefore saturatedness is a natural property for calculi and thus for abstract consistency classes, since it is a direct consequence of the deduction theorem.

**Theorem 5.3.13** *Let $\Phi$ be a set of propositions, and furthermore, let*

1. *$\mathcal{K}$ be a class of $\Sigma$-model structures and $\mathcal{C} = \Sigma\mathfrak{T}\eta$, or*
2. *$\mathcal{K}$ be a class of $\Sigma$-models and $\mathcal{C} = \Sigma\mathfrak{T}\mathfrak{E}$,*

*then the following assertions hold:*

1. *If $\mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash_{\mathcal{C}} \mathbf{B}$ such that $\mathbf{B}$ is false in some $\mathcal{M} \in \mathcal{K}$, then $\mathcal{M} \models \bigvee_{i=1}^{n}(\neg\mathbf{A}^i)$.*
2. *If $\mathcal{M} \in \mathcal{K}$, then $\Psi := \{\mathbf{A} \in wsf_{\mathbb{O}}(\Sigma, \Gamma) \mid \mathcal{M} \models \mathbf{A}\}$ is $\mathcal{C}$-consistent.*
3. *$\Phi$ is $\mathcal{C}$-inconsistent with $\mathbf{A}^1, \ldots, \mathbf{A}^n$, iff $\Phi \Vdash_{\mathcal{C}} \bigvee_{i=1}^{n}(\neg\mathbf{A}^i)$.*

**Proof:** By 5.3.6 we know that $\mathcal{C}$ is sound with respect to $\mathcal{K}$. We show the first assertion by induction on the size of the $\mathcal{C}$-derivation $\mathcal{D}$ of $\mathbf{B}$ from $\Phi$. As $\mathcal{C}$ is sound with respect to $\mathcal{K}$, the proposition $\perp_o$ cannot be an axiom of $\mathcal{C}$, thus the base case is vacuously true. For the inductive case let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon) \in \mathcal{K}$ be a $\Sigma$-model structure such that $\mathfrak{V}_\varphi(\mathbf{B}) = \mathtt{F}$, and let $\mathcal{D}$ end in an application of the rule $\mathcal{R} := \mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash \mathbf{B} \in \mathcal{C}$. Since $\mathcal{C}$ is sound with respect to $\mathcal{K}$, we have $\mathfrak{V}_\varphi(\mathbf{A}^1, \ldots, \mathbf{A}^n \Rightarrow \mathbf{B}) = \mathtt{T}$ by modus ponens, and therefore $\mathfrak{V}_\varphi(\bigvee_{i=1}^{n}(\neg\mathbf{A}^i)) = \mathtt{T}$. Therefore one of the premises of $\mathcal{R}$ is false in $\mathcal{M}$, and by induction we get the assertion.

To prove the second assertion consider the contrapositive statement: let $\Psi$ be $\mathcal{C}$-inconsistent and $\Phi := \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ be a $\mathcal{C}$-inconsistent subset of $\Psi$, then $\mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash_{\mathcal{C}} \perp_o$, and therefore one of the $\mathbf{A}^i$ is false in $\mathcal{M}$ by the first assertion, which contradicts the assumption.

For the third assertion we note that, if $\Phi$ is $\mathcal{C}$-inconsistent with $\mathbf{A}^1, \ldots, \mathbf{A}^n$, then $\Phi, A^1, \ldots, \mathbf{A}^n \Vdash_{\mathcal{C}} \neg\mathbf{A}^1$, so $\Phi \Vdash_{\mathcal{C}} \mathbf{A}^1 \Rightarrow \ldots\ldots \Rightarrow \mathbf{A}^n \Rightarrow \neg\mathbf{A}^1$, and therefore $\Phi \Vdash_{\mathcal{C}} \bigvee_{i=1}^{n}(\neg\mathbf{A}^i)$. The other direction is immediate. $\square$

**Lemma 5.3.14** *$\Sigma\mathfrak{T}$ and $\Sigma\mathfrak{T}\eta$ are not complete with respect to general $\Sigma$-models.*

**Proof sketch:** We do not have the means of proving this lemma here, since the argumentation involves the use of the resolution calculus $\Sigma\mathcal{HR}$ presented in section 6, so we only sketch the proof and refer to 6.4.7. Let $\Sigma := \{[c::\mathbb{O} \to \mathbb{O}], [b::\mathbb{O}]\}$, $\mathbf{A} := (cb)$, and $\mathbf{B} := c(\neg\neg b)$. The proof of the assertion has three ingredients:

- $\mathbf{C} := \neg\mathbf{A} \wedge \mathbf{B}$ is not $\Sigma\mathcal{HR}$-refutable.
- $\Sigma\mathfrak{T}\eta$ is sound and refutation-complete with respect to $\Sigma$-model structures (see 5.5.7).
- and $\Sigma\mathcal{HR}$ is sound and complete with respect to $\Sigma$-model structures (6.4.4).

$\neg\mathbf{C}$ cannot be derivable in $\Sigma\mathfrak{T}\eta$, since otherwise $\mathbf{C}$ could be refuted in $\Sigma\mathcal{HR}$, as $\Sigma\mathcal{HR}$ is refutation complete. The same argumentation holds for $\Sigma\mathfrak{T}$, since it is weaker lacking the $\eta$-axiom of $\Sigma\mathfrak{T}$. □

**Example 5.3.15 ($\Sigma\mathfrak{T}$ with Sorts)** We can give an alternative formalization of the calculus $\Sigma\mathfrak{T}$ using term declarations: if we augment the signature from 5.1.6 with the declarations

$$\begin{array}{l}
[\forall[P::\mathbb{O}].(P \vee P) \Rightarrow P::\mathbb{O}], \\
[\forall[P::\mathbb{O}].(P \Rightarrow (P \Rightarrow P)::\mathbb{O}], \\
[\forall[P,Q::\mathbb{O}].(P \vee Q) \Rightarrow (Q \Rightarrow P)::\mathbb{O}], \\
[\forall[P,Q,R::\mathbb{O}].(P \vee Q) \Rightarrow .(R \vee P) \Rightarrow (R \vee Q)::\mathbb{O}], \\
[\forall[F::\mathbb{A} \to \mathbb{O}],[X::\mathbb{A}].\Pi^{\tau(\mathbb{A})}F \Rightarrow FX::\mathbb{O}], \\
[\forall[P::\mathbb{O}],[F::\mathbb{A} \to \mathbb{O}].\forall X::\mathbb{A}(P \vee FX) \Rightarrow .P \vee \Pi^{\mathbb{A}}F::\mathbb{O}],
\end{array}$$

and augment $\Sigma\mathfrak{T}$ with the inference rule

$$\frac{\Gamma \vdash_{\Sigma} \mathbf{A}::\mathbb{T}}{\Gamma \Vdash_{\Sigma} \mathbf{A}}$$

then we can drop all of the propositional axioms of $\Sigma\mathfrak{T}$. The gain of this measure is greater than apparent at first glance. As we have seen in example 5.1.6, a whole class of tautologies (which can be arbitrarily extended by providing further term declarations) can be shown to be of sort $\mathbb{T}$, and thus is specially treated by the calculus. Thus the term declaration mechanism offers a tool to adapt certain calculi to specific needs.

## 5.4 Unifying Principles

In this subsection we introduce an important tool for proving completeness results in higher-order logic. The importance of unifying principles lies in the fact that they abstract over the model theoretic part of various completeness proofs. Unifying principles were first introduced by Smullyan in [Smu63, Smu68] based on work by Hintikka and Beth and later generalized to higher-order logic by Andrews in [And71]. Unifying principles generalize the process of extending a given $\mathcal{C}$-consistent set $\Phi$ of sentences and constructing from it a $\Sigma$-model structure for $\Phi$ by capturing the conditions necessary for this extension in the notion of an abstract consistency class. Thus with the help of a unifying principle the completeness proof for a given logical system $\mathcal{C}$ is reduced to the (purely proof-theoretic) demonstration that the class of $\mathcal{C}$-consistent sets is an abstract consistency class. Since there is no simple Herbrand theorem in higher-order logic, Andrews unifying principle for type theory from [And71] has become the standard method for completeness proofs in higher-order logic.

The most important tools used in the proofs of the unifying principles are the so-called $\Sigma$-Hintikka sets. These sets are maximal elements in abstract consistency classes, and allow computations that resemble those in $\Sigma$-model structures. The key step in the proof of the unifying principles is an abstract extensional lemma, which guarantees a $\Sigma$-Hintikka set $\mathcal{H}$ for any set $H$ of sentences in $\nabla_\Sigma$.

**Definition 5.4.1** Let $\nabla_\Sigma$ be a class of sets.

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

2. $\nabla_\Sigma$ is called **of finite character**, iff for every set $S$ the following condition holds: $S \in \nabla_\Sigma$, iff every finite subset of $S$ is a member of $\nabla_\Sigma$.

**Lemma 5.4.2** *If $\nabla_\Sigma$ is of finite character, then $\nabla_\Sigma$ is closed under subsets.*

**Proof:** Suppose $S \subseteq T$ and $T \in \nabla_\Sigma$. Every finite subset $A$ of $S$ is a finite subset of $T$, and since $\nabla_\Sigma$ is of finite character, we know that $A \in \nabla_\Sigma$. Thus $S \in \nabla_\Sigma$. □

**Definition 5.4.3 (Abstract Consistency Class)** Let $\Gamma$ be a negative annotated variable context, and let $\nabla_\Sigma(\Gamma)$ be a class of sets of propositions, then $\nabla_\Sigma := \{\nabla_\Sigma(\Gamma)\}$ is called an **abstract consistency class**, iff each $\nabla_\Sigma(\Gamma)$ is closed under subsets, and for all sets $\Phi \in \nabla_\Sigma(\Gamma)$ the following conditions hold:

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

2. If $\mathbf{A} \in \Phi$ and if$\mathbf{B}$ is the long $\beta\eta$-normal form of $\mathbf{A}$, then $\mathbf{B} * \Phi \in \nabla_\Sigma(\Gamma)$.

3. If $\neg\neg\mathbf{A} \in \Phi$, then $\mathbf{A} * \Phi \in \nabla_\Sigma(\Gamma)$.

4. If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \nabla_\Sigma(\Gamma)$ or $\Phi * \mathbf{B} \in \nabla_\Sigma(\Gamma)$.

5. If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \nabla_\Sigma(\Gamma)$.

6. If $\Pi^{\mathbb{A}}\mathbf{A} \in \Phi$, then $\Phi * \mathbf{AB} \in \nabla_\Sigma(\Gamma)$ for each $\mathbf{B} \in wsf_{\mathbb{A}}(\Sigma, \Gamma)$.

7. If $\neg\Pi^{\mathbb{A}}\mathbf{A} \in \Phi$, then $\Phi * \neg(\mathbf{A}X) \in \nabla_\Sigma(\Gamma, [X^-{::}\mathbb{A}])$.

We call an abstract consistency class **saturated**, iff for all $\Phi \in \nabla_\Sigma(\Gamma)$ and all atomic propositions $\mathbf{A} \in wsf_{\mathbb{O}}(\Sigma, \Gamma)$ we have $\Phi * \mathbf{A} \in \nabla_\Sigma(\Gamma)$ or $\Phi * \neg\mathbf{A} \in \nabla_\Sigma(\Gamma)$.

**Remark 5.4.4** Note that if $\Sigma$ is trivially sorted, then this definition corresponds to that of Andrews in [And72]. In contrast to the presentation there, we work with saturated abstract consistency classes in order to obtain total $\Sigma$-valuations, which make the proof of the unifying principle much simpler and moreover yield much more natural $\Sigma$-model structures.

**Lemma 5.4.5** *Let $\nabla_\Sigma$ be a saturated abstract consistency class, $\Phi \in \nabla_\Sigma(\Gamma)$, and $\mathbf{A}$ an atomic sentence, then $\Phi * (\mathbf{A} \vee \neg\mathbf{A}) \in \nabla_\Sigma(\Gamma)$.*

**Proof:** Since $\nabla_\Sigma$ is saturated and $\Phi \in \nabla_\Sigma(\Gamma)$, we must have $\Phi * (\mathbf{A} \vee \neg\mathbf{A}) \in \nabla_\Sigma(\Gamma)$ or $\Phi * \neg(\mathbf{A} \vee \neg\mathbf{A}) \in \nabla_\Sigma(\Gamma)$. We prove the assertion by refuting the second alternative. If $\Phi * \neg(\mathbf{A} \vee \neg\mathbf{A}) \in \nabla_\Sigma(\Gamma)$, then $\Phi \cup \{\neg(\mathbf{A} \vee \neg\mathbf{A}), \neg\mathbf{A}, \neg\neg\mathbf{A}, \mathbf{A}\} \in \nabla_\Sigma(\Gamma)$ by 5.4.3.5 and 5.4.3.3. Since $\mathbf{A}$ is an atomic sentence this contradicts 5.4.3.1. □

**Theorem 5.4.6** *For each abstract consistency class $\nabla_\Sigma$ there exists an abstract consistency class $\nabla'_\Sigma$ such that $\nabla_\Sigma(\Gamma) \subseteq \nabla'_\Sigma(\Gamma)$, and $\nabla'_\Sigma$ is of finite character. Furthermore, $\nabla_\Sigma$ is saturated, iff $\nabla'_\Sigma$ is.*

**Proof:** (following [And86]) Let

$$\nabla'_\Sigma(\Gamma) := \{\Phi \subseteq \mathit{wsf}_{\mathbb{O}}(\Sigma, \Gamma) \mid \text{every finite subset of } \Phi \text{ is in } \nabla_\Sigma(\Gamma)\}\,.$$

To see that $\nabla_\Sigma(\Gamma) \subseteq \nabla'_\Sigma(\Gamma)$, suppose that $\Phi \in \nabla_\Sigma(\Gamma)$. $\nabla_\Sigma(\Gamma)$ is closed under subsets, so every finite subset of $\Phi$ is in $\nabla_\Sigma(\Gamma)$, and thus $\Phi \in \nabla'_\Sigma(\Gamma)$.

Next let us show that $\nabla'_\Sigma(\Gamma)$ is of finite character. Suppose $\Phi \in \nabla'_\Sigma(\Gamma)$ and $\Psi$ is an arbitrary finite subset of $\Phi$. By definition of $\nabla'_\Sigma(\Gamma)$ all finite subsets of $\Psi$ are in $\nabla_\Sigma(\Gamma)$, and therefore $\Psi \in \nabla'_\Sigma(\Gamma)$. Thus all finite subsets of $\Phi$ are in $\nabla'_\Sigma(\Gamma)$ whenever $\Psi$ is in $\nabla'_\Sigma(\Gamma)$. On the other hand, suppose all finite subsets of $\Psi$ are in $\nabla'_\Sigma(\Gamma)$. Then by the definition of $\nabla'_\Sigma(\Gamma)$ the finite subsets of $\Psi$ are also in $\nabla_\Sigma(\Gamma)$, so $\Phi \in \nabla'_\Sigma(\Gamma)$. Thus $\nabla'_\Sigma(\Gamma)$ is of finite character.

Now we show that $\nabla'_\Sigma(\Gamma)$ is an abstract consistency class, and $\Phi \in \nabla'_\Sigma(\Gamma)$. By lemma 5.4.2 it is closed under subsets.

1. Suppose there is an atom $\mathbf{A} \in \Phi$ such that $\neg\mathbf{A} \in \Phi$. Then $\{\mathbf{A}, \neg\mathbf{A}\} \in \nabla_\Sigma(\Gamma)$ contradicting 5.4.3(1).
2. Let $(\neg\neg\mathbf{A}) \in \Phi$, and $\Psi$ be any finite subset of $\Phi * \mathbf{A}$ and $\Theta := (\Psi \setminus \{\mathbf{A}\}) * (\neg\neg\mathbf{A})$. $\Theta$ is a finite subset of $\Phi$, so $\Theta \in \nabla_\Sigma(\Gamma)$. Since $\nabla_\Sigma(\Gamma)$ is an abstract consistency class and $(\neg\neg\mathbf{A}) \in \Theta$, we get $\Theta * \mathbf{A} \in \nabla_\Sigma(\Gamma)$. We know that $\Psi \subseteq \Theta * \mathbf{A}$, and $\nabla_\Sigma(\Gamma)$ is closed under subsets, so $\Psi \in \nabla_\Sigma(\Gamma)$. Thus every finite subset $\Psi$ of $\Phi * \mathbf{A}$ is in $\nabla_\Sigma(\Gamma)$, therefore by definition $\Phi * \mathbf{A} \in \nabla'_\Sigma(\Gamma)$.

3.-7. are treated analogously to 2. (see [And86] for a complete presentation).

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

**Definition 5.4.7 (Extensional Abstract Consistency Class)** An abstract consistency class $\nabla_\Sigma$ is called an **extensional abstract consistency class**, iff the following additional conditions hold for all sets $\Phi \in \nabla_\Sigma(\Gamma)$:

8. If $\neg(\mathbf{A} =^{\mathbb{A}} \mathbf{B}) \in \Phi$ and $\mathbb{A} \in \mathcal{S}^f$, then $\Phi * (\neg\mathbf{A}X = \mathbf{B}X) \in \nabla_\Sigma(\Gamma, [X^-::\mathfrak{d}(\mathbb{A})])$.

9. If $\{\mathbf{A}, \mathbf{B}\} \subseteq \Phi$, then $\Phi * (\mathbf{A} = \mathbf{B}) \in \nabla_\Sigma(\Gamma)$.

10. If $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \Phi$, then $\Phi * (\mathbf{A} = \mathbf{B}) \in \nabla_\Sigma(\Gamma)$.

**Theorem 5.4.8** *For each extensional abstract consistency class $\nabla_\Sigma$ there exists an extensional abstract consistency class $\nabla'_\Sigma$ such that $\nabla_\Sigma \subseteq \nabla'_\Sigma$ and $\nabla'_\Sigma$ is of finite character. Just as in 5.4.6 the abstract consistency class $\nabla'_\Sigma$ is saturated, if $\nabla_\Sigma$ is.*

**Proof:** Let $\nabla'_\Sigma$ be the abstract consistency class of 5.4.6, that is

$$\nabla'_\Sigma(\Gamma) := \{\Phi \subseteq wsf_{\mathbb{O}}(\Sigma, \Gamma) \mid \text{every finite subset of } \Phi \text{ is in } \nabla_\Sigma(\Gamma)\} .$$

Then $\nabla_\Sigma(\Gamma) \subseteq \nabla'_\Sigma(\Gamma)$ and $\nabla'_\Sigma$ is a saturated abstract consistency class. To convince ourselves that the additional conditions for extensional case hold, we redo the proof for 5.4.7(8) as a model for the rest.

Let $\mathbf{A} =^{\mathbb{A}} \mathbf{B} \in \Phi$ and $\Psi$ be any finite subset of $\Phi * (\mathbf{A}X = \mathbf{B}X)$, we show that $\Psi \in \nabla'_\Sigma(\Gamma, [X^-::\mathfrak{d}(\mathbb{A})])$. Clearly $\Theta := (\Psi \setminus \{\mathbf{A} = \mathbf{B}\}) * (\mathbf{A}X = \mathbf{B}X)$ is a finite subset of $\Phi$, and therefore $\Theta \in \nabla_\Sigma(\Gamma, [X^-::\mathfrak{d}(\mathbb{A})])$. Since $\nabla_\Sigma(\Gamma)$ is an abstract consistency class and $(\mathbf{A} =^{\mathbb{A}} \mathbf{B}) \in \Theta$, we have $\Theta * (\mathbf{A}X = \mathbf{B}X) \in \nabla_\Sigma(\Gamma, [X^-::\mathfrak{d}(\mathbb{A})])$. Furthermore, $\Psi \subseteq \Theta * (\mathbf{A} =^{\mathbb{A}} \mathbf{B})$ and $\nabla_\Sigma(\Gamma)$ is closed under subsets, so $\Psi \in \nabla_\Sigma(\Gamma)$. Thus every finite subset $\Psi$ of $\Phi * (\mathbf{A}X = \mathbf{B}X)$ is in $\nabla_\Sigma(\Gamma, [X^-::\mathfrak{d}(\mathbb{A})])$, therefore by definition we have $\Phi * (\mathbf{A} = \mathbf{B}) \in \nabla'_\Sigma(\Gamma)$. □

**Definition 5.4.9 (Σ-Hintikka Set)** Let $\nabla_\Sigma$ be an abstract consistency class and $H \in \nabla_\Sigma(\Gamma)$. Then $\mathcal{H} \in \nabla_\Sigma(\Gamma')$ is called a **$\nabla_\Sigma$-extension of** $H$, iff $H \subseteq \mathcal{H}$ and $\Gamma \subseteq \Gamma'$. A set $\mathcal{H}$ is called **maximal in** $\nabla_\Sigma(\Gamma)$, iff for each sentence $\mathbf{D} \in \nabla_\Sigma(\Gamma)$ such that $\mathcal{H} * \mathbf{D} \in \nabla_\Sigma(\Gamma)$, we already have $\mathbf{D} \in \mathcal{H}$. A set $\mathcal{H} \in \nabla_\Sigma(\Gamma)$ is called a **Σ-Hintikka set for $\nabla_\Sigma$ and** $H$, iff $\mathcal{H}$ is maximal in $\nabla_\Sigma(\Gamma)$ and $H \subseteq \mathcal{H}$.

We now give some technical properties of Σ-Hintikka sets that are useful for manipulating formulae. Since the case of extensional abstract consistency classes concerns a superset of conditions, we always treat the non-extensional case first, and then extend the result for the extensional case.

**Theorem 5.4.10 (Hintikka Lemma)** *If $\nabla_\Sigma$ is an abstract consistency class, and $\mathcal{H}$ is maximal in $\nabla_\Sigma$, then the following statements hold:*

1. *If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$.*

2. *If $(\neg\neg\mathbf{A}) \in \mathcal{H}$, then $\mathbf{A} \in \mathcal{H}$.*

3. *If $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$, then we have $\mathbf{A} \in \mathcal{H}$, iff $\mathbf{B} \in \mathcal{H}$.*

4. *If $(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$, then $\mathbf{A} \in \mathcal{H}$ or $\mathbf{B} \in \mathcal{H}$.*

5. *If $\neg(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$, then $\neg\mathbf{A} \in \mathcal{H}$ and $\neg\mathbf{B} \in \mathcal{H}$.*

6. *If $\Pi^{\mathbb{A}}\mathbf{A} \in \mathcal{H}$, then for each $\mathbf{B} \in wsf_{\mathbb{A}}(\Sigma, \Gamma)$ we have $\mathbf{AB} \in \mathcal{H}$.*

7. *If* $\neg\Pi^{\mathbb{A}}\mathbf{A} \in \mathcal{H}$*, then there is a* $\mathbf{B} \in wsf_{\mathbb{A}}(\Sigma,\Gamma)$ *such that* $\neg\mathbf{AB} \in \mathcal{H}$*.*

*Furthermore for any atomic sentence* $\mathbf{A}$ *we have* $(\mathbf{A} \vee \neg\mathbf{A}) \in \mathcal{H}$*.*

**Proof:** The assertions are all of the same form, and have analogous proofs, therefore we only prove the first assertion. If $\neg\neg\mathbf{A} \in \mathcal{H}$, then $\mathcal{H} * \mathbf{A} \in \nabla_\Sigma(\Gamma)$ ($\nabla_\Sigma(\Gamma)$ is an abstract consistency class). The maximality of $\mathcal{H}$ now gives the assertion. The last claim of the theorem can be proven with the same methods using 5.4.5. □

**Theorem 5.4.11** *If* $\nabla_\Sigma$ *is an extensional abstract consistency class, and* $\mathcal{H}$ *is a* $\Sigma$*-Hintikka set for* $\nabla_\Sigma$*, then the following statements hold:*

8. *Let* $\mathbf{A},\mathbf{B} \in wsf_{\mathbb{A}}(\Sigma,\Gamma)$*, then there is a* $\mathbf{C} \in wsf_{\mathfrak{d}(\mathbb{A})}(\Sigma,\Gamma)$*, such that* $(\neg\mathbf{AC} = \mathbf{BC}) \in \mathcal{H}$*, if* $\neg(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$*.*

9. *Let* $\mathbf{A},\mathbf{B} \in wsf_{\mathbb{O}}(\Sigma,\Gamma)$*, then* $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$*, if* $\{\mathbf{A},\mathbf{B}\} \subseteq \mathcal{H}$*.*

10. *Let* $\mathbf{A},\mathbf{B} \in wsf_{\mathbb{O}}(\Sigma,\Gamma)$*, then* $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$*, if* $\{\neg\mathbf{A},\neg\mathbf{B}\} \subseteq \mathcal{H}$*.*

**Proof:** The proofs are analogous to those of 5.4.10. □

**Lemma 5.4.12** *Let* $\nabla_\Sigma$ *be a saturated abstract consistency class, let* $\mathcal{H}$ *be maximal in* $\nabla_\Sigma(\Gamma)$*, and* $\mathbf{A} \in wsf_{\mathbb{O}}(\Sigma,\Gamma)$*, then* $\mathbf{A} \in \mathcal{H}$*, iff* $\neg\mathbf{A} \notin \mathcal{H}$*.*

**Proof:** We prove the assertion by induction on the structure of $\mathbf{A}$. If $\mathbf{A}$ is atomic, then $\mathcal{H}*\mathbf{A} \in \nabla_\Sigma(\Gamma)$ or $\mathcal{H}*\neg\mathbf{A} \in \nabla_\Sigma(\Gamma)$, since $\nabla_\Sigma$ is saturated and $\mathcal{H} \in \nabla_\Sigma(\Gamma)$. The maximality of $\mathcal{H}$ tells us that $\mathbf{A} \in \mathcal{H}$ or $\neg\mathbf{A} \in \mathcal{H}$. Now the assertion is a simple consequence of 5.4.3(1).

If $\mathbf{A} \doteq \neg\mathbf{B}$, then $\neg\mathbf{A} \doteq (\neg\neg\mathbf{B}) \in \mathcal{H}$, and therefore $\mathbf{B} \in \mathcal{H}$ by 5.4.3(3), contradicting the induction hypothesis. If $\mathbf{A} \doteq \mathbf{B} \vee \mathbf{C}$, then $\mathbf{B} \in \mathcal{H}$ or $\mathbf{C} \in \mathcal{H}$ by 5.4.3(4). On the other hand $\neg\mathbf{A} \doteq \neg(\mathbf{B} \vee \mathbf{C})$, and by 5.4.3(5) we have $\{\neg\mathbf{B},\neg\mathbf{C}\} \subseteq \mathcal{H}$, contradicting the inductive hypothesis. The rest of the cases can be shown analogously. □

**Corollary 5.4.13** *If we assume that the abstract consistency class in 5.4.10 and 5.4.11 is saturated, then the statements there also hold in the other direction. For instance,* $\neg\Pi^{\mathbb{A}}\mathbf{A} \in \mathcal{H}$*, iff there is a* $\mathbf{B} \in wsf_{\mathbb{A}}(\Sigma,\Gamma)$ *such that* $\neg\mathbf{AB} \in \mathcal{H}$*. Furthermore, if* $\mathbf{A},\mathbf{B} \in wsf_{\mathbb{O}}(\Sigma,\Gamma)$*, then* $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$*, iff* $\{\mathbf{A},\mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\neg\mathbf{A},\neg\mathbf{B}\} \subseteq \mathcal{H}$*.*

**Proof:** Since all proofs are analogous we only show the case of existential quantification. From 5.4.10 we know that $\neg\neg\mathbf{A} \in \mathcal{H}$ implies that $\mathbf{A} \in \mathcal{H}$. So suppose that $\mathbf{A} \in \mathcal{H}$, then by 5.4.12 we know that $\neg\mathbf{A} \notin \mathcal{H}$ and again by 5.4.12 $\neg\neg\mathbf{A} \in \mathcal{H}$. □

**Lemma 5.4.14** *Let* $\nabla_\Sigma$ *be an extensional abstract consistency class and* $\mathcal{H}$ *a* $\Sigma$*-Hintikka set for* $\nabla_\Sigma$*. If* $\mathbf{A},\mathbf{B} \in wsf_{\mathbb{A}}(\Sigma,\Gamma)$*, then* $\forall X_{\mathfrak{d}(\mathbb{A})}(\mathbf{A}X = \mathbf{B}X) \in \mathcal{H}$*, implies* $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$*.*

**Proof:** We have $(\mathbf{A} = \mathbf{B}) \notin \mathcal{H}$, iff $\neg(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$ by 5.4.12 (and possibly 5.4.10(2)). So by 5.4.11(9) there is a $\mathbf{C}_\beta \in wsf_{\mathbb{B}}(\Sigma,\Gamma)$ such that $\neg(\mathbf{AC} = \mathbf{BC}) \in \mathcal{H}$. If we assume that $\forall X_\beta(\mathbf{A}X = \mathbf{B}X) \in \mathcal{H}$, then we obtain $(\mathbf{AC} = \mathbf{BC}) \in \mathcal{H}$ by 5.4.10(6), which contradicts 5.4.12. □

**Lemma 5.4.15** *Let $\nabla_\Sigma$ be an extensional abstract consistency class and $H \in \nabla_\Sigma(\Gamma)$. If $\mathcal{H}$ is a $\Sigma$-Hintikka set for $\nabla_\Sigma$ and $H$, then $(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$ implies $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$.*

**Proof:** Let $(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$, then by 5.4.10(5) we also have $\neg\mathbf{A} \vee \mathbf{B} \in \mathcal{H}$, and moreover, $\mathbf{A} \vee \neg\mathbf{B} \in \mathcal{H}$. Because of 5.4.10(4) we have to consider two cases: if $\mathbf{B} \in \mathcal{H}$, then $\neg\mathbf{B} \notin \mathcal{H}$, and therefore $\mathbf{A} \in \mathcal{H}$. If $\neg\mathbf{A} \in \mathcal{H}$, then $\mathbf{A} \notin \mathcal{H}$, and therefore $\neg\mathbf{B} \in \mathcal{H}$. In both cases we get the assertion $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$ by 5.4.11(10) or 5.4.11(11). $\Box$

**Lemma 5.4.16** *If $\mathcal{H}$ is a $\Sigma$-Hintikka set and $\mathbf{A}, \mathbf{B}$ are propositions, then either $\mathbf{A} = \mathbf{B} \in \mathcal{H}$ or $\mathbf{A} = \neg\mathbf{B} \in \mathcal{H}$.*

**Proof:** A tedious, but straightforward computation using the results from lemma 5.4.10 shows that $\neg(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$, iff $\mathbf{A} \Leftrightarrow \neg\mathbf{B} \in \mathcal{H}$. Now we conclude with 5.4.12, that either $\mathbf{A} \Leftrightarrow \mathbf{B} \in \mathcal{H}$ or $(\mathbf{A} \Leftrightarrow \neg\mathbf{B}) \in \mathcal{H}$, from which we get the assertion by 5.4.15. $\Box$

We now come to the proof of the abstract extension lemma, which nearly immediately yield the unifying principle for $\Sigma$-model structures. For the proof we adapt the construction of Henkin's completeness proof for $\Sigma\mathfrak{T}\mathfrak{E}$ from [Hen50].

**Theorem 5.4.17 (Abstract Extension Lemma)** *Let $\nabla_\Sigma$ be an (extensional) abstract consistency class of finite character, and let $H \in \nabla_\Sigma(\Gamma)$ be a set of propositions. Then there exists a $\Sigma$-Hintikka set $\mathcal{H}$ for $\nabla_\Sigma$ and $H$.*

**Proof:** For the case where $\nabla_\Sigma$ is extensional we construct $\mathcal{H}$ by inductively constructing a sequence of sets $\mathcal{H}^i \in \nabla_\Sigma(\Gamma)$ and negative annotated variable contexts $\Gamma^i$ such that $\mathcal{H}^i \in \nabla_\Sigma(\Gamma^i)$ and $\Gamma^i || \Gamma^{i+1}$. Then the $\Sigma$-Hintikka set is $\mathcal{H} := \bigcup_{i \in \mathbb{N}} \mathcal{H}^i \in \nabla_\Sigma(\Gamma)$, where $\Gamma := \bigcup_{i \in \mathbb{N}} \Gamma^i$. For the base case we choose $\mathcal{H}^0 := H$ and $\Gamma^0 := \Gamma$.

Now let $\mathcal{H}^i$ and $\Gamma^i$ be already defined, then we can arrange all propositions in $wsf_{\mathbb{O}}(\Sigma, \Gamma)$ as two infinite sequences $\mathbf{C}^1, \mathbf{C}^2, \ldots$ and $\mathbf{D}^i, \mathbf{D}^2, \ldots$, where the $\mathbf{D}^i$ are of the form $\neg(\Pi^{\mathbb{A}}\mathbf{A})$ or $\neg\mathbf{A} =^{\mathbb{A}} \mathbf{B}$ for some $\mathbb{A} \in \mathcal{S}^f$, and the $\mathbf{C}^i$ are not. For each $n \in \mathbb{N}$ we inductively define a set $H^n \subseteq wsf_{\mathbb{O}}(\Sigma, \Gamma^i)$ of propositions by

1. $H^0 := \mathcal{H}^i$.
2. If $H^n * \mathbf{C}^n \notin \nabla_\Sigma(\Gamma)$, then $H^{n+1} := H^n$.
3. If $H^n * \mathbf{C}^n \in \nabla_\Sigma(\Gamma)$ then $H^{n+1} := H^n * \mathbf{C}^n$.

Let $\mathcal{K}^i := \bigcup_{n \in \mathbb{N}} H^n$, then clearly each of the $H^n \in \nabla_\Sigma(\Gamma^i)$, and therefore $\mathcal{K}^i \in \nabla_\Sigma(\Gamma^i)$, since $\nabla_\Sigma$ is of finite character. Now we inductively define negative annotated variable contexts $\Delta^n$ and sets $K^n \subseteq wsf_{\mathbb{O}}(\Sigma, \Delta^i)$ of propositions by

1. $\Delta^0 := \Gamma^i$ and $K^0 := \mathcal{K}^i$.
2. If $K^n * \mathbf{D}^n \in \nabla_\Sigma(\Gamma)$ and $\mathbf{D}^n$ is of the form $(\neg\Pi^{\mathbb{A}}\mathbf{A})$, then $K^{n+1} := K^n * \neg\Pi^{\mathbb{A}}\mathbf{A} * \neg(\mathbf{A}X)$, where $X \notin \mathbf{Dom}(\Delta^n)$ and $\Delta^{n+1} := \Delta^n, [X^-{::}\mathbb{A}]$.
3. If $K^n * \mathbf{D}^n \in \nabla_\Sigma(\Gamma)$ and $\mathbf{D}^n$ is of the form $(\mathbf{A} \neq^{\mathbb{A}} \mathbf{B})$ with $\mathbb{A} \in \mathcal{S}^f$, then we choose $K^{n+1} := K^n * \mathbf{A} \neq \mathbf{B} * (\mathbf{A}X \neq \mathbf{B}X)$, where $X \notin \mathbf{Dom}(\Delta^n)$ and $\Delta^{n+1} := \Delta^n, [X^-{::}\mathfrak{d}(\mathbb{A})]$.

and set $\mathcal{H}^{i+1} := \bigcup_{n\in\mathbb{N}} K^n$ and $\Gamma^{i+1} := \bigcup_{n\in\mathbb{N}} \Delta^n$. We have to treat the cases 2 and 3 in separate rules, since without 3 we would only obtain the witness $X\mathbf{A} \neq X\mathbf{B}$ for $\mathbf{A} \neq \mathbf{B}$ instead of $\mathbf{A}X \neq \mathbf{B}X$.

Next we show by induction that $K^n \in \nabla_\Sigma(\Gamma^n)$ for all $n \in \mathbb{N}$. The base case holds by construction. So let $K^n * \mathbf{D}^n \in \nabla_\Sigma(\Gamma)$ where $\mathbf{D}^n$ is of the form $\neg\Pi^{\mathbb{A}}\mathbf{A}$. By construction $X^- \notin \mathbf{Dom}(\Delta^n)$, so by 5.4.3(7) we have $K^{n+1} \in \nabla_\Sigma(\Delta^{n+1})$. Since $\nabla_\Sigma$ is of finite character, we also have $\mathcal{H}^{i+1} \in \nabla_\Sigma(\Gamma)$.

In order to prove the maximality of $\mathcal{H}$, let $\mathbf{A} \in wsf_{\mathbb{O}}(\Sigma,\Gamma)$ be an arbitrary proposition such that $\mathcal{H} * \mathbf{D} \in \nabla_\Sigma(\Gamma)$. Since $\mathbf{A}$ has only finitely many free variables, there is an $n \in \mathbb{N}$ with $\Gamma^n \vdash_\Sigma \mathbf{A}{::}\mathbb{O}$. Furthermore, we know that $\mathbf{A} = \mathbf{C}^k$ or $\mathbf{A} = \mathbf{D}^k$ for some $k \in \mathbb{N}$. If $\mathbf{A} = \mathbf{C}^k$, then $H^n * \mathbf{C} \subseteq \mathcal{H} * \mathbf{C} \in \nabla_\Sigma(\Gamma^n)$ and $H^n * \mathbf{C} \in \nabla_\Sigma(\Delta)$, since $\nabla_\Sigma$ is closed under subsets. Hence by definition we know that $\mathbf{A} \in H^{n+1} \subseteq \mathcal{H}^{k+1}$, and therefore $\mathbf{A} \in \mathcal{H}$. The case for $\mathbf{A} = \mathbf{D}^k$ can be treated analogously.

If $\nabla_\Sigma$ is not extensional, then we do not have to treat the case where $\mathbf{D}^k$ is of the form $(\mathbf{A} \neq^{\mathbb{A}} \mathbf{B})$, and the same construction without rule 3 yields the desired $\Sigma$-Hintikka set $\mathcal{H}$. □

We now use the $\Sigma$-Hintikka set, guaranteed by the previous lemma, to construct a $\Sigma$-valuation for the $\Sigma$-term structure that turns it into a $\Sigma$-model structure.

**Corollary 5.4.18 (Unifying Principle for $\Sigma$-Model Structures)** *Let $H \in \nabla_\Sigma(\Gamma)$ and $\nabla_\Sigma$ be a saturated abstract consistency class, then there is a $\Sigma$-model structure $\mathcal{M}$ with $\mathcal{M} \models H$.*

**Proof:** Let $\mathcal{H}$ be the maximal $\nabla_\Sigma$-extension guaranteed by 5.4.17, then we chose $v(\mathbf{C}) = \mathtt{T}$, if $\mathbf{C} \in \mathcal{H}$ and $v(\mathbf{C}) = \mathtt{F}$, if $\neg\mathbf{C} \in \mathcal{H}$. By 5.4.12 $v$ is a total function, and by 5.4.10 $v$ is a $\Sigma$-valuation of the $\Sigma$-term structure $\mathcal{TS}(\Sigma,\Gamma)$. Thus $\mathcal{M} := (\mathcal{TS}(\Sigma,\Gamma), v)$ is a $\Sigma$-model structure with $\mathcal{M} \models H$. □

We now state a variant of the previous theorem, which is related to Andrews' unifying principle for type theory from [And71]. It is not a generalization precise of his theorem, since $\eta$-equivalence, which we need for the functionality of $\Sigma$-model structures, is not considered there. In particular, it seems difficult to extend our methods to obtain his result.

**Theorem 5.4.19 (Unifying Principle for $\Sigma\mathfrak{T}\eta$)** *If $\nabla_\Sigma$ is a saturated abstract consistency class, and $\Phi \in \nabla_\Sigma(\Gamma)$ is a finite set of sentences, then $\Phi$ is $\Sigma\mathfrak{T}\eta$-consistent.*

**Proof:** Let $\mathcal{K}$ be the class of $\Sigma$-model structures and $\mathcal{M} \in \mathcal{K}$ the $\Sigma$-model structure guaranteed by 5.4.18. By theorem 5.3.6 $\Sigma\mathfrak{T}\eta$ is sound with respect to $\mathcal{K}$, and therefore 5.3.13(2) gives the assertion. □

We now turn to the unifying principle for general $\Sigma$-models. In contrast to the case for $\Sigma$-models structures, we have to construct a $\Sigma$-structure with $\mathcal{D}_{\mathbb{O}} = \{\mathtt{T},\mathtt{F}\}$. We do this by extracting a $\Sigma$-congruence $\sim_{\mathcal{H}}$ from the $\Sigma$-Hintikka set $\mathcal{H}$ guaranteed by 5.4.17 and taking the quotient pre-$\Sigma$-structure of the $\Sigma$-term structure with respect to $\sim_{\mathcal{H}}$

**Definition 5.4.20** Let $\nabla_\Sigma$ be an extensional abstract consistency class, and let $\mathcal{H}$ be maximal in $\nabla_\Sigma$. Then formulae $\mathbf{A}$ and $\mathbf{B}$ are called **$\mathcal{H}$-congruent** ($\mathbf{A} \sim_{\mathcal{H}} \mathbf{B}$), iff the universal closure of $\mathbf{A} = \mathbf{B}$ is a member of $\mathcal{H}$.

**Lemma 5.4.21 (Congruence Lemma)** *Let $\nabla_\Sigma$ be an abstract consistency class, and let $\mathcal{H} \neq \emptyset$ be maximal in $\nabla_\Sigma$, then $\sim_\mathcal{H}$ is a functional $\Sigma$-congruence on $wsf(\Sigma, \Gamma)$.*

**Proof:** To obtain the assertion we first have to make sure that $\sim_\mathcal{H}$ is an equivalence relation. We only give the tedious details of the proof of symmetry as an example for proofs in abstract consistency classes, since the syntactic manipulations for transitivity and reflexivity are analogous.

Let $(\mathbf{A} = \mathbf{B}) \doteq (\forall P_{\mathbb{A}\to\mathbb{O}}.P\mathbf{A} \Rightarrow P\mathbf{B}) \in \mathcal{H}$ and $\mathbf{P} \in wsf_{\mathbb{A}\to\mathbb{O}}(\Sigma, \Gamma)$ be an arbitrary formula, then by 5.4.10(6) we have $(\neg\mathbf{PA} \Rightarrow \neg\mathbf{PB}) = ((\neg\neg\mathbf{PA}) \vee .\neg\mathbf{PB}) \in \mathcal{H}$. Now by 5.4.10(4) we have to consider two cases. If $\neg\neg\mathbf{PA} \in \mathcal{H}$, then $\mathbf{PA} \in \mathcal{H}$, and therefore $\mathbf{PA} \vee .\neg\mathbf{PB} \in \mathcal{H}$ by 5.4.10(2) and 5.4.10(4). If on the other hand $\neg\mathbf{PB} \in \mathcal{H}$, then $\neg\mathbf{PB} \vee \mathbf{PA} \in \mathcal{H}$. In both cases we have $(\neg\mathbf{PB} \Rightarrow \mathbf{PA}) \in \mathcal{H}$ for all $\mathbf{P} \in wsf_{\mathbb{A}\to\mathbb{O}}(\Sigma, \Gamma)$, and therefore $(\mathbf{A} = \mathbf{B}) \doteq \forall P_{\mathbb{A}\to\mathbb{O}}.P\mathbf{B} \Rightarrow P\mathbf{A} \in \mathcal{H}$ by 5.4.10(6).

Now we verify the congruence property. Let $\mathbf{A}, \mathbf{B} \in wsf_{\mathfrak{d}(\mathbb{A})}(\Sigma, \Gamma)$, we only prove that $\mathbf{CA} \sim_\mathcal{H} \mathbf{CB}$ for all $\mathbf{C} \in wsf_{\mathbb{A}}(\Sigma, \Gamma)$ whenever $\mathbf{A} \sim_\mathcal{H} \mathbf{B}$, since the other condition is analogous. So let $(\mathbf{A} = \mathbf{B}) \doteq (\forall P_{\mathbb{A}\to\mathbb{O}}.P\mathbf{A} \Rightarrow P\mathbf{B}) \in \mathcal{H}$ and $\mathbf{P} \in wsf_{\mathfrak{d}(\mathbb{A})\to\mathbb{O}}(\Sigma, \Gamma)$ be an arbitrary formula, then $\mathbf{P}(\mathbf{CA}) \Rightarrow \mathbf{P}(\mathbf{CB}) \in \mathcal{H}$, since $\Gamma \vdash_\Sigma \lambda_{\mathfrak{d}(\mathbb{A})}.\mathbf{P}(\mathbf{C}X)::\mathfrak{d}(\mathbb{A}) \to \mathbb{O}$, and therefore $\mathbf{CA} = \mathbf{CB} \in \mathcal{H}$, since $\mathbf{P}$ was arbitrary.

To see that $\sim_\mathcal{H}$ is functional let $\mathbf{A}, \mathbf{B} \in wsf_{\mathbb{A}}(\Sigma, \Gamma)$ and $\mathbf{AC} \sim_\mathcal{H} \mathbf{BC}$ for all $\mathbf{C} \in wsf_{\mathfrak{d}(\mathbb{A})}(\Sigma, \Gamma)$, then we have $\mathbf{AC} =^{\mathfrak{r}(\mathbb{A})} \mathbf{BC} \in \mathcal{H}$ for all $\mathbf{C} \in wsf_{\mathfrak{d}(\mathbb{A})}(\Sigma, \Gamma)$. By 5.4.13 we know that $\mathbf{A} \neq^{\mathbb{A}} \mathbf{B} \in \mathcal{H}$, iff there is a formula $\mathbf{D} \in wsf_{\mathfrak{d}(\mathbb{A})}(\Sigma, \Gamma)$ such that $\mathbf{AD} =^{\mathfrak{r}(\mathbb{A})} \mathbf{BD} \in \mathcal{H}$, thus $\mathbf{A} \neq^{\mathbb{A}} \mathbf{B} \notin \mathcal{H}$. By 5.4.12 this entails $\mathbf{A} =^{\mathbb{A}} \mathbf{B} \in \mathcal{H}$, and thus $\mathbf{A} \sim_\mathcal{H} \mathbf{B}$. □

**Remark 5.4.22** Note that in the proof of the congruence lemma we have implicitly used lemma 5.4.15, since we have only considered the congruence properties of $\sim_\mathcal{H}$ as given by the presence of some equalities in $\mathcal{H}$. Since we treat equality as an abbreviation of Leibniz' indiscernability formula, the congruence properties follow almost immediately from the use of logical constants and the definition of the abstract consistency class. Thus, with the help of 5.4.15, we do not have to consider the congruence properties of equivalence and the interaction of equivalence and equality.

**Theorem 5.4.23 (Unifying Principle for General Σ-Models)** *If $\nabla_\Sigma$ is a saturated, extensional abstract consistency class, then $H$ has a countable general $\Sigma$-model.*

**Proof:** We can assume without loss of generality (5.4.8) that $\nabla_\Sigma$ is of finite character, so the preconditions of 5.4.17 are met, and therefore there exists a $\Sigma$-Hintikka set $\mathcal{H} \subseteq wsf_{\mathbb{O}}(\Sigma, \Gamma)$ for $\nabla_\Sigma$ with $H \subseteq \mathcal{H}$. By 5.4.21 the relation $\sim_\mathcal{H}$ is a functional $\Sigma$-congruence, so the quotient structure $\mathcal{M}^\mathcal{H} = \mathcal{TS}(\Sigma, \Gamma)/_{\sim_\mathcal{H}} = (\mathcal{D}^\mathcal{H}, @, \mathcal{I}^\mathcal{H})$ with respect to the $\mathcal{H}$-congruence is a $\Sigma$-structure by lemma 3.3.10. From lemma 5.4.16 we know that $\sim_\mathcal{H}$ has exactly two equivalence classes on $\mathcal{TS}_{\mathbb{O}}(\Sigma, \Gamma)$. Thus we have $\mathcal{D}_{\mathbb{O}} = \{\mathtt{T}, \mathtt{F}\}$, if we define $\mathtt{T} := [\![\mathbf{A} \vee \neg\mathbf{A}]\!]$ and $\mathtt{F} := [\![\mathbf{A} \wedge \neg\mathbf{A}]\!]$ for some atomic sentence. By lemma 3.5.13 we have $\mathcal{I}_\varphi(\mathbf{A}) = [\![\varphi(\mathbf{A})]\!] = \pi_\mathcal{H}(\varphi(\mathbf{A}))$.

By lemmata 5.2.11 and 5.2.7 it suffices to show, that $\mathcal{I}(\mathbf{Q}^{\mathbb{A}})$ is the identity relation on $\mathcal{D}_{\mathbb{A}}$. Since $\pi_\mathcal{H}$ is an epimorphism $\mathcal{I}_\varphi(\mathbf{A}) = [\![\varphi(\mathbf{A})]\!]$ and $\mathcal{I}_\varphi(\mathbf{B}) = [\![\varphi(\mathbf{B})]\!]$ be two arbitrary members of $\mathcal{D}_{\mathbb{A}}$. By construction $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$, iff $(\mathbf{A} = \mathbf{B}) \in \mathcal{H}$, iff $\mathtt{T} = \mathcal{I}(\mathbf{Q}^{\mathbb{A}}\mathbf{AB}) = \mathcal{I}(\mathbf{Q}^{\mathbb{A}})@\mathcal{I}(\mathbf{A})@\mathcal{I}(\mathbf{B})$, thus $\mathcal{I}(\mathbf{Q}^{\mathbb{A}})$ is indeed the identity relation on $\mathcal{D}_{\mathbb{A}}$, and $\mathcal{M}^\mathcal{H}$ is a general $\Sigma$-model.

We have $\mathcal{I}_\varphi(\mathcal{H}) = \{\mathtt{T}\}$ for each assignment $\varphi$ into $\mathcal{D}$, since $\mathbf{A} \vee \neg\mathbf{A} \in \mathcal{H}$. Furthermore, we have $H \subseteq \mathcal{H}$, hence we get $\mathcal{I}_\varphi(H) = \{\mathtt{T}\}$, and therefore $\mathcal{M} \models H$.

If we pay attention to the constructions in the proof of 5.4.17, it is easy to see that $\mathcal{M}^{\mathcal{H}}$ is indeed countable, since the sets of well-sorted formulae are countable. $\square$

## 5.5 Completeness

In this subsection we use the unifying principles for $\Sigma\mathcal{HOL}$ to give short and elegant proofs of completeness for $\Sigma\mathfrak{T}\eta$ and $\Sigma\mathfrak{TE}$.

**Theorem 5.5.1** *The class $\nabla_\Sigma := \{\Phi \subseteq wsf_{\mathbb{O}}(\Sigma,\Gamma) \mid \Phi$ is $\Sigma\mathfrak{TE}$-consistent$\}$ is an extensional abstract consistency class.*

**Proof:** Obviously $\nabla_\Sigma$ is closed under subsets, since any subset of a $\Sigma\mathfrak{TE}$-consistent set is $\Sigma\mathfrak{TE}$-consistent. Also by definition no well-formed formula $\mathbf{A}$ can be in a $\Sigma\mathfrak{TE}$-consistent set along with its negation $\neg\mathbf{A}$, this establishes 5.4.3(1). $\nabla_\Sigma$ is saturated by 5.3.11.

To verify 5.4.3(3), 5.4.3(5), and 5.4.3(6) we note that, if $\Vdash_{\Sigma\mathfrak{TE}} \mathbf{C} \Rightarrow \mathbf{D}^1 \wedge \ldots \wedge \mathbf{D}^n$ for some $\mathbf{C} \in \Phi$ where $\Phi$ is $\Sigma\mathfrak{TE}$-consistent, then $\Phi \cup \{\mathbf{D}^1,\ldots,\mathbf{D}^n\}$ must be $\Sigma\mathfrak{TE}$-consistent (5.3.13(3)). The observation that the proposition $((\neg\mathbf{A}\wedge\neg\mathbf{B})\vee(\mathbf{A}\wedge\mathbf{B})) \Leftrightarrow .\mathbf{A} \Leftrightarrow \mathbf{B}$ is tautologous can be used to extend this argument to a proof of 5.4.7(9) and 5.4.7(10).

If $\Phi$ is $\Sigma\mathfrak{TE}$-consistent, and $\Phi{*}\mathbf{A}$ and $\Phi{*}\mathbf{B}$ are both $\Sigma\mathfrak{TE}$-inconsistent, then $\Phi \Vdash_{\Sigma\mathfrak{TE}} \neg\mathbf{A}$ and $\Phi \Vdash_{\Sigma\mathfrak{TE}} \neg\mathbf{B}$, so $\Phi \Vdash_{\Sigma\mathfrak{TE}} \neg(\mathbf{A} \vee \mathbf{B})$ by rule $P$ (cf. 5.3.8), therefore $(\mathbf{A} \vee \mathbf{B}) \notin \Phi$, which is just the contrapositive of 5.4.3(4).

To establish the remaining cases 5.4.3(7) and 5.4.7(8), where the variable context is extended with a new variable $X^-$, let $\Phi \subseteq wsf_{\mathbb{O}}(\Sigma,\Gamma)$ and $X \notin \mathbf{Dom}(\Gamma)$. We only show the first case, since the other is analogous.

We assume that $\neg\Pi^{\mathbb{A}}\mathbf{A} \in \Phi$ and $\Phi$ is $\Sigma\mathfrak{TE}$-consistent, but $\Phi{*}\neg(\mathbf{A}X)$ is $\Sigma\mathfrak{TE}$-inconsistent. So there is a $\Sigma\mathfrak{TE}$-derivation $\mathcal{D}\colon \Phi \Vdash_{\Sigma\mathfrak{TE}} \mathbf{A}X$ by 5.3.13 and 5.3.8. By adding an application of $\Sigma\mathfrak{T}(UG)$ the root of $\mathcal{D}'$ we obtain a $\Sigma\mathfrak{TE}$-derivation of $\Phi \Vdash_{\Sigma\mathfrak{T}} \Pi\mathbf{A}$, which contradicts our assumption that $\Phi$ is $\Sigma\mathfrak{T}$-consistent. Thus $\neg(\mathbf{A}X) \notin \Phi$. $\square$

**Corollary 5.5.2 (Henkin's Theorem for $\Sigma\mathfrak{TE}$)** *Every $\Sigma\mathfrak{TE}$-consistent set of sentences has a countable general $\Sigma$-model.*

**Proof:** By 5.5.1 we know that the class of sets of $\Sigma\mathfrak{TE}$-consistent propositions constitute a saturated, extensional abstract consistency class $\nabla_\Sigma$ with $\Phi \in \nabla_\Sigma(\Gamma)$. Thus 5.4.23 guarantees a countable general $\Sigma$-model for $\Phi$. $\square$

**Corollary 5.5.3 (Completeness Theorem for $\Sigma\mathfrak{TE}$)** *We have $\mathbf{A} \Vdash_{\Sigma\mathfrak{TE}} \mathbf{B}$, iff $\mathbf{A} \models \mathbf{B}$ with respect to the class of general $\Sigma$-models.*

**Remark 5.5.4** In the light of the previous theorem it is not surprising that we can prove the formula that was used to show incompleteness 5.3.14 of $\Sigma\mathfrak{T}\eta$ in $\Sigma\mathfrak{TE}$. Here we sketch the direct proof. We have $\Vdash_{\Sigma\mathfrak{TE}} b \Leftrightarrow .\neg\neg b$ and by extensionality $\Vdash_{\Sigma\mathfrak{TE}} b = .\neg\neg b$, which expands to $\Vdash_{\Sigma\mathfrak{TE}} \forall P_{\mathbb{O}\to\mathbb{O}}.Pb \Rightarrow P.\neg\neg b$ and by substitution $\Vdash_{\Sigma\mathfrak{TE}} cb \Rightarrow c.\neg\neg b$. $\square$

With the same methods we can prove the following theorems.

**Theorem 5.5.5** *The class $\nabla_\Sigma$ with $\nabla_\Sigma := \{\Phi \subseteq wsf_{\mathbb{O}}(\Sigma,\Gamma) \mid \Phi$ is $\Sigma\mathfrak{T}\eta$-consistent$\}$ is a saturated abstract consistency class.*

**Theorem 5.5.6 (Henkin's Theorem for $\Sigma\mathfrak{T}$)** *Every $\Sigma\mathfrak{T}\eta$-consistent set of sentences has a countable $\Sigma$-model structure.*

**Theorem 5.5.7 (Completeness Theorem for $\Sigma\mathfrak{T}\eta$)** *We have $\mathbf{A} \Vdash_{\Sigma\mathfrak{T}\eta} \mathbf{B}$, iff $\mathbf{A} \models \mathbf{B}$ in the class of $\Sigma$-model structures.*

Finally we can use the completeness theorems obtained so far to prove a compactness theorem for our semantics.

**Corollary 5.5.8 (Compactness Theorem)** *Let $\Phi$ be a set of sentences, then $\Phi$ has a general $\Sigma$-model ($\Sigma$-model structure), iff every finite subset of $\Phi$ has a general $\Sigma$-model ($\Sigma$-model structure).*

**Proof:** Let every finite subset $\Psi$ of $\Phi$ be satisfiable by a general $\Sigma$-model, then $\Psi$ is $\Sigma\mathfrak{TE}$-consistent by 5.5.3, so $\Phi$ is $\Sigma\mathfrak{TE}$-consistent (every $\Sigma\mathfrak{TE}$-proof is finite), and thus satisfiable by a general $\Sigma$-model by 5.5.3.

For $\Sigma$-model structures we use the same argumentation with $\Sigma\mathfrak{T}\eta$ and 5.5.7. $\square$

# 6 $\Sigma\mathcal{HR}$: Resolution for $\Sigma\mathcal{HOL}$

In this section we present a sorted variant of Huet's "Constrained Resolution" calculus [Hue72], and prove it correct and complete with respect to $\Sigma$-model structures.

Since resolution calculi operate on formulae in clause normal form, we will begin with a discussion of an inference system $\mathcal{RC}$ that transforms arbitrary formulae into clause normal forms, conserving satisfiability. The only conceptually difficult step in this reduction is the one that deals with existential quantifications in the scope of universal quantifications. This is traditionally treated by a technique called Skolemization [Sko19], which is basically a syntactic trick that allows to employ the occurs-check of unification to reject any instantiation that does not obey the semantic restrictions imposed by $\forall\exists$-quantifications. As Andrews pointed out in [And73] naive Skolemization is not sound in higher-order logic. In fact, it is possible to prove an instance of the axiom of choice (which is known to be independent of higher-order logic) in the resolution systems [And71, Hue72] with naive Skolemization.

In his thesis [Mil83] Miller presents a sound version of Skolemization in the context of expansion trees and higher-order matings, and further developed the technique in [Mil91, Mil92] for the context of higher-order logic programming. Soundness of the refutation calculus given there is guaranteed by explicitly keeping track of the variable dependencies coming from the quantifier prefix and modifying the classical higher-order unification procedure to reject all solutions that do not conform to these restrictions. In a first-order setting a similar alternative to Skolemization has also been considered by Bibel in [Bib82]. In section 4 we have already introduced the mechanism of variable conditions, which we use for maintaining the satisfiability of generalized $\Sigma$-clauses during clause form reduction.

## 6.1 Reduction to Clause Normal Form

One of the most prominent features of resolution calculi is that they manipulate formulae in clause (conjunctive) normal form. The conjunctive normal form is a prenex normal form, where all existential quantifications have been eliminated and where the matrix has been transformed by DeMorgan laws such that the matrix is a conjunction of disjunctions and such that negations have minimal scope. This normal form is traditionally written in clause form where the quantifier prefix is dropped, and the matrix is written as a set of clauses, which are in turn sets of literals. This set notation emphasizes the commutativity, associativity, and idempotence of conjunction and disjunction.

In $\Sigma\mathcal{HR}$ we take $\Sigma$-clauses to be disjunctions of literals, which are just atomic formulae, labeled with their intended truth value. In contrast to the tradition in first-order resolution theorem proving, we do not eliminate existential quantifications by Skolemization, but rather use a variable condition to keep track of the dependencies. Finally, since $\Sigma$-unification is undecidable, we have to augment clauses with unification constraints that allow us to delay the computation of $\Sigma$-unifiers. These unification constraints of a $\Sigma$-clause are sets of negatively labeled equality literals.

**Definition 6.1.1 (Literal)** Let $\mathbf{A}$ be a proposition and $\alpha \in \{\mathtt{T}, \mathtt{F}\}$, then we call a pair $\mathbf{A}^\alpha$ a **labeled proposition**. A proposition $\mathbf{A}$ where $\mathbf{head}(\mathbf{A})$ is a parameter or variable is called **atomic**. Labeled propositions $\mathbf{A}^\alpha$ are called **literals**, if $\mathbf{A}$ is atomic. For the definition of $\Sigma$-clauses we will need a special kind of literals of the form $(\mathbf{A} =^? \mathbf{B})^{\mathsf{F}}$ where $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, $\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}$, and $\mathbb{A}$ **Rdom** $\mathbb{B}$. We call these literals **pairs**, since they serve the same purpose as pairs in unification problems, and we often write them as $\mathbf{A} \neq^? \mathbf{B}$ to conserve space. If we specifically want to reference literals that are not pairs, we call them **proper literals**.

**Definition 6.1.2 ($\Sigma$-Clause)** Let $\Gamma$ be an annotated variable context and $\mathcal{R}$ a variable condition for $\Gamma$. If $\Gamma \vdash_\Sigma \mathbf{M}_i::\mathbb{O}$ and $\alpha_i \in \{\mathtt{T}, \mathtt{F}\}$, then we call a formula $\mathcal{D} := \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C} \vee \mathcal{E}$ a **generalized $\Sigma$-clause**, if $\mathbf{C}$ is of the form $\mathbf{C} := \mathbf{M}_1^{\alpha_1} \vee \ldots \vee \mathbf{M}_n^{\alpha_n}$, and if $\mathcal{E}$ is a conjunction of pairs of the form $\mathbf{A}^1 \neq^? \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^m \neq^? \mathbf{B}^m$. We call $\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}$ the **clause part** of $\mathcal{D}$ and $\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}$ the **unification constraint** of $\mathcal{D}$. We call $\mathcal{C}$ a **$\Sigma$-clause**, iff the $\mathbf{M}_i^{\alpha_i}$ are literals. In the following we will identify $\Sigma$-clauses that only differ in the ordering of literals, and we will often treat $\Sigma$-clauses as sets or multisets of literals.

**Remark 6.1.3** Let $\overline{\mathcal{E}} := \mathbf{A}^1 =^? \mathbf{B}^1 \vee \ldots \vee \mathbf{A}^m =^? \mathbf{B}^m$ be the "syntactic negation" of the set $\mathcal{E}$ of pairs of $\mathcal{D}$, then $\mathcal{F} := \langle\Gamma\colon\mathcal{R}\rangle.\overline{\mathcal{E}}$ is a $\Sigma$-unification problem. Since this "syntactic negation" is only an adaptation to the context of $\Sigma$-clauses, where unification problems appear as constraints, we will often neglect this distinction, and apply all methods from section 4 directly to $\mathcal{E}$.

**Notation 6.1.4** We use the symbols $\mathbf{A}^\alpha, \mathbf{B}^\alpha, \ldots$ for labeled formulae and literals, $\mathcal{E}, \mathcal{F}, \ldots$ for disjunctions of pairs, and $\mathbf{C}, \mathbf{D}, \ldots$ for disjunctions of labeled formulae, literals, and pairs.

Since each generalized $\Sigma$-clause $\mathcal{C} = \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}$ determines a unique variable condition $\mathcal{R}_\Gamma$, we say that $\sigma$ is a **$\mathcal{C}$-substitution**, iff $\sigma$ is an $\mathcal{R}_\Gamma$-substitution.

**Definition 6.1.5 (Empty $\Sigma$-Clause)** We call a $\Sigma$-clause **initial**, iff its unification constraint is pre-$\Sigma$-solved, and **terminal**, iff if does not contain any proper literals, i.e. $n = 0$. In accordance with the practice from first-order resolution we call the class of $\Sigma$-clauses that are initial and terminal **empty**, since these play the role of the empty clause in our resolution calculus and we denote them collectively by $\Box$.

We present the process of transforming a sentence $\mathbf{A}$ into clause normal form as a calculus $\mathcal{RC}$, in order to facilitate the study of the interaction with the resolution calculus $\Sigma\mathcal{HR}$ defined below.

**Definition 6.1.6 (Reduction Rules ($\mathcal{RC}$))** The objects manipulated by the $\mathcal{RC}$-calculus are generalized $\Sigma$-clauses. Since $\mathcal{RC}$-derivations do not change the constraints of generalized $\Sigma$-clauses, we only show the effects on the formula part.

We use the rules with the convention that $\vee$ is associative (as we have already suggested by leaving out the parentheses) and commutative. Furthermore, after each application the formulae in the new $\Sigma$-clauses are reduced to sorted $\beta\eta$-normal form.

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\mathbf{A}\wedge\mathbf{B})^{\mathtt{T}}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathbf{A}^{\mathtt{T}}}\,\mathcal{RC}(\wedge l)\qquad\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\mathbf{A}\wedge\mathbf{B})^{\mathtt{T}}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathbf{B}^{\mathtt{T}}}\,\mathcal{RC}(\wedge r)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\mathbf{A}\wedge\mathbf{B})^{\mathtt{F}}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathbf{A}^{\mathtt{F}}\vee\mathbf{B}^{\mathtt{F}}}\,\mathcal{RC}(\vee)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\neg\mathbf{A})^{\mathtt{T}}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathbf{A}^{\mathtt{F}}}\,\mathcal{RC}(\neg^{\mathtt{T}})\qquad\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\neg\mathbf{A})^{\mathtt{F}}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathbf{A}^{\mathtt{T}}}\,\mathcal{RC}(\neg^{\mathtt{F}})$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\Pi^{\mathbb{A}}\mathbf{A})^{\mathtt{T}}}{\langle\Gamma,[X^{+}::\mathbb{A}]\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathbf{A}X^{\mathtt{T}}}\,\mathcal{RC}(\forall)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee(\Pi^{\mathbb{A}}\mathbf{A})^{\mathtt{F}}}{\langle\Gamma,[X^{-}::\mathbb{A}]\colon\mathcal{R}\cup(\mathbf{Free}(\mathbf{A})\times\{X^{-}\})\rangle.\mathbf{C}\vee(\mathbf{A}X^{-})^{\mathtt{F}}}\,\mathcal{RC}(\exists)$$

We can extend this calculus to act on sets of sets of generalized $\Sigma$-clauses. Since the notions are equivalent, we will always adopt the notion most convenient for our purposes.

**Lemma 6.1.7** *The reduction relation induced by $\mathcal{RC}$ on sets of generalized $\Sigma$-clauses is confluent, terminating, and the $\mathcal{RC}$-normal forms are $\Sigma$-clauses.*

**Proof:** For the confluence note that the rules of $\mathcal{RC}$ act only on one labeled proposition in the $\Sigma$-clause without changing the others, and applicability of the rules is determined by the head symbol of the chosen proposition.

By a simple induction over the number of logical constants that occur at top level in a generalized $\Sigma$-clause we observe that the $\mathcal{RC}$-rules can only be applied finitely often to a finite set of generalized $\Sigma$-clauses, so the reduction relation is terminating. $\square$

**Remark 6.1.8** Sometimes we do not want to exercise the idempotence of $\vee$ to collapse multiple occurences of literals in $\Sigma$-clauses in order to obtain tighter control over $\mathcal{RC}$-derivations in the proofs of the lifting lemmata. In these cases we use $\mathcal{RC}$ with an explicit inference rule for collapsing multiple occurrences of literals:

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{M}^{\alpha}\vee\mathbf{M}^{\alpha}\vee\mathbf{C}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{M}^{\alpha}\vee\mathbf{C}}\,\mathcal{RC}(coll)$$

**Definition 6.1.9 (Clause Normal Form)** Let $\mathcal{C}$ be a generalized $\Sigma$-clause, then we call the set $\mathbf{CNF}(\mathcal{C})$ of $\Sigma$-clauses that are derivable from $\mathcal{C}$ in $\mathcal{RC}$ the **clause normal form of** $\mathcal{C}$.

If $\Gamma$ is an annotated variable context, and $\mathbf{A}^\alpha$ is a labeled proposition such that $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{O}$, then we call the set $\mathbf{CNF}(\langle\Gamma:\emptyset\rangle.\mathbf{A}\!\downarrow^{\mathsf{T}})$ **clause normal form of A**, and denote it with $\mathbf{CNF}(\mathbf{A})$. Note that, since $\mathcal{RC}$ conserves long $\beta\eta$-normal forms, all literals in $\mathbf{CNF}(\mathbf{A})$ are in long $\beta\eta$-normal form as well. If $\Phi = \{\mathbf{A}_1, \ldots, \mathbf{A}_n\}$ is a set of sentences, then we call the set $\mathbf{CNF}(\Phi) := \bigcup_{i\leq n} \mathbf{CNF}(\langle\Gamma:\emptyset\rangle.\mathbf{A}_i^{\mathsf{T}})$ the **clause normal form of $\Phi$.**

**Remark 6.1.10** Note that only top level occurrences of propositional subformulae are considered in clause normal form. In general there be "buried" propositional subformulae in general form in $\Sigma$-clauses. For instance, if $\mathbf{A} := \forall X_{\mathbb{O}\to\mathbb{O}}\forall Y_{\mathbb{O}}.X(\neg\neg Y) \vee \neg(XY)$, then clearly $\vdash_\Sigma \mathbf{A}::\mathbb{O}$ and

$$\mathbf{CNF}(\mathbf{A}) = \langle[X^+::\mathbb{O} \to \mathbb{O}], [Y^+::\mathbb{O}]:\emptyset\rangle.(X.\neg\neg Y)^{\mathsf{T}} \vee .(XY)^{\mathsf{F}}$$

We now proceed to give a definition of validity for $\Sigma$-clauses that are the basis of the soundness considerations. This notion of validity takes positive variables in generalized $\Sigma$-clauses to be implicitly, universally quantified, and uses the notion of $\mathcal{R}_\Gamma$-correspondences as a semantic counterpart of variable conditions that specify the dependencies of variables recorded during the clause normal form transformation.

**Definition 6.1.11 (Validity for $\Sigma$-Clauses)** Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v)$ be a $\Sigma$-model structure, $\Gamma$ an annotated variable context, and $\mathcal{R}$ a variable condition for $\Gamma$. If $Y^- \in \mathbf{Dom}(\Gamma^-)$, $\{X_1^+, \ldots, X_n^+\} = \mathcal{R}^{-1}(Y)$, and $\Gamma(X_i) = \mathbb{A}_i$, then a total function $f_Y: \mathcal{D}_{\mathbb{A}_1} \times \cdots \times \mathcal{D}_{\mathbb{A}_n} \longrightarrow \mathcal{D}_{\Gamma(Y)}$ is called an **$\mathcal{R}_\Gamma$-function** for $Y$ in $\mathcal{M}$. We call a complete set $\{f_Y \mid Y \in \mathbf{Dom}(\Gamma^-)\}$ of $\mathcal{R}_\Gamma$-functions an **$\mathcal{R}_\Gamma$-correspondence** for $\mathcal{M}$. Note that in the case, where $n = 0$ $Y^- \in \mathbf{Dom}(\Gamma^-)$ is not in $\mathbf{Im}(\mathcal{R})$, but we still need an $f_Y \in \mathcal{D}_{\Gamma(Y)}$ in $\mathcal{F}$.

If $\mathcal{F}$ is an $\mathcal{R}_\Gamma$-correspondence for $\mathcal{M}$ and $\varphi$ is a $\Gamma$-assignment into $\mathcal{M}$, then we define the $\Gamma$-assignment $\varphi_\mathcal{F}$ by

$$\varphi_\mathcal{F}(Y) := \begin{cases} \varphi(Y), & \text{if} \quad Y \notin \mathbf{Dom}(\Gamma^-) \\ f_Y @\varphi(X_1)@ \cdots @\varphi(X_n), & \text{if} \quad Y \in \mathbf{Dom}(\Gamma^-) \text{ and} \\ & \qquad \{X_1, \ldots, X_n\} = \mathcal{R}^{-1}(Y) \end{cases}$$

Let $\mathcal{C} = \langle\Gamma:\mathcal{R}\rangle.\mathbf{C}$ be a generalized $\Sigma$-clause, $\varphi$ a $\Gamma$-assignment, and $\mathcal{F}$ an $\mathcal{R}_\Gamma$-correspondence for $\mathcal{M}$. We say that a labeled proposition $\mathbf{M}^\alpha$ in $\mathcal{C}$ is **satisfied by $\varphi$ in $\mathcal{M}$**, iff $v(\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{M})) = \alpha$, analogously for a pair $\mathbf{A} \neq^? \mathbf{B}$ in $\mathcal{C}$, iff $\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{A}) \neq \mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{B})$. We call $\mathcal{C}$ **valid in $\mathcal{M}$** ($\mathcal{M} \models \mathcal{C}$), iff there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ for $\mathcal{M}$ such that for all $\Gamma$-assignments $\varphi$ there is a labeled proposition or pair in $\mathcal{C}$ that is valid in $\mathcal{M}$.

A consequence of this definition, which regards positive variables as implicitly, universally quantified, is that the names of these do not carry any semantic meaning.

**Lemma 6.1.12 ($\alpha$-Conversion for $\Sigma$-Clauses)** *Let $\Gamma = \Delta, [X::\mathbb{A}]$ and $\Gamma' = \Delta, [Y::\mathbb{A}]$ be annotated variable contexts, and let $\mathcal{R}$ be a variable condition for $\Gamma$. Then for any $\Sigma$-model structure $\mathcal{M}$ we have $\mathcal{M} \models \langle\Gamma:\mathcal{R}\rangle.\mathbf{C}$, iff $\mathcal{M} \models \langle\Gamma':\mathcal{R}'\rangle.[Y/X]\mathbf{C}$ where $\mathcal{R}' = \mathcal{R}[Y/X]$.*

**Proof:** Let $\mathcal{C} = \langle\Gamma:\mathcal{R}\rangle.\mathbf{C}$, $\mathcal{C}' = \langle\Gamma':\mathcal{R}'\rangle.[Y/X]\mathbf{C}$, and $\mathcal{M} \models \mathcal{C}$, then there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ such that for all $\Gamma$-assignments $\varphi$ into $\mathcal{M}$, some labeled proposition or pair in $\mathcal{C}$ is satisfied by $\varphi$ in $\mathcal{M}$. Clearly $\mathcal{F}$ is also an $\mathcal{R}'_{\Gamma'}$-correspondence. Let

$\varphi' := \varphi, [\varphi(X)/Y]$, then $\varphi'$ is a $\Gamma'$-assignment into $\mathcal{M}$ with $\varphi'_{\mathcal{F}} := \varphi_{\mathcal{F}}, [\varphi(X)/Y]$. Thus for a labeled proposition $\mathbf{M}^\alpha$ in $\mathcal{C}$, we have $\mathcal{I}_{\varphi'_{\mathcal{F}}}(\mathbf{M}) = \alpha = \mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{M})$ ($\mathbf{M}^\alpha$ is satisfied by $\varphi'$), or some pair in $\mathcal{C}'$ is satisfied by $\varphi'$ in $\mathcal{M}$. Since we have chosen $\varphi$ arbitrarily for all $\Gamma'$-assignments $\varphi'$, there is a labeled proposition or pair in $\mathcal{C}'$ that is satisfied by $\varphi'$, so $\mathcal{C}'$ is valid in $\mathcal{M}$. □

**General Assumption 6.1.13** Just as in the case of $\Sigma$-unification problems (cf. 4.4.8) we consider the declaration $\langle\Gamma\colon\mathcal{R}\rangle$. in a $\Sigma$-clause as a binder for all variables in $\mathbf{Dom}(\Gamma)$, and we keep $\alpha$-conversion for $\Sigma$-clauses implicit, renaming them whenever variable disjointness is required.

**Remark 6.1.14** For generalized $\Sigma$-clauses of the form $\langle\emptyset\colon\emptyset\rangle.\mathbf{A}^{\mathtt{T}}$, the notion of validity for $\Sigma$-clauses coincides with the classical notion, as defined in 5.2.10. Indeed if the variable condition is empty, and the variable context does not contain negative variables, the variable correspondence must be empty too. Since $\mathbf{A}$ is a sentence, its validity is independent of the assignment considered.

**Lemma 6.1.15** *If $\Gamma$ is an annotated variable context, $\mathcal{R}$ is a variable condition for $\Gamma$ and $\Gamma \vdash_\Sigma \mathbf{C} =_{\beta\eta} \mathbf{D}$, then $\mathcal{M} \models \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}$, iff $\mathcal{M} \models \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{D}$ for any $\Sigma$-model structure $\mathcal{M}$.*

**Proof:** The assertion is a direct consequence of lemma 3.3.7 and the definition of validity for $\Sigma$-clauses. □

**Lemma 6.1.16** *Let $\mathcal{C} \vdash_{\mathcal{RC}} \mathcal{D}$ and $\mathcal{M}$ be a $\Sigma$-model structure, then $\mathcal{M} \models \mathcal{C}$, iff $\mathcal{M} \models \mathcal{D}$.*

**Proof:** Without loss of generality we can restrict ourselves to $\mathcal{RC}$-derivations of length 1, since the general case follows by a simple induction on the length. Also we only present the proof for the case where $\mathcal{C} \vdash_{\mathcal{RC}} \mathcal{D}$ by $\mathcal{RC}(\exists)$, since all others are unproblematic, because the variable condition is not altered by the transformation.

If $\mathcal{C} = \langle\Gamma\colon\mathcal{R}\rangle.(\Pi^{\mathbb{A}}\mathbf{A})^{\mathtt{F}} \vee \mathbf{C}$, then $\mathcal{D}$ must be of the form $\mathcal{D} = \langle\Gamma, [X^-{::}\mathbb{A}]\colon\mathcal{R}'\rangle.\mathbf{C} \vee (\mathbf{A}X^-)^{\mathtt{F}}$ up to sorted $\beta\eta$-conversion and $\mathcal{R}' := \mathcal{R} \cup (\mathbf{Free}(\mathbf{A}) \times \{X^-\})$.

If $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v) \models \mathcal{C}$, then there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ for $\mathcal{M}$ such that for all $\Gamma$-assignments $\varphi$ there is a labeled proposition or pair in $\mathcal{C}$ that is satisfied by $\varphi$ in $\mathcal{M}$. We can without loss of generality assume that $v(\mathcal{I}_{\varphi_{\mathcal{F}}}(\Pi^{\mathbb{A}}\mathbf{A})) = \mathtt{F}$, since otherwise the assertion is trivial. As $v$ is a $\Sigma$-valuation, there is an $a \in \mathcal{D}_{\mathbb{A}}$ such that $\mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A})@a = \mathtt{F}$, and thus $\mathcal{I}_\psi(\mathbf{A}X^-) = \mathtt{F}$, where $\psi := \varphi_{\mathcal{F}}, [a/X^-]$. Since for any $\psi'$ that agrees with $\psi$ on $\mathbf{Free}(\mathbf{A}) = \{X_1, \ldots, X_n\}$, we have $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_{\psi'}(\mathbf{A})$, this $a$ only depends on $\psi|_{\mathbf{Free}(\mathbf{A})} = \varphi_{\mathcal{F}}|_{\mathbf{Free}(\mathbf{A})}$. Since we have made no assumptions on $\varphi$, the set

$$f_X := \{(\psi(X_1), \ldots, \psi(X_n), a) \mid \varphi \text{ is a } \Gamma\text{−assignment}\}$$

is a total function, which makes $\mathcal{F}' := \mathcal{F} * f_X$ to an $\mathcal{R}'$-correspondence. Furthermore, we have $\psi = \varphi_{\mathcal{F}}, [a/X^-] = \varphi_{\mathcal{F}'}$, so $\mathcal{I}_{\varphi_{\mathcal{F}'}}(\mathbf{A}X^-) = \mathtt{F}$ for all $\Gamma$-assignments $\varphi$ into $\mathcal{M}$ and thus $\mathcal{M} \models \mathcal{D}$ by definition.

For the converse direction let $\mathcal{M} \models \mathcal{D}$. We assume the existence of an $\mathcal{R}'_{\Gamma,[X^-::\mathbb{A}]}$-correspondence $\mathcal{F}'$ for $\mathcal{M}$ such that for all $\Gamma, [X^-{::}\mathbb{A}]$-assignments $\varphi$ we have $v(\mathcal{I}_{\varphi_{\mathcal{F}'}}(\mathbf{A}X^-)) = \mathtt{F}$. Since $X^- \in \mathbf{Dom}(\Gamma^-, [X^-{::}\mathbb{A}])$ there must be a function $f_X\colon \mathcal{D}_{\Gamma(X_1)} \times \cdots \times \mathcal{D}_{\Gamma(X_n)} \longrightarrow$

$\mathcal{D}_{\Gamma(X^-)}$ in $\mathcal{F}'$. Let $\mathcal{F} := \mathcal{F}' \setminus \{f_X\}$, then $\mathcal{F}$ is an $\mathcal{R}$-correspondence and $\varphi_{\mathcal{F}'} = \varphi_{\mathcal{F}}, [f_X@\varphi(X_1)@\cdots@\varphi(X_n)/X^-]$. Thus $\mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A}X) = \mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A})@(f_X@\varphi(X_1)@\cdots@\varphi(X_n))$, and therefore $\upsilon(\mathcal{I}_{\varphi_{\mathcal{F}}}(\Pi^{\mathbb{A}}\mathbf{A})) = \mathtt{F}$, since $\upsilon$ is a $\Sigma$-valuation. Since we have taken $\varphi$ to be an arbitrary $\Gamma$-valuation, we have $\mathcal{M} \models \mathcal{C}$.

Since the $\mathcal{RC}$-rules are used with implicit subsequent $\beta\eta$-normalization, we need lemma 6.1.15 to complete the proof of the assertion. $\square$

If we instantiate this result with maximal $\mathcal{RC}$-derivations, we obtain the following clause normal form theorem.

**Theorem 6.1.17 (Clause Normal Form Theorem)** *Let $\Phi$ be a set of sentences, and let $\mathcal{M}$ be a $\Sigma$-model structure, then $\mathcal{M} \models \Phi$, iff $\mathcal{M} \models \mathbf{CNF}(\Phi)$.*

**Proof:** For any sentence $\mathbf{A} \in \Phi$ we have $\mathcal{M} \models \mathbf{A}$, iff $\mathcal{M} \models \langle\Gamma{:}\,\emptyset\rangle.\mathbf{A}^{\mathtt{T}}$ (cf. 6.1.14) and by 6.1.16 $\mathcal{M} \models \mathcal{C}$ for any $\Sigma$-clause $\mathcal{C} \in \mathbf{CNF}(\mathbf{A})$. We obtain the assertion by extending this argument to the set $\Phi$. $\square$

Note that this theorem is stronger than traditional variants for calculi that use Skolemization, which can only assert that satisfiability is preserved, since Skolem functions have to be given exactly one interpretation which entails an implicit uniqueness condition for the models of Skolemized formulae holds (that need not be valid in the models of the original formula). Now we convince ourselves that $\Sigma$-instantiation also conserves satisfiability with respect to $\Sigma$-model structures.

**Theorem 6.1.18** *Let $\Gamma$ be an annotated variable context such that $\Gamma^+(X) = \mathbb{A}$, and let $\mathcal{R}$ be a variable condition for $\Gamma$. If $\mathcal{M}$ is a $\Sigma$-model structure with $\mathcal{M} \models \langle\Gamma, [X^+{::}\mathbb{A}]{:}\,\mathcal{R}\rangle.\mathbf{C}$ and $\Gamma \vdash_\Sigma \overline{\mathcal{R}}(X^+, \mathbf{A})$, then $\mathcal{M} \models \langle\Gamma{:}\,\mathcal{R}[\mathbf{A}/X^+]\rangle.[\mathbf{A}/X^+]\mathbf{C}$.*

**Proof:** Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon)$, $\mathcal{C} = \langle\Gamma, [X^+{::}\mathbb{A}]{:}\,\mathcal{R}\rangle.\mathbf{C}$, and $\mathcal{C}' = \langle\Gamma{:}\,\mathcal{R}[\mathbf{A}/X^+]\rangle.[\mathbf{A}/X^+]\mathbf{C}$, then there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ for $\mathcal{M}$, such that for all $\Gamma$-assignments $\varphi$ there is a labeled proposition or pair in $\mathbf{C}$ that is satisfied by $\varphi$. Let $Y^- \in \mathbf{Dom}(\Gamma^-)$ with $\mathcal{R}^{-1}(Y) = \{X^-, X_2, \ldots, X_n\}$, then there is a function $f_Y \in \mathcal{F}$ with $f_Y{:}\,\mathcal{D}_{\Gamma(X^-)} \times \mathcal{D}_{\Gamma(X_2)} \times \cdots \times \mathcal{D}_{\Gamma(X_n)} \longrightarrow \mathcal{D}_{\Gamma(Y)}$ and $(\mathcal{R}[\mathbf{A}/X^-])^{-1}(Y) = \{Z_1, \ldots, Z_k, X_2, \ldots, X_n\}$, if $\mathbf{Free}(\mathbf{A}) = \{Z_1, \ldots, Z_k\}$. Furthermore, let

$$f'_Y{:}\,\mathcal{D}_{\Gamma(Z_1)} \times \cdots \times \mathcal{D}_{\Gamma(Z_k)} \times \mathcal{D}_{\Gamma(X_2)} \times \cdots \times \mathcal{D}_{\Gamma(X_n)} \longrightarrow \mathcal{D}_{\Gamma(Y)}$$

be defined by $f'_Y@a_1@\ldots@a_k := f_Y@\mathcal{I}_{[a_i/Z_i]}(\mathbf{A})$ and $f'_Y := f_Y$ for all $Y \in \mathbf{Dom}(\Gamma^-)$ with $X^- \notin \mathcal{R}^{-1}(Y)$, then $\mathcal{F}' := \{f'_Y \mid f_Y \in \mathcal{F}\}$ is an $\mathcal{R}[\mathbf{A}/X]_\Gamma$-correspondence and moreover, $f'_Y@\varphi(Z_1)@\cdots@\varphi(Z_n) = f_Y@\mathcal{I}_\varphi(\mathbf{A})$, so $\varphi_{\mathcal{F}'} = \varphi, [\mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A})/X]$, and therefore $\mathcal{I}_{\varphi_{\mathcal{F}'}}(\mathbf{M}) = \mathcal{I}_{\varphi_{\mathcal{F}}, [\mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A})/X]}(\mathbf{M}) = \alpha$ by the substitution value theorem 3.3.8. $\square$

**Lemma 6.1.19** *Any empty clause $\square$ is unsatisfiable with respect to $\Sigma$-model structures.*

**Proof:** We have defined empty clauses to be the initial and terminal clauses, thus $\square$ must be of the form $\langle\Gamma{:}\,\mathcal{R}\rangle.\mathcal{E}$, since it is terminal, and furthermore, the unification constraint $\mathcal{E} = \mathbf{A}_1 \neq^? \mathbf{B}_1 \vee \ldots \vee \mathbf{A}_n \neq^? \mathbf{B}_n$ must be pre-$\Sigma$-solved, since $\square$ is initial. Thus by 4.6.6 there is an $\mathcal{R}_\Gamma$-substitution $\theta$, that solves all pairs in $\square$. Thus if $\mathcal{M} \models \square$, then by 6.1.18 $\mathcal{M} \models \langle\Gamma{:}\,\mathcal{R}(\theta)\rangle.\theta(\mathbf{A}_1) \neq^? \theta(\mathbf{B}_1) \vee \ldots \vee \theta(\mathbf{A}_n) \neq^? \theta(\mathbf{B}_n)$, which is clearly impossible. $\square$

## 6.2 The Resolution Calculus $\Sigma\mathcal{HR}$

Now we turn to the actual resolution calculus $\Sigma\mathcal{HR}$. The previous results set the stage by giving a semantic justification of a resolution calculus that proves well-sorted sentences $\mathbf{A}$ by converting $\langle\emptyset\colon\emptyset\rangle.\mathbf{A}^{\mathsf{F}}$ to clause normal form and then by deriving the empty clause $\Box$ from that.

In contrast to Huet's calculus we allow pre-$\Sigma$-unification transformations to be applied to $\Sigma$-clauses during the resolution process. This generalization allows us to investigate more realistic strategies than in Huet's calculus, which uses the "lazy unification" strategy, that only allows unification to happen after a terminal $\Sigma$-clause has been derived.

**Definition 6.2.1 (Sorted Higher-Order Resolution ($\Sigma\mathcal{HR}$))** The calculus $\Sigma\mathcal{HR}$ is a variant of Huet's resolution calculus from [Hue72], and has the following rules of inference:

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{N}^{\alpha}\vee\mathbf{C}\quad\langle\Gamma'\colon\mathcal{R}'\rangle.\mathbf{M}^{\beta}\vee\mathbf{D}\quad\alpha\neq\beta}{\langle\Gamma,\Gamma'\colon\mathcal{R}\cup\mathcal{R}'\rangle.\mathbf{C}\vee\mathbf{D}\vee\mathbf{M}\neq^{?}\mathbf{N}}\,\Sigma\mathcal{HR}(Res)$$

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{M}^{\alpha}\vee\mathbf{N}^{\alpha}\vee\mathbf{C}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{M}^{\alpha}\vee\mathbf{C}\vee\mathbf{M}\neq^{?}\mathbf{N}}\,\Sigma\mathcal{HR}(Fac)$$

which operate on the clause part of $\Sigma$-clauses. For the $\Sigma\mathcal{HR}(Res)$ rule we assume that the contexts $\Gamma$ and $\Gamma'$ are disjoint. Note that this assumption does not result in a loss of generality, since we can always take a suitable $\alpha$-variant by 6.1.12. For manipulating the unification constraints $\Sigma\mathcal{HR}$ utilizes the $\Sigma\mathcal{PT}$ rules $\Sigma\mathcal{PT}(flex-rig)$ and $\Sigma\mathcal{PT}(guess)$ (cf. 4.6.7) by extending them to $\Sigma$-clauses in the obvious way. The following inference rule

$$\frac{\langle\Gamma,[F^{+}{::}\mathbb{A}]\colon\mathcal{R}\rangle.F\overline{\mathbf{U}^{k}}^{\alpha}\vee\mathbf{C}\quad\Gamma\vdash_{\Sigma}\overline{\mathcal{R}}(F^{+},\mathbf{P})}{\langle\Gamma,\mathcal{C},[P{::}\mathbb{A}]\colon\mathcal{R}[\mathbf{P}/X]\rangle.F\overline{\mathbf{U}^{k}}^{\alpha}\vee\mathbf{C}\vee F\neq^{?}\mathbf{P}}\,\Sigma\mathcal{HR}(Prim)$$

generates instantiations for flexible literals, i.e. literals where the head symbol is a positive variable. Here $\mathbf{P}\in\mathcal{A}^{k}_{\mathbb{A}}(\Sigma,\Gamma,\mathcal{C})$ is a general binding of sort $\mathbb{A}$ that approximates some logical constant $k\in\{\wedge,\neg,\Pi^{\mathbb{B}}\mid\mathbb{B}\in\mathcal{S}\}$. $\Sigma\mathcal{HR}$ has one further inference rule

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C}\vee\mathcal{E}\vee\mathcal{E}_{\sigma}}{\mathcal{C}}\,\Sigma\mathcal{HR}(Solv)$$

where $\mathcal{E}_{\sigma}$ is $\Sigma$-solved in $\mathcal{E}\vee\mathcal{E}_{\sigma}$ and $\mathcal{C}\in\mathbf{CNF}(\langle\Gamma\colon\mathcal{R}\rangle.\sigma(\mathbf{C})\vee\mathcal{E})$. This rule propagate partial solutions from the constraints to the clause part, and thus help detect clashes early. Since the instantiation may well change the propositional structure of the $\Sigma$-clause by instantiating a predicate variable, we have to renormalize the $\Sigma$-clause on the fly.

**Definition 6.2.2** We call a set $\Phi$ of well-formed sentences **$\Sigma\mathcal{HR}$-refutable**, iff $\Box$ is derivable from the set of $\Sigma$-clauses $\mathbf{CNF}(\Phi)$. A $\Sigma\mathcal{HR}$-derivation $\mathcal{R}$ of an empty $\Sigma$-clause $\Box$ from a set $\mathcal{C}$ of $\Sigma$-clauses is called a **$\Sigma\mathcal{HR}$-refutation of $\mathcal{C}$**. By a slight abuse of notation we call a sentence $\mathbf{A}$ $\Sigma\mathcal{HR}$-refutable, iff $\mathbf{CNF}(\langle\emptyset{:}\,\emptyset\rangle.\mathbf{A}^{\mathtt{F}})$ is $\Sigma\mathcal{HR}$-refutable.

**Example 6.2.3** Let $\Sigma := \{[c{::}\mathbb{O} \to \mathbb{O}], [b{::}\mathbb{O}]\}$, $\mathbf{A} := (cb)$, and $\mathbf{B} := c(\neg\neg b)$, then we can convince ourselves that there is no $\Sigma\mathcal{HR}$-refutation of $\mathbf{C} := \neg\mathbf{A} \wedge \mathbf{B}$: The clause normal form of $\mathbf{C}$ is $\{\langle\emptyset{:}\,\emptyset\rangle.(cb)^{\mathtt{F}}, \langle\emptyset{:}\,\emptyset\rangle.c(\neg\neg b)^{\mathtt{T}}$. Clearly the only rule that applies to $\mathbf{CNF}(\mathbf{C})$ is $\Sigma\mathcal{HR}(Res)$ yielding $\langle\emptyset{:}\,\emptyset\rangle.(cb) \neq^? c(\neg\neg b)$, which simplifies to the unsolvable constraint $\langle\emptyset{:}\,\emptyset\rangle.b \neq^? (\neg\neg b)$.

**Lemma 6.2.4** *Let $\Phi$ be a set of $\Sigma$-clauses and $\Phi \vdash_{\Sigma\mathcal{HR}} \mathcal{D}$, then for any $\Sigma$-model structure $\mathcal{M}$ we have $\mathcal{M} \models \mathcal{D}$, if $\mathcal{M} \models \Phi$.*

**Proof:** Let $\mathcal{D}{:}\,\Phi \vdash_{\Sigma\mathcal{HR}} \mathcal{D}$, we prove the assertion by induction over the structure of $\mathcal{D}$. If $\mathcal{D}$ is the empty $\Sigma\mathcal{HR}$-derivation, then the assertion is trivial. If $\mathcal{D}$ is obtained from $\Phi$ by $\Sigma\mathcal{HR}(Res)$, then we have the following situation:

$$\frac{\dfrac{\mathcal{D}}{\langle\Gamma{:}\,\mathcal{R}\rangle.\mathbf{N}^\alpha \vee \mathbf{C}} \quad \dfrac{\mathcal{D}'}{\langle\Gamma'{:}\,\mathcal{R}'\rangle.\mathbf{M}^\beta \vee \mathbf{D} \quad \alpha \neq \beta}}{\langle\Gamma, \Gamma'{:}\,\mathcal{R} \cup \mathcal{R}'\rangle.\mathbf{C} \vee \mathbf{D} \vee \mathbf{M} \neq^? \mathbf{N}}\ \Sigma\mathcal{HR}(Res)$$

By inductive hypothesis,there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ and an $\mathcal{R}'_{\Gamma'}$-correspondence $\mathcal{F}'$ such that for all $\Gamma$-assignments $\varphi$ and $\Gamma'$-assignments $\varphi'$ we have $\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{N}) = \alpha$ or $\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{L}) = \gamma$ for some $\mathbf{L}^\gamma \in \mathbf{C}$ and $\mathcal{I}_{\varphi'_{\mathcal{F}'}}(\mathbf{M}) = \beta$ or $\mathcal{I}_{\varphi'_{\mathcal{F}'}}(\mathbf{K}) = \delta$ for some $\mathbf{K}^\delta \in \mathbf{D}$.

Clearly $\mathcal{F} \cup \mathcal{F}'$ is an $\mathcal{R}_\Gamma \cup \mathcal{R}'_{\Gamma'}$-correspondence for $\Gamma, \Gamma'$, since we have assumed variable disjointness, and furthermore, any $\Gamma, \Gamma'$-assignment $\psi$ is of the form $\varphi \cup \varphi'$. We now convince ourselves, that $\mathcal{I}_{\psi_{\mathcal{F}\cup\mathcal{F}'}}(\mathbf{L}) = \gamma$ for some $\mathbf{L}^\gamma$ in $\mathbf{C} \vee \mathbf{D} \vee \mathbf{M} \neq^? \mathbf{N}$. We have two cases: if $\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{L}) = \gamma$ or $\mathcal{I}_{\varphi'_{\mathcal{F}'}}(\mathbf{K}) = \delta$, then the claim is trivial, in the other case we have

$$\mathcal{I}_{\psi_{\mathcal{F}\cup\mathcal{F}'}}(\mathbf{M}) = \mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{M}) = \alpha \neq \beta \mathcal{I}_{\varphi'_{\mathcal{F}'}}(\mathbf{N}) = \mathcal{I}_{\psi_{\mathcal{F}\cup\mathcal{F}'}}(\mathbf{N})$$

and in particular, $\mathcal{I}_{\psi_{\mathcal{F}\cup\mathcal{F}'}}(\mathbf{M} =^? \mathbf{N}) = \mathtt{F}$. Now the only remaining non-trivial case is that of $\Sigma\mathcal{HR}(Solv)$, since the case of $\Sigma\mathcal{HR}(Fac)$ is analogous to $\Sigma\mathcal{HR}(Res)$, and the rules $\Sigma\mathcal{HR}(Prim)$, $\Sigma\mathcal{PT}(flex-rig)$, and $\Sigma\mathcal{PT}(guess)$ only add pairs. This entails the assertion, since additional pairs weaken disjunctions.

The $\Sigma\mathcal{HR}(Solv)$ inference rule can be divided into two parts, first instantiating a $\Sigma$-clause $\mathcal{C} = \langle\Gamma{:}\,\mathcal{R}\rangle.\mathbf{C} \vee X \neq^? \mathbf{A}$ to a generalized $\Sigma$-clause $\mathcal{C}' := \langle\Gamma{:}\,\mathcal{R}[\mathbf{A}/X]\rangle.[\mathbf{A}/X]\mathbf{C}$, and then reducing it to clause normal form. Thus soundness of this case is a consequence of 6.1.18, the inductive hypothesis, and 6.1.17. $\Box$

**Theorem 6.2.5** *The $\Sigma\mathcal{HR}$ calculus is sound, i.e. if $\mathbf{A}$ is a well-sorted sentence such that $\mathbf{A}$ is $\Sigma\mathcal{HR}$-refutable, then $\mathbf{A}$ is unsatisfiable with respect to $\Sigma$-model structures.*

**Proof:** Let $\Phi = \mathbf{CNF}(\langle\emptyset{:}\,\emptyset\rangle.\mathbf{A})$. If $\Phi \vdash_{\Sigma\mathcal{HR}} \Box$, then $\Phi$ is unsatisfiable with respect to $\Sigma$-model structures by 6.2.4 and 6.1.19. Now the clause normal form theorem 6.1.17 gives the desired result. $\Box$

### 6.3 Lifting Properties for $\Sigma\mathcal{HR}$

A central part of the completeness proofs for unification-based refutation calculi are the lifting properties. The central lifting theorem for $\Sigma\mathcal{HR}$ states that, if $\theta(\mathbf{A})$ is $\Sigma\mathcal{HR}$-refutable, then there exists a (lifted) $\Sigma\mathcal{HR}$-refutation of the proposition $\mathbf{A}$. Note that, since $\Sigma\mathcal{HR}$ differs from Huet's constrained resolution in that we allow interleaving $\Sigma$-unification and resolution, the lifting lemmata become considerably more complex and take up most of the work in the completeness proof for $\Sigma\mathcal{HR}$.

The key technical device for the lifting property is the notion of a clause set isomorphism, which ties the structure of $\Sigma$-clauses in $\Sigma\mathcal{HR}$-derivations to the structure of the lifted $\Sigma\mathcal{HR}$-derivations.

**Definition 6.3.1 (Clause Set Isomorphism)** Let $\mathcal{C}$ and $\mathcal{C}'$ be generalized $\Sigma$-clauses such that $\Phi$ and $\Phi'$ are the respective sets of proper labeled formulae in $\mathcal{C}$ and $\mathcal{C}'$, then we call a bijection $\omega\colon \Phi \longrightarrow \Phi'$ a **clause isomorphism**, iff $\omega(\mathbf{M}^\alpha)$ is of the form $\mathbf{N}^\alpha$. If the constraints of $\mathcal{C}$ and $\mathcal{C}'$ are $\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}$ and $\langle\Gamma'\colon\mathcal{R}'\rangle.\mathcal{E}'$, and moreover $\theta$ is a $\mathcal{C}$-substitution, we say that $\omega$ is **$\theta$-compatible**, iff $\omega(\mathbf{M}) = \theta(\mathbf{M})$ and $\mathbf{wsPU}(\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}\vee\mathcal{E}_\theta) \subseteq \mathbf{wsPU}(\langle\Gamma'\colon\mathcal{R}'\rangle.\mathcal{E}')$.

Let $\Phi$ and $\Psi$ be sets of $\Sigma$-clauses, then a bijection $\omega\colon \Phi \longrightarrow \Psi$ together with a family of mappings $\omega_\mathcal{C}\colon\mathcal{C} \longrightarrow \omega(\mathcal{C})$ for all $\mathcal{C} \in \Phi$ is called a **clause set isomorphism**, iff all $\omega_\mathcal{C}$ are clause isomorphisms. We call $\omega$ **$\theta$-compatible**, iff all $\omega_\mathcal{C}$ are. Similarly we can define ($\theta$-compatible) isomorphisms of derivations as isomorphisms of the underlying trees such that corresponding nodes have clause isomorphisms.

We need the following technical lemma, which allow us to mimic $\mathcal{RC}$-derivations by $\Sigma\mathcal{HR}$-derivations using *$\Sigma\mathcal{HR}(Prim)$* inferences and factorization.

**Lemma 6.3.2 (Lifting Lemma for $\mathcal{RC}$)** *Let $\mathcal{C},\mathcal{C}_\theta,\widetilde{\mathcal{C}}_\theta$ be generalized $\Sigma$-clauses and let $\mathcal{C}_\theta \vdash_{\mathcal{RC}} \widetilde{\mathcal{C}}_\theta$ be a maximal $\mathcal{RC}$-derivation. Furthermore, let $\theta$ be a $\mathcal{C}$-substitution, and let $\omega\colon\mathcal{C} \longrightarrow \mathcal{C}_\theta$ be a $\theta$-compatible clause isomorphism. Then there exists a $\Sigma\mathcal{HR}$-derivation $\mathcal{C} \vdash_{\Sigma\mathcal{HR}} \widetilde{\mathcal{C}}$, a $\widetilde{\mathcal{C}}$-substitution $\widetilde{\theta}$ with $\widetilde{\theta} \doteq \theta[\mathbf{Dom}(\theta)]$, and a $\widetilde{\theta}$-compatible clause isomorphism $\widetilde{\omega}\colon\widetilde{\mathcal{C}} \longrightarrow \mathcal{C}_\theta$, so that we have the following commutative diagram.*

$$\begin{array}{ccc}
\mathcal{C}_\theta & \vdash_{\mathcal{RC}} & \widetilde{\mathcal{C}}_\theta \\
\theta\Big\uparrow\omega & & \widetilde{\theta}\Big\uparrow\widetilde{\omega} \\
\mathcal{C} & \vdash_{\Sigma\mathcal{HR}} & \widetilde{\mathcal{C}}
\end{array}$$

**Proof:** As remarked above the number of nodes in maximal $\mathcal{RC}$-derivations that do not exercise idempotence of $\vee$ is independent of the concrete $\Sigma\mathcal{HR}$-derivation. So let $\mathcal{D}_\theta\colon\mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{C}'_\theta \vdash_{\mathcal{RC}(coll)} \widetilde{\mathcal{C}}_\theta$ where the $\mathcal{RC}$-derivation $\widetilde{\mathcal{D}}_\theta\colon\mathcal{C}'_\theta \vdash_{\mathcal{RC}} \widetilde{\mathcal{C}}_\theta$ consists entirely of $\mathcal{RC}(coll)$ steps collapsing duplicate literals. We first construct a $\Sigma\mathcal{HR}$-derivation $\mathcal{R}'\colon\mathcal{C} \vdash_{\Sigma\mathcal{HR}} \mathcal{C}'$ by induction on the number of nodes in $\mathcal{D}_\theta$, and then extend this appropriately by *$\Sigma\mathcal{HR}(Fac)$* steps to account for idempotence.

If $\mathcal{D}_\theta$ is empty, then $\mathcal{C}'_\theta = \mathcal{C}_\theta$, so we obtain the assertion with $\mathcal{C}' := \mathcal{C}$, $\theta' := \theta$, and the empty $\Sigma\mathcal{HR}$-derivation. If $\mathcal{D}_\theta$ begins with an $\mathcal{RC}(\vee)$-step, then $\mathcal{D}_\theta$ is of the form

$$\cfrac{\mathcal{C}_\theta = \langle\Gamma\colon\mathcal{R}\rangle.\theta(\mathbf{F})^{\mathsf{F}} \vee \theta(\mathbf{C}) \vee \mathcal{F}}{\cfrac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{D}_1^{\mathsf{F}} \vee \mathbf{D}_2^{\mathsf{F}} \vee \theta(\mathbf{C}) \vee \mathcal{F}}{\langle\Gamma'\colon\mathcal{R}'\rangle.\mathbf{D} \vee \mathcal{F}}}\ \mathcal{RC}(\vee)$$

and $\mathcal{C} = \langle\Delta\colon\mathcal{Q}\rangle.\mathbf{F} \vee \mathbf{C} \vee \mathcal{E}$, since $\omega$ is a $\theta$-compatible clause isomorphism. Note that the head of $\mathbf{F}$ must be a positive variable $P^+$, since $\mathbf{F}^{\mathsf{T}}$ is a literal ($\mathbf{head}(\mathbf{F})$ cannot be a constant or negative variable $h$, since then $\mathbf{head}(\theta(\mathbf{F}))$ would be $h$, which contradicts our assumption). On the other hand, the head of $\theta(\mathbf{F})$ must be the constant $\vee$, since $\mathcal{RC}(\vee)$ acts on $\theta(\mathbf{F}^{\mathsf{T}})$, thus $\mathbf{head}(\theta(P))$ is the logical constant $\vee$, or $\theta(P)$ is a projection formula. Let $\Delta^+(P) = \mathbb{A}$, then there is a general binding $\mathbf{G} \in \mathcal{A}_{\mathbb{A}}^{\mathbf{head}(\theta(P))}(\Sigma, \Delta, \mathcal{C})$ and a $\Sigma$-substitution $\rho$ such that $\Delta \vdash_\Sigma \rho(\mathbf{G}) =_{\beta\eta} \theta(P)$ according to the general binding theorem (4.2.4). So let

$$\cfrac{\cfrac{\langle\Delta\colon\mathcal{Q}\rangle.\mathbf{F} \vee \mathbf{C} \vee \mathcal{E}}{\langle\Delta,\mathcal{C}\colon\mathcal{Q}[\mathbf{G}/F]\rangle.\mathbf{F} \vee \mathbf{C} \vee \mathcal{E} \wedge P \neq^? \mathbf{G}}\ \mathit{\Sigma\mathcal{HR}(Prim)}}{\langle\Delta'\colon\mathcal{Q}'\rangle.\mathbf{H}_1^{\alpha_1} \vee \ldots \vee \mathbf{H}_l^{\alpha_l} \vee [\mathbf{G}/P]\mathbf{C} \vee [\mathbf{G}/P]\mathcal{E}}\ \mathit{\Sigma\mathcal{HR}(Solv)}$$

where $\mathcal{H}\colon\langle\Delta,\mathcal{C}\colon\mathcal{Q}[\mathbf{G}/F]\rangle.[\mathbf{G}/P]\mathbf{F} \vdash_{\mathcal{RC}} \langle\Delta'\colon\mathcal{Q}'\rangle.\mathbf{H}_1^{\alpha_1} \vee \ldots \vee \mathbf{H}_l^{\alpha_l}$ is a maximal $\mathcal{RC}$-derivation in $\mathit{\Sigma\mathcal{HR}(Solv)}$. We have to consider the possibility that $l > 2$ in the conclusion of $\mathbf{R}$, since $\mathbf{G}$ may contain more than one logical constant, if the corresponding term declaration does. Clearly $\theta'' := \theta \cup \rho$ is a $\mathcal{Q}[\mathbf{G}/P]_{\Delta,\mathcal{C}}$ substitution. Furthermore, we have $\Delta \vdash_\Sigma \theta(P^+) =_{\beta\eta} \rho(\mathbf{G})$, and therefore

$$\mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.[\mathbf{G}/P]\mathcal{E} \wedge P =^? \mathbf{G} \wedge \mathcal{E}_{\theta''}) \subseteq \mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{E} \wedge \mathcal{E}_\theta)$$

by 4.4.15, and thus

$$\mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{E} \wedge P =^? \mathbf{G} \wedge \mathcal{E}_{\theta''}) \subseteq \mathbf{wsPU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{F})$$

since $\omega$ is $\theta$-compatible, so $\omega$ is also $\theta''$-compatible. Since $[\mathbf{G}/P]$ approximates $\theta$, the same $\mathcal{RC}$-derivations apply to $\theta(\mathbf{F})$ and $[\mathbf{G}/P]\mathbf{F}$. Thus by a simple induction on the length of $\mathcal{H}$ we obtain a nonempty $\mathcal{RC}$-derivation $\mathcal{D}_{\theta''}\colon\mathcal{C}_\theta \vdash \mathcal{C}''_\theta$ and a $\theta''$-compatible clause isomorphism $\omega''\colon\mathcal{C}'' \longrightarrow \mathcal{C}''_\theta$. Finally, we have $\langle\Delta,\mathcal{C}\colon\mathcal{Q}[\mathbf{G}/P]\rangle.[\mathbf{G}/P]\mathbf{F} \vee [\mathbf{G}/P]\mathbf{C} \vdash_{\mathcal{RC}} \mathcal{C}''$ by a maximal $\mathcal{RC}$-derivation that is nonempty, as the head of $\theta(\mathbf{F})$ is $\vee$.

Since we do not exercise idempotence of $\vee$ and $\mathcal{RC}$ is confluent, there must be an $\mathcal{RC}$-derivation $\mathcal{D}''_\theta\colon\mathcal{C}''_\theta \vdash_{\mathcal{RC}} \mathcal{C}'_\theta$ that has fewer nodes than $\mathcal{D}_\theta$. Thus we obtain the assertion with the inductive hypothesis by combining $\Sigma\mathcal{HR}$-derivations according to the following diagram:

$$\begin{array}{ccccc}
\mathcal{C}_\theta & \vdash_{\mathcal{RC}} & \mathcal{C}''_\theta & \vdash_{\mathcal{RC}} & \mathcal{C}'_\theta \\
\theta \Big\uparrow \omega & & \theta'' \Big\uparrow \omega'' & & \theta' \Big\uparrow \omega' \\
\mathcal{C} & \vdash_{\mathit{\Sigma\mathcal{HR}(Prim)}} & \mathcal{C}'' & \vdash^{IH}_{\Sigma\mathcal{HR}} & \mathcal{C}'
\end{array}$$

This completes our first goal for the $\mathcal{RC}(\vee)$ case. Let us recapitulate the argumentation: we have started out with an $\mathcal{RC}(\vee)$ node in $\mathcal{D}_\theta$ and have simulated that by a $\Sigma\mathcal{HR}(Prim)$ step, using a general binding $\mathbf{G}$ that approximates the head $\vee$. Since $\mathbf{G}$ can have more logical constants, the reduction of clause normal form in the subsequent $\Sigma\mathcal{HR}(Solv)$ step can be an $\mathcal{RC}$-derivation of length greater than one. Fortunately the same $\mathcal{RC}$ inference rules apply to both generalized clauses. So we have identified a $\mathcal{RC}$-derivation $\mathcal{D}''_\theta$, and obtained the assertion by applying the inductive hypothesis to the remaining reduction. The remaining cases are similar, and can be solved with the same methods. With this inductive argument we have constructed a $\Sigma\mathcal{HR}$-derivation $\mathcal{D}'{:}\,\mathcal{C} \vdash_{\mathcal{RC}} \mathcal{C}'$ and a $\theta'$-compatible clause isomorphism $\omega'{:}\,\mathcal{C}' \longrightarrow \mathcal{C}'_\theta$. We did not exercise the idempotence of $\vee$, since we would had more problems maintaining the clause isomorphisms. Accounting for this is the next and final step of the proof.

Now $\widetilde{\mathcal{C}}_\theta$ is obtained from $\mathcal{C}'_\theta$ by collapsing duplicate literals, which we simulate by $\Sigma\mathcal{HR}(Fac)$ steps on the more general level. Let $\widetilde{\theta} := \theta'$, we proceed by constructing a $\widetilde{\theta}$-compatible clause isomorphism $\widetilde{\omega}$ from $\omega'$ by induction on the structure $\widetilde{\mathcal{D}}_\theta{:}\,\mathcal{C}'_\theta \vdash_{\mathcal{RC}} \widetilde{\mathcal{C}}_\theta$. If $\widetilde{\mathcal{D}}_\theta$ is empty, then we take $\widetilde{\omega} := \omega'$, and there is nothing to do. If $\widetilde{\mathcal{C}}_\theta$ is nonempty, then it must be of the form

$$\cfrac{\cfrac{\mathcal{C}'_\theta}{\langle\Gamma{:}\,\mathcal{R}\rangle.\widetilde{\theta}(\mathbf{M}^\alpha) \vee \widetilde{\theta}(\mathbf{N}^\alpha) \vee \widetilde{\theta}(\mathbf{C}) \vee \mathcal{F}}}{\langle\Gamma{:}\,\mathcal{R}\rangle.\widetilde{\theta}(\mathbf{M}^\alpha) \vee \widetilde{\theta}(\mathbf{C}) \vee \mathcal{F}}\ \mathcal{RC}(coll)$$

where $\widetilde{\theta}(\mathbf{M}) \doteq \widetilde{\theta}(\mathbf{N})$, then we can imitate the step with the following one

$$\cfrac{\cfrac{\mathcal{C}'}{\langle\Delta'{:}\,\mathcal{Q}'\rangle.\mathbf{M}^\alpha \vee \mathbf{N}^\alpha \vee \mathbf{C} \vee \mathcal{E}}}{\langle\Delta'{:}\,\mathcal{Q}'\rangle.\mathbf{M}^\alpha \vee \mathbf{C} \vee \mathbf{M} \neq^? \mathbf{N} \vee \mathcal{E}}\ \Sigma\mathcal{HR}(Fac)$$

and inductively obtain a $\Sigma\mathcal{HR}$-derivation $\widetilde{\mathcal{D}}{:}\,\mathcal{C}' \vdash_{\Sigma\mathcal{HR}} \widetilde{\mathcal{C}}$. Furthermore, $\widetilde{\omega} := \omega_{-\mathbf{N}^\alpha}$ is a $\widetilde{\theta}$-compatible clause isomorphism, since

$$\begin{aligned}\mathbf{wsPU}(\Sigma, \langle\Delta'{:}\,\mathcal{Q}'\rangle.\mathcal{E} \wedge \mathbf{M} =^? \mathbf{N} \wedge \mathcal{E}_{\theta'}) &= \mathbf{wsPU}(\Sigma, \langle\Delta'{:}\,\mathcal{Q}'\rangle.\mathcal{E} \wedge \widetilde{\theta}(\mathbf{M}) =^? \widetilde{\theta}(\mathbf{N}) \wedge \mathcal{E}_{\widetilde{\theta}})\\ &= \mathbf{wsPU}(\Sigma, \langle\Gamma{:}\,\mathcal{R}\rangle.\mathcal{E} \wedge \mathcal{E}_{\widetilde{\theta}})\end{aligned}$$

by 4.4.15 and the fact that $\widetilde{\theta}(\mathbf{M}) \doteq \widetilde{\theta}(\mathbf{N})$, so we obtain the assertion by inductive hypothesis. Finally, it only remains to combine the two $\Sigma\mathcal{HR}$-derivations, we have constructed in this proof, according to the following commutative diagram:

$$\begin{tikzcd}
\mathcal{C}_\theta & \vdash_{\mathcal{RC}} & \mathcal{C}'_\theta & \vdash_{\mathcal{RC}} & \widetilde{\mathcal{C}}_\theta \\
\mathcal{C} \arrow[u, "\theta", "\omega"'] & \vdash_{\Sigma\mathcal{HR}} & \mathcal{C}' \arrow[u, "\theta'", "\omega'"'] & \vdash_{\Sigma\mathcal{HR}} & \widetilde{\mathcal{C}} \arrow[u, "\widetilde{\theta}", "\widetilde{\omega}"']
\end{tikzcd}$$

□

**Lemma 6.3.3 ($\mathcal{RC}$-Normalization Lifting Lemma)** *Let $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{O}$ and $\Gamma \vdash_\Sigma \theta::\Delta^+$, then there is a $\Sigma\mathcal{HR}$-derivation $\mathcal{D}$ of a set $\Xi$ of initial $\Sigma$-clauses from $\mathbf{CNF}(\mathbf{A}^\alpha)$, a $\Sigma$-substitution $\widetilde{\theta}$, and a $\widetilde{\theta}$-compatible clause set isomorphism $\widetilde{\omega}\colon \Phi \longrightarrow \mathbf{CNF}(\theta(\mathbf{A})^\alpha)$, so that we have the following commutative diagram.*

$$\begin{tikzcd}
\theta(\mathbf{A})^\alpha & & \vdash_{\mathcal{RC}} & & \mathbf{CNF}(\theta(\mathbf{A})^\alpha) \\
\mathbf{A}^\alpha \arrow[u, "\theta"'] & \vdash_{\mathcal{RC}} & \mathbf{CNF}(\mathbf{A}^\alpha) & \vdash_{\Sigma\mathcal{HR}} & \Phi \arrow[u, "\widetilde{\theta}", "\widetilde{\omega}"']
\end{tikzcd}$$

**Proof:** Let $\Theta = \mathbf{CNF}(\theta(\mathbf{A})^\alpha)$ and $\Psi = \mathbf{CNF}(\mathbf{A}^\alpha)$. Furthermore, let $\mathcal{C} \in \Psi$ be a $\Sigma$-clause and $\mathcal{R}\colon \langle\Delta\colon\emptyset\rangle.\mathbf{A}^\alpha \vdash_{\mathcal{RC}} \mathcal{C}$ its reduction to clause form. Clearly the same $\mathcal{RC}$-reductions also apply to $\langle\Gamma\colon\emptyset\rangle.\theta(\mathbf{A}^\alpha)$, since if $\mathbf{head}(\mathbf{A}) \in \{\vee, \neg, \Pi^{\mathbb{A}}\}$, then $\mathbf{head}(\mathbf{A}) = \mathbf{head}(\theta(\mathbf{A}))$. Thus we have an $\mathcal{RC}$-derivation $\mathcal{D}\colon \langle\Gamma\colon\emptyset\rangle.\theta(\mathbf{A}^\alpha) \vdash_{\mathcal{RC}} \mathcal{C}_\theta$ and a clause isomorphism $\omega\colon \mathcal{C} \longrightarrow \mathcal{C}_\theta$. As the constraint parts of $\mathcal{C}$ and $\mathcal{C}_\theta$ are empty, $\omega$ is trivially $\theta$-compatible.

Now $\mathcal{C}_\theta$ need not be a $\Sigma$-clause yet, so let $\mathcal{R}\colon \mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{C}'_\theta \in \Theta$ be a maximal $\mathcal{RC}$-derivation. By the lifting lemma for $\mathcal{RC}$ (6.3.2) there is a $\Sigma$-clause $\mathcal{C}'$, a $\Sigma\mathcal{HR}$-derivation $\mathcal{D}\colon \mathcal{C} \vdash_{\Sigma\mathcal{HR}} \mathcal{C}'$, and a $\theta'$-compatible clause isomorphism $\omega'\colon \mathcal{C}' \longrightarrow \mathcal{C}'_\theta$.

Thus for any $\Sigma$-clause $\mathcal{C} \in \Psi$ and each $\Sigma$-clause $\mathcal{C}_\theta \in \Theta$ we have a $\Sigma\mathcal{HR}$-derivation $\mathcal{C} \vdash_{\Sigma\mathcal{HR}} \mathcal{C}'$ and a $\theta'$-compatible clause isomorphism $\omega'\colon \mathcal{C}' \longrightarrow \mathcal{C}'_\theta$. Hence we obtain the assertion by collecting all such $\Sigma$-clauses $\mathcal{C}'$ in the set $\Phi$. If we take care to keep the domains of the contexts in the respective derivations disjoint, then the $\Sigma$-substitutions $\theta'$ all have the form $\theta \cup \rho$, where all $\rho$ have disjoint domains, so we can construct a single $\Sigma$-substitution $\widetilde{\theta} := \theta \cup \bigcup_\Theta \rho$, which verifies the assertion. $\square$

**Lemma 6.3.4 (Lifting Lemma for $\Sigma\mathcal{HR}$)** *Let $\omega\colon \Psi \longrightarrow \Theta$ be a $\theta$-compatible clause set isomorphism and $\mathcal{D}\colon \Theta \vdash_{\Sigma\mathcal{HR}} \mathcal{C}_\theta$ such that the constraint of $\mathcal{C}_\theta$ is pre-$\Sigma$-unifiable, then there is a $\Sigma\mathcal{HR}$-derivation $\Psi \vdash_{\Sigma\mathcal{HR}} \mathcal{C}$ and a $\theta'$-compatible clause isomorphism $\omega'\colon \mathcal{C} \longrightarrow \mathcal{C}_\theta$ for a $\Sigma$-substitution $\theta'$, so that we have the following commutative diagram.*

$$\begin{tikzcd}
\Theta & \vdash_{\Sigma\mathcal{HR}} & \mathcal{C}_\theta \\
\Psi \arrow[u, "\theta", "\omega"'] & \vdash_{\Sigma\mathcal{HR}} & \mathcal{C} \arrow[u, "\theta'", "\omega'"']
\end{tikzcd}$$

**Proof:** We prove the assertion by induction on the structure of $\mathcal{D}$. If $\mathcal{D}$ is the empty tree, then we choose $\omega' := \omega$ and $\theta' := \theta$, and obtain the assertion from the definition of clause set isomorphism.

If $\mathcal{D}$ ends in a $\Sigma\mathcal{HR}(Res)$ step, then we have the following situation:

$$\dfrac{\dfrac{\Theta}{\mathcal{C}'_\theta}\,\mathcal{D}' \quad \dfrac{\Theta}{\mathcal{C}''_\theta}\,\mathcal{D}''}{\mathcal{C}_\theta}\;\Sigma\mathcal{HR}(Res)$$

By inductive hypothesis we have $\Sigma\mathcal{HR}$-derivations $\mathcal{R}'\colon \Psi \vdash_{\Sigma\mathcal{HR}} \mathcal{C}'$ and $\mathcal{R}''\colon \Psi \vdash_{\Sigma\mathcal{HR}} \mathcal{C}''$ and $\theta'$-compatible clause isomorphisms $\omega'\colon \mathcal{C}' \longrightarrow \mathcal{C}'_\theta$ and $\omega''\colon \mathcal{C}'' \longrightarrow \mathcal{C}''_\theta$ for a $\Sigma$-substitution $\theta'$.

We can restrict ourselves to a single $\Sigma$-substitution $\theta'$ here, since the parent $\Sigma$-clauses can be renamed to suitable $\alpha$-variants that have disjoint variable contexts. Since $\omega'$ and $\omega''$ are clause isomorphisms, we must have $\mathcal{C}' = \langle\Delta'\colon \mathcal{Q}'\rangle.\mathbf{A}^\alpha \vee \mathbf{C}' \vee \mathcal{E}'$ and $\mathcal{C}'' = \langle\Delta'\colon \mathcal{Q}'\rangle.\mathbf{B}^\beta \vee \mathbf{C}'' \vee \mathcal{E}''$ with $\alpha \neq \beta$, and $\mathcal{D}$ must be of the form

$$\dfrac{\dfrac{\Theta}{\dfrac{}{\langle\Gamma'\colon\mathcal{R}'\rangle.\theta(\mathbf{A})^\alpha \vee \theta(\mathbf{C}') \vee \mathcal{F}'}}\,\mathcal{D}' \qquad \dfrac{\Theta}{\dfrac{}{\langle\Gamma''\colon\mathcal{R}''\rangle.\theta(\mathbf{B})^\beta \vee \theta(\mathbf{C}'') \vee \mathcal{F}''}}\,\mathcal{D}''}{\langle\Gamma',\Gamma''\colon\mathcal{R}'\cup\mathcal{R}''\rangle.\theta(\mathbf{C}') \vee \theta(\mathbf{C}'') \vee \mathcal{F}' \vee \mathcal{F}'' \vee \theta(\mathbf{A}) \neq^? \theta(\mathbf{B})}\,\Sigma\mathcal{HR}(Res)$$

Let $\mathcal{R}$ be the following $\Sigma\mathcal{HR}$-derivation:

$$\dfrac{\dfrac{\Psi}{\dfrac{}{\langle\Delta'\colon\mathcal{Q}'\rangle.\mathbf{A}^\alpha \vee \mathbf{C}' \vee \mathcal{E}'}}\,\mathcal{R}' \qquad \dfrac{\Theta}{\dfrac{}{\langle\Delta''\colon\mathcal{Q}''\rangle.\mathbf{B}^\beta \vee \mathbf{C}'' \vee \mathcal{E}''}}\,\mathcal{R}''}{\langle\Delta',\Delta''\colon\mathcal{Q}'\cup\mathcal{Q}''\rangle.\mathbf{C}' \vee \mathbf{C}'' \vee \mathcal{E}' \vee \mathcal{E}'' \vee \mathbf{A} \neq^? \mathbf{B}}\,\Sigma\mathcal{HR}(Res)$$

Now we trace the clause isomorphism through the $\Sigma\mathcal{HR}$-derivation. Let us first treat $\Sigma$-clauses as multisets to avoid the hassle with factoring. By $\theta'$-compatibility from the inductive hypothesis we have

$$\begin{aligned}
\mathbf{wsPU}(\Sigma, \langle\Delta'\colon\mathcal{Q}'\rangle.\mathcal{E}' \wedge \mathcal{E}_{\theta'}) &\subseteq \mathbf{wsPU}(\Sigma, \langle\Gamma'\colon\mathcal{R}'\rangle.\mathcal{F}') \\
\mathbf{wsPU}(\Sigma, \langle\Delta''\colon\mathcal{Q}''\rangle.\mathcal{E}'' \wedge \mathcal{E}_{\theta'}) &\subseteq \mathbf{wsPU}(\Sigma, \langle\Gamma''\colon\mathcal{R}''\rangle.\mathcal{F}')
\end{aligned}$$

and moreover,

$$\mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.\mathbf{A} =^? \mathbf{B} \wedge \mathcal{E}_\theta) = \mathbf{wsPU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\theta(\mathbf{A}) =^? \theta'(\mathbf{B}) \wedge \mathcal{E}_{\theta'})$$

by 4.4.15, if we set $\Delta := \Delta',\Delta''$, $\mathcal{Q} := \mathcal{Q}'\cup\mathcal{Q}''$, $\Gamma := \Gamma',\Gamma''$, and $\mathcal{R} := \mathcal{R}'\cup\mathcal{R}''$, so that

$$\begin{aligned}
&\mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{E}' \wedge \mathcal{E}'' \wedge \mathbf{E} =^? \mathbf{F} \wedge \mathcal{E}_{\theta'}) \\
&= \mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{E}' \wedge \mathcal{E}'' \wedge \theta(\mathbf{E}) =^? \theta(\mathbf{F}) \wedge \mathcal{E}_{\theta'}) \\
&\subseteq \mathbf{wsPU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E} \wedge \mathcal{F} \wedge \theta(\mathbf{E}) =^? \theta(\mathbf{F}))
\end{aligned}$$

which proves that the obvious choice for $\omega$ is $\theta'$-compatible. The $\Sigma\mathcal{HR}(Fac)$ case is analogous.

$\Sigma\mathcal{HR}(Prim)$, $\Sigma\mathcal{PT}(flex-rig)$, and $\Sigma\mathcal{PT}(guess)$ share the following common structure: all introduce a general binding $F =^? \mathbf{G}$ for a variable $F \notin \mathbf{Dom}(\theta)$. Since $F \notin \mathbf{Dom}(\theta)$, we can lift the rule applications directly (i.e. create a more general $\Sigma\mathcal{HR}$-derivation by introducing exactly the same general binding), and obtain the assertion by the methods we have exemplified for the $\Sigma\mathcal{HR}(Res)$ case above. For the $\Sigma\mathcal{HR}(Solv)$ case we have the following situation

$$\dfrac{\langle\Gamma\colon\mathcal{R}\rangle.\theta(\mathbf{C}) \vee \mathcal{F} \vee F \neq^? \mathbf{A}}{\langle\Gamma\colon\mathcal{R}[\mathbf{A}/F]\rangle.\mathbf{D} \vee [\mathbf{A}/F]\mathcal{F} \vee F \neq^? \mathbf{A}}\,\Sigma\mathcal{HR}(Solv)$$

where $[\mathbf{Z}/F]\theta(\mathbf{C}) \vdash_{\mathcal{RC}} \mathbf{D}$, so we obtain the assertion with 6.3.2. ☐

**Theorem 6.3.5 ($\Sigma\mathcal{HR}$-Refutation Lifting)** *Let $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{O}$ and $\Gamma \vdash_\Sigma \theta::\Delta$, then $\mathbf{A}$ is $\Sigma\mathcal{HR}$-refutable, if $\theta(\mathbf{A})$ is $\Sigma\mathcal{HR}$-refutable.*

**Proof:** Let $\Theta = \mathbf{CNF}(\theta(\mathbf{A}))$ and $\Psi = \mathbf{CNF}(\mathbf{A})$, then by the $\mathcal{RC}$-normalization lifting lemma 6.3.3 there is a set $\Phi$ of $\Sigma$-clauses, a $\Sigma$-substitution $\theta'$ with $\theta \doteq \theta'[\mathbf{Dom}(\theta)]$, and a $\theta'$-compatible clause set isomorphism $\omega\colon \Theta \longrightarrow \Phi$. Furthermore, there is a $\Sigma\mathcal{HR}$-derivation $\mathcal{H}\colon \Psi \vdash_{\Sigma\mathcal{HR}} \Phi$. We continue the proof according to the following diagram:

$$
\begin{array}{ccccccccc}
\theta(\mathbf{A}) & & \vdash_{\mathcal{RC}} & & \Theta & \vdash^{\mathcal{D}}_{\Sigma\mathcal{HR}} & \Box & & \\
\uparrow{\scriptstyle \theta} & & & & {\scriptstyle \theta'}\uparrow{\scriptstyle \omega'} & & {\scriptstyle \theta''}\uparrow{\scriptstyle \omega''} & & \\
\mathbf{A} & \vdash_{\mathcal{RC}} & \mathbf{CNF}(\mathbf{A}) & \vdash^{\mathcal{H}}_{\Sigma\mathcal{HR}} & \Phi & \vdash_{\Sigma\mathcal{HR}} & \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{F}\vee\mathcal{F} & \vdash_{\Sigma\mathcal{HR}} & \Box
\end{array}
$$

Let $\mathcal{D}$ be the $\Sigma\mathcal{HR}$-refutation of $\theta(\mathbf{A})$, i.e. $\mathcal{D}\colon \Theta \vdash_{\Sigma\mathcal{HR}} \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}$ where $\mathcal{E}$ is pre-$\Sigma$-solved, then by the lifting lemma for $\Sigma\mathcal{HR}$ (6.3.4) there is a $\Sigma\mathcal{HR}$-derivation $\mathcal{R}'\colon \Psi \vdash_{\Sigma\mathcal{HR}} \langle\Delta\colon\mathcal{Q}\rangle.\mathbf{C}\vee\mathcal{F}$ and a $\theta''$-compatible clause isomorphism $\omega''\colon \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{F} \longrightarrow \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}$. Thus $\mathbf{C}$ is the empty disjunction and $\mathbf{wsPU}(\Sigma, \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{E}\wedge\mathcal{E}_{\theta''}) = \mathbf{wsPU}(\Sigma, \langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E})$, which was nonempty by assumption. However, each pre-$\Sigma$-unifier of the subproblem $\mathcal{F}$ also unifies the system $\mathcal{F}\wedge\mathcal{E}_{\theta''}$. Consequently, there is a $\Sigma\mathcal{PT}$-derivation that transforms $\mathcal{F}$ to pre-$\Sigma$-solved form by 4.6.10. Let $\mathcal{R}''$ be the corresponding $\Sigma\mathcal{HR}$-derivation, then we obtain a $\Sigma\mathcal{HR}$-refutation of $\mathbf{A}$ by $\mathcal{H}\colon \Psi \vdash^{\mathcal{H}}_{\Sigma\mathcal{HR}} \Psi \vdash^{\mathcal{R}'}_{\Sigma\mathcal{HR}} \langle\Delta\colon\mathcal{Q}\rangle.\mathcal{F} \vdash^{\mathcal{R}''}_{\Sigma\mathcal{HR}} \Box$, that validates the assertion. $\Box$

**Definition 6.3.6 (Tautology)** A $\Sigma$-clause $\mathcal{C} = \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{M}^{\mathtt{T}}\vee\mathbf{N}^{\mathtt{F}}\vee\mathbf{C}\vee\mathcal{E}$ is called a **tautology**, if $\Gamma \vdash_\Sigma \mathbf{M}=_{\beta\eta}\mathbf{N}$ and $\langle\Gamma\colon\mathcal{R}\rangle.\mathcal{E}$ is pre-$\Sigma$-solved. $\mathcal{C}$ is called **elementary**, iff $\mathbf{C}$ empty.

**Remark 6.3.7** If $\mathcal{C}$ is an elementary tautology and $\Phi, \mathcal{C} \vdash_{\Sigma\mathcal{HR}} \Box$, then $\Phi \vdash_{\Sigma\mathcal{HR}} \Box$. By the lifting lemma (6.3.4) we have a $\Sigma\mathcal{HR}$-refutation $\Phi * \mathcal{C}'$ where $\mathcal{C}' = \langle[P::\mathbb{O}]\colon\emptyset\rangle.P^{\mathtt{F}}\vee P^{\mathtt{T}}$. Note that the only inference rule that can be applied to $\mathcal{C}'$ is a $\Sigma\mathcal{HR}(Res)$ step of the form

$$
\frac{\mathcal{D} = \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{B}^{\alpha}\vee\mathbf{C} \quad \langle[P::\mathbb{O}]\colon\emptyset\rangle.P^{\mathtt{F}}\vee P^{\mathtt{T}}}{\mathcal{D}' = \langle\Gamma,[P::\mathbb{O}]\colon\mathcal{R}\rangle.\mathbf{C}\vee P^{\alpha}\vee P \neq^{?} \mathbf{B}}\ \Sigma\mathcal{HR}(Res)
$$

Clearly any $\Sigma\mathcal{HR}$-derivation using $\mathcal{D}$ can also use $\mathcal{D}'$.

## 6.4 Completeness of $\Sigma\mathcal{HR}$

We now investigate the relative completeness of $\Sigma\mathcal{HR}$ and use the unifying principle to show along the lines of [Hue72], that the class $\nabla_\Sigma(\Gamma) := \{\Phi \subseteq wsf_{\mathbb{O}}(\Sigma,\Gamma) \mid \Phi \not\vdash_{\Sigma\mathcal{HR}} \Box\}$ is an abstract consistency class.

**Lemma 6.4.1** *Let $\Phi$ be a set of $\Sigma$-clauses, $\Gamma$ and $\Xi$ annotated variable contexts such that $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{O}$, $\Xi \vdash_\Sigma \mathbf{B}::\mathbb{O}$, and $\mathbf{Dom}(\Gamma)\cap\mathbf{Dom}(\Xi) = \emptyset$. Furthermore, let $\mathcal{R}$ be a variable condition for $\Gamma$, and $\Upsilon$ one for $\Xi$. If $\Phi * \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A} \vdash_{\Sigma\mathcal{HR}} \langle\Delta\colon\mathcal{Q}\rangle.\mathbf{C}\vee\mathcal{E}_\sigma$, then $\Phi*\langle\Gamma,\Xi\colon\mathcal{R}\cup\Upsilon\rangle.\mathbf{A}\vee\mathbf{B} \vdash_{\Sigma\mathcal{HR}} \langle\Delta\colon\mathcal{Q}\rangle.\mathbf{C}$ or $\Phi*\langle\Gamma,\Xi\colon\mathcal{R}\cup\Upsilon\rangle.\mathbf{A}\vee\mathbf{B} \vdash_{\Sigma\mathcal{HR}} \langle\Delta,\Xi\colon\mathcal{Q}\cup\Upsilon\rangle.\mathbf{C}\vee\mathbf{B}$.*

**Proof:** Let $\mathcal{D}: \Phi * \langle\Gamma: \mathcal{R}\rangle.\mathbf{A} \vdash_{\Sigma\mathcal{HR}} \langle\Delta: \mathcal{Q}\rangle.\mathbf{C}$, then we prove the assertion by induction over $\mathcal{D}$. If $\mathcal{D}$ is empty, then $\langle\Delta: \mathcal{Q}\rangle.\mathbf{C} \in \Phi$ or $\langle\Delta: \mathcal{Q}\rangle.\mathbf{C} = \langle\Gamma: \mathcal{R}\rangle.\mathbf{A}$. In both cases we obtain the assertion with the empty $\Sigma\mathcal{HR}$-derivation.

If $\mathcal{D}$ ends in $\Sigma\mathcal{HR}(Res)$, then $\mathcal{D}$ is of the form

$$\dfrac{\dfrac{\Phi * \langle\Gamma: \mathcal{R}\rangle.\mathbf{A}}{\langle\Gamma': \mathcal{R}'\rangle.\mathbf{N}^\alpha \vee \mathbf{C}_1}\mathcal{D}' \quad \dfrac{\Phi * \langle\Gamma: \mathcal{R}\rangle.\mathbf{A}}{\langle\Gamma'': \mathcal{R}''\rangle.\mathbf{M}^\beta \vee \mathbf{C}_2}\mathcal{D}'' \quad \alpha \neq \beta}{\langle\Delta: \mathcal{Q}\rangle.\mathbf{C}_1 \vee \mathbf{C}_2 \vee \mathbf{M} \neq^? \mathbf{N}_1} \Sigma\mathcal{HR}(Res)$$

where $\Delta := \Gamma', \Gamma''$ and $\mathcal{Q} := \mathcal{R}' \cup \mathcal{R}''$. By inductive hypothesis we have $\Sigma\mathcal{HR}$-derivations

1. $\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B} \vdash \langle\Gamma': \mathcal{R}'\rangle.\mathbf{N}^\alpha \vee \mathbf{C}_1$ or $\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B} \vdash_{\Sigma\mathcal{HR}} \langle\Gamma', \Xi: \mathcal{R}' \cup \Upsilon\rangle.\mathbf{N}^\alpha \vee \mathbf{C}_1 \vee \mathbf{B}$
2. $\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B} \vdash \langle\Gamma'': \mathcal{R}''\rangle.\mathbf{M}^\beta \vee \mathbf{C}_2$ or $\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B} \vdash_{\Sigma\mathcal{HR}} \langle\Gamma'', \Xi: \mathcal{R}'' \cup \Upsilon\rangle.\mathbf{M}^\beta \vee \mathbf{C}_2 \vee \mathbf{B}$

Thus we have four cases, but as all of them can be treated with the same methods, we show the most complex one, where for both applications of the inductive hypothesis the latter alternative is assumed. In this case, we have

$$\dfrac{\dfrac{\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B}}{\langle\Gamma', \Xi: \mathcal{R}' \cup \Upsilon\rangle.\mathbf{N}^\alpha \vee \mathbf{C}_1 \vee \mathbf{B}}\mathcal{D}' \quad \dfrac{\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B}}{\langle\Gamma'', \Xi: \mathcal{R}'' \cup \Upsilon\rangle.\mathbf{M}^\beta \vee \mathbf{C}_2 \vee \mathbf{B}}\mathcal{D}'' \quad \alpha \neq \beta}{\langle\Delta, \Xi: \mathcal{Q} \cup \Upsilon\rangle.\mathbf{C}_1 \vee \mathbf{C}_2 \vee \mathbf{M} \neq^? \mathbf{N}_1 \vee \mathbf{B}} \Sigma\mathcal{HR}(Res)$$

The $\Sigma\mathcal{HR}(Fac)$ case is similar but much less complex, since we only need one application of the inductive hypothesis. In the $\Sigma\mathcal{HR}(Solv)$ case we have the following situation:

$$\dfrac{\dfrac{\Phi * \langle\Gamma: \mathcal{R}\rangle.\mathbf{A}}{\mathcal{C} = \langle\Gamma': \mathcal{R}'\rangle.\mathbf{C} \vee F \neq^? \mathbf{A}}\mathcal{D}'}{\langle\Delta: \mathcal{Q}\rangle.\mathbf{D} \vee F \neq^? \mathbf{A}} \Sigma\mathcal{HR}(Res)$$

where $\langle\Gamma': \mathcal{R}'\rangle.[\mathbf{A}/F]\mathbf{C} \vdash_{\mathcal{RC}} \langle\Delta: \mathcal{Q}\rangle.\mathbf{D}$ by a maximal $\mathcal{RC}$-derivation. By inductive hypothesis we have $\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B} \vdash \langle\Gamma': \mathcal{R}'\rangle.\mathbf{C} \vee F \neq^? \mathbf{A}$ or $\Phi * \langle\Gamma, \Xi: \mathcal{R} \cup \Upsilon\rangle.\mathbf{A} \vee \mathbf{B} \vdash_{\Sigma\mathcal{HR}} \langle\Gamma', \Xi: \mathcal{R}' \cup \Upsilon\rangle.\mathbf{C} \vee F \neq^? \mathbf{A} \vee \mathbf{B}$. Note that because of $F \in \mathbf{Dom}(\Gamma)$ we have $[\mathbf{A}/F]\mathbf{B} = \mathbf{B}$, therefore we obtain the assertion by a single application of $\Sigma\mathcal{HR}(Solv)$.

The remaining cases $\Sigma\mathcal{HR}(Prim)$, $\Sigma\mathcal{PT}(flex-rig)$, and $\Sigma\mathcal{PT}(guess)$ are nearly trivial, since they simply add a pair to $\mathcal{C}$, thus we obtain the assertion directly by the inductive hypothesis. □

**Lemma 6.4.2** *Let $\Phi$ and $\Psi$ be sets of sentences, and $\Phi \vee \Psi := \{\mathbf{A} \vee \mathbf{B} \mid \mathbf{A} \in \Phi, \mathbf{B} \in \Psi\}$. Furthermore, let $\Phi \vdash_{\Sigma\mathcal{HR}} \Box$ and $\Psi \vdash_{\Sigma\mathcal{HR}} \Box$, then $\Phi \vee \Psi \vdash_{\Sigma\mathcal{HR}} \Box$.*

**Proof:** Let $\mathbf{B} \in \Psi$, $\Phi = \{\mathbf{A}^i, \ldots, \mathbf{A}^n\}$, and $\Phi^i := \{\mathbf{A}^1 \vee \mathbf{B}, \ldots, \mathbf{A}^i \vee \mathbf{B}, \mathbf{A}^{i+1}, \ldots, \mathbf{A}^n\}$, then we first convince ourselves by induction on $i$, that $\Phi^i \vdash_{\Sigma\mathcal{HR}} \Box$ or $\Phi^i \vdash_{\Sigma\mathcal{HR}} \mathbf{C}$. If $i = 0$, then $\Phi = \Phi^0 \vdash_{\Sigma\mathcal{HR}} \Box$ by assumption. If $i > 0$, then $\Phi^{i-1} \vdash_{\Sigma\mathcal{HR}} \Box$ or $\Phi^{i-1} \vdash_{\Sigma\mathcal{HR}} \mathbf{B}$ by inductive hypothesis, thus we obtain the assertion by the previous lemma.

In particular, for $i = n$ we have $\mathcal{D}_\mathbf{B}\colon \Phi^n = \Phi \vee \{\mathbf{B}\} \vdash_{\Sigma\mathcal{HR}} \Box$ or $\mathcal{D}_\mathbf{B}\colon \Phi \vee \{\mathbf{B}\} \vdash \mathbf{B}$ for all $\mathbf{B} \in \Psi$. If the first case is assumed for some $\mathbf{B} \in \Psi$, then we have proven the assertion. Otherwise we have $\Phi \vee \Psi \vdash_{\Sigma\mathcal{HR}} \Psi$ by combining all $\Sigma\mathcal{HR}$-derivations $\mathcal{D}_\mathbf{B}$, which we can combine with the $\Sigma\mathcal{HR}$-refutation of $\Psi$ to obtain the assertion. □

Now we are in the position to attack the completeness proof for $\Sigma\mathcal{HR}$. We use the unifying principle for $\Sigma$-model structures, which we have proven in section 5.

**Theorem 6.4.3** *Let $\nabla_\Sigma(\Gamma) := \{\Phi \subseteq wsf_{\mathbb{O}}(\Sigma, \Gamma) \mid \Phi \nvdash_{\Sigma\mathcal{HR}} \Box\}$, then $\nabla_\Sigma(\Gamma)$ is a saturated abstract consistency class.*

**Proof:** To see that $\nabla_\Sigma$ is saturated, let $\Phi * \mathbf{A}^\mathtt{T} \vdash_{\Sigma\mathcal{HR}} \Box$ and $\Phi * \mathbf{A}^\mathtt{F} \vdash_{\Sigma\mathcal{HR}} \Box$. By the previous lemma 6.4.2 we can find a $\Sigma\mathcal{HR}$-refutation $\Phi * \mathbf{A}^\mathtt{T} \vee \mathbf{A}^\mathtt{F}$ and one of $\Phi$ by 6.3.7. We verify the properties of 5.4.3 for some set $\Phi \in \nabla_\Sigma(\Gamma)$ of sentences.

Let $\mathbf{A} \in \Phi$ be an atom such that $\mathbf{A}, \neg\mathbf{A} \in \Phi$, therefore we have $\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A}^\mathtt{T}, \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A}^\mathtt{F} \in \Omega$ for the corresponding unit $\Sigma$-clauses. So we obtain the assertion with the following $\Sigma\mathcal{HR}$-derivation

$$\dfrac{\dfrac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A}^\mathtt{T} \qquad \langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A}^\mathtt{F}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{A} \neq^? \mathbf{A}}\,\Sigma\mathcal{HR}(Res)}{\Box}\,\mathcal{SIM}$$

The remaining cases all share the following form: if $\mathbf{A} \in \Phi$, then $\Phi \cup \Psi \in \nabla_\Sigma(\Gamma')$ for some set of formulae $\Psi$. So we have to prove that, if $\mathbf{A} \in \Phi$ and $\Omega \nvdash \Box$, then $\Omega \cup \mathbf{CNF}(\Psi) \nvdash_{\Sigma\mathcal{HR}} \Box$, or equivalently, that the existence of a $\Sigma\mathcal{HR}$-refutation of $\Omega \cup \mathbf{CNF}(\Psi)$ guarantees a $\Sigma\mathcal{HR}$-refutation of $\Omega$. Condition 4 is a direct consequence of 6.4.2. For the cases 2, 3, and 5 the argumentation is trivial, since any $\Sigma\mathcal{HR}$-refutation of $\Phi * \mathbf{A}{\downarrow}$, $\Phi * \neg\neg\mathbf{A}$ or $\Phi * \neg(\mathbf{A} \vee \mathbf{B})$ is also one for $\Phi$, because the clause normal forms of $\mathbf{A}{\downarrow}$, $\neg\neg\mathbf{A}$ and that of $\mathbf{A}$ are identical, and finally, $\mathbf{CNF}(\neg(\mathbf{A} \vee \mathbf{B})) = \mathbf{CNF}(\{\neg\mathbf{A}, \neg\mathbf{B}\})$.

To verify 5.4.3(6) let $\Pi^{\mathbb{A}}\mathbf{A} \in \Phi$ and $\mathcal{D}_\theta$ be a $\Sigma\mathcal{HR}$-refutation of $\Psi_\theta := \Phi * \mathbf{AB}$. Let $\Phi = \Phi' * \Pi^{\mathbb{A}}\mathbf{A}$, then $\mathcal{D}_\theta$ is also a $\Sigma\mathcal{HR}$-refutation of $\Psi'_\theta := \Phi' \cup \{\mathbf{A}X^+, Pi^{\mathbb{A}}\mathbf{A}\}$, since the clause normal forms of $\Psi_\theta$ and $\Psi'_\theta$ are $\alpha$-variants (cf. 6.1.12). Now let $\Psi = \{\mathbf{A}X^+, \mathbf{A}Y^+\}$ and $\theta = [\mathbf{B}/Y]$, then $\theta(\Psi) = \Psi'_\theta$, so by 6.3.5 there is a $\Sigma\mathcal{HR}$-refutation $\mathcal{D}$ of $\Psi$. As the clause normal forms of $\mathbf{A}X^+$ and $\mathbf{A}Y^+$ are $\alpha$-variants, $\mathcal{D}$ is also a $\Sigma\mathcal{HR}$-refutation of $\Phi$.

For 5.4.3(7) let $\neg\Pi^{\mathbb{A}}\mathbf{A} \in \Phi$ and $\mathcal{D}$ be a $\Sigma\mathcal{HR}$-refutation of $\Phi * \neg\mathbf{A}X^-$ for some $X^- \notin \mathbf{Dom}(\Gamma)$. Note that the clause normal forms of $\Phi * \neg\mathbf{A}X^-$ and $\Phi$ are $\alpha$-variants, since those of $\neg\Pi^{\mathbb{A}}\mathbf{A}$ and $\neg\mathbf{A}X^-$ are, so $\mathcal{D}$ is also a $\Sigma\mathcal{HR}$-refutation of $\Phi * \mathbf{A}X^-$. □

Now the completeness theorem is only a simple corollary.

**Corollary 6.4.4 (Completeness of $\Sigma\mathcal{HR}$)** *Let $\Phi$ be a finite set of well-sorted sentences. If $\Phi$ is unsatisfiable in the class of $\Sigma$-model structures, then $\Phi \vdash_{\Sigma\mathcal{HR}} \Box$.*

**Proof:** Let $\nabla_\Sigma(\Gamma) := \{\Psi \subseteq wsf_{\mathbb{O}}(\Sigma,\Gamma) \mid \Psi \not\vdash_{\Sigma\mathcal{HR}} \Box\}$. If $\Phi$ were not $\Sigma\mathcal{HR}$-refutable, then $\Phi \in \nabla_\Sigma(\emptyset)$ by construction. Furthermore, by 6.4.3 $\nabla_\Sigma$ is a saturated abstract consistency class and therefore by 5.4.19 there is a $\Sigma$-model structure $\mathcal{M} \models \Phi$, which contradicts the assumption. $\Box$

**Theorem 6.4.5** *A well-sorted sentence* $\mathbf{A}$ *is valid in the class of* $\Sigma$*-model structures, iff* $\neg\mathbf{A}$ *is* $\Sigma\mathcal{HR}$*-refutable.*

**Proof:** The result is an immediate consequence of 6.4.4 and 6.2.5. $\Box$

**Theorem 6.4.6 (Relative Completeness of $\Sigma\mathcal{HR}$)** *Let* $\mathbf{A}$ *be a well-sorted* $\Sigma$*-sentence, then* $\vdash_{\Sigma\mathfrak{T}\eta} \mathbf{A}$*, iff* $\mathbf{A}^{\mathsf{F}} \vdash_{\Sigma\mathcal{HR}} \Box$*.*

**Proof:** By the previous theorem and 5.5.7. $\Box$

Now we can prove the statement about the completeness of our sorted Hilbert-Style calculi $\Sigma\mathfrak{T}*$, which we made above in section 5.

**Corollary 6.4.7** $\Sigma\mathfrak{T}$ *and* $\Sigma\mathfrak{T}\eta$ *are not complete with respect to general* $\Sigma$*-models.*

**Proof:** As we have seen in 6.2.3, there is a well-sorted formula $\mathbf{C}$ that is not $\Sigma\mathcal{HR}$-refutable. With the relative completeness theorem 6.4.6 above it is clear that $\neg\mathbf{C}$ is not derivable in $\Sigma\mathfrak{T}\eta$. Since the calculus $\Sigma\mathfrak{T}$ is weaker than $\Sigma\mathfrak{T}\eta$ (because it lacks the $\eta$-axiom) the result also holds for $\Sigma\mathfrak{T}$. $\Box$

We conclude our exposition of $\Sigma\mathcal{HR}$ with an example.

**Example 6.4.8** Let $\Sigma$ be the signature of 4.5.17 augmented by the sort $\mathbb{N}$ of natural numbers and the declarations

$$[0{::}\mathbf{N}], [s{::}\mathbb{N} \to \mathbb{N}], [+{:}\mathbb{N} \to \mathbb{N} \to \mathbb{N}],$$
$$[>{::}\mathbb{R} \to \mathbb{R} \to \mathbb{O}], [\mathrm{deg}{::}\mathbb{M} \to \mathbb{N}], [p{::}(\mathbb{R} \to \mathbb{R}) \to \mathbb{O}], [nc{::}(\mathbb{R} \to \mathbb{R}) \to \mathbb{O}]$$

We consider the following axiomatization of the degree of monomials

D1 $\forall Y_{\mathbb{R}}.\, \mathrm{deg}(\lambda X_{\mathbb{R}}.Y) = 0$

D2 $\mathrm{deg}(\lambda X_{\mathbb{R}}.X) = s(0)$

D3 $\forall F_{\mathbb{M}}, G_{\mathbb{M}}.\, \mathrm{deg}(\lambda X_{\mathbb{R}}.*(FX)(GX) = \mathrm{deg}(F) + \mathrm{deg}(F)$

and the following basic facts about $>$, non-constant ($nc$), and positive functions ($p$):

$>$ $\forall X_{\mathbb{N}}.x(X) > 0$

N $\forall F_{\mathbb{M}}.\, \mathrm{deg}(F) > 0 \Rightarrow nc(F)$

P $\forall F_{\mathbb{R}\to\mathbb{R}^+}.p(F)$

$+$ $\forall X_{\mathbb{N}}, Y_{\mathbb{N}}.(X > 0 \land Y > 0 \Rightarrow X + Y > 0$

From these we want to prove the assertion that there is a differentiable function that is positive, but not constant:

$$\exists F_{\mathbb{D}}.p(F) \wedge nc(F)$$

The clause normal form of its negation is

$$\mathrm{T}\ \langle [F{::}\mathbb{M}] {:}\, \emptyset\rangle.p(F)^{\mathsf{F}} \vee nc(F)^{\mathsf{F}}$$

from which we can obtain

R1=R(T,N) $\langle [F{::}\mathbb{M}] {:}\, \emptyset\rangle.p(F)^{\mathsf{F}} \vee (\deg(F) > 0)^{\mathsf{F}}$
R2=R(R1,P) $\langle [F{::}\mathbb{M}], [G{::}\mathbb{R} \to \mathbb{R}^{+} {:}\, \emptyset\rangle.(\deg(F) > 0)^{\mathsf{F}} \vee F \neq^{?} G$

by $\Sigma\mathcal{HR}(Res)$. Since all variable conditions in this example are empty, we drop the declarations from the clauses, and indicate the sort of variables by indices. With the $\Sigma\mathcal{PT}$-derivation (cf 4.6.12) of 4.5.17 and $\Sigma\mathcal{HR}(Solv)$ we obtain

R3=U(R2) $(\deg(F) > 0)^{\mathsf{F}} \vee F \neq^{?} (\lambda X_{\mathbb{R}}. * (H^5_{\mathbb{M}}X)(H^5_{\mathbb{M}}X))$
R4=S(R3) $(\deg(\lambda X_{\mathbb{R}}. * (H^5_{\mathbb{M}}X)(H^5_{\mathbb{M}}X)) > 0)^{\mathsf{F}}$

Since the clause normal form of $D3$ is

$$P_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}}. * (F_{\mathbb{M}}X)(G_{\mathbb{M}}X))^{\mathsf{T}} \vee P(\deg(F) + \deg(F))^{\mathsf{F}}$$

we have

R5=R(R4,G3) $P_{\mathbb{N}\to\mathbb{O}}(\deg(F_{\mathbb{M}}) + \deg(F_{\mathbb{M}}))^{\mathsf{F}} \vee$
$(\deg(\lambda X_{\mathbb{R}}. * (H^5_{\mathbb{M}}X)(H^5_{\mathbb{M}}X)) > 0)^{\mathsf{F}} \neq^{?} P(\deg(\lambda X_{\mathbb{R}}. * (F_{\mathbb{M}}X)(G_{\mathbb{M}}X))$

With the general bindings

$$\begin{array}{lcl} P_{\mathbb{N}\to\mathbb{O}} & =^{?} & \lambda X_{\mathbb{N}}. > (H^1_{\mathbb{N}\to\mathbb{N}}X)(H^2_{\mathbb{N}\to\mathbb{N}}X) \\ H^1_{\mathbb{N}\to\mathbb{N}} & =^{?} & (\lambda X_{\mathbb{N}}.X) \\ H^2_{\mathbb{N}\to\mathbb{N}} & =^{?} & (\lambda X_{\mathbb{N}}.0) \end{array}$$

and reduction to $\mathcal{SIM}$-normal form we obtain

R6=U(R5) $P_{\mathbb{N}\to\mathbb{O}}(\deg(F_{\mathbb{M}}) + \deg(F_{\mathbb{M}}) > 0)^{\mathsf{F}} \vee (H^5_{\mathbb{M}}X) =^{?} (F_{\mathbb{M}}X) \vee (H^5_{\mathbb{M}}X) =^{?} (G_{\mathbb{M}}X))$

which reduces to

R7=U(R6) $P_{\mathbb{N}\to\mathbb{O}}(\deg(F_{\mathbb{M}}) + \deg(F_{\mathbb{M}}) > 0)^{\mathsf{F}} \vee H^5_{\mathbb{M}} =^{?} F_{\mathbb{M}} \vee H^5_{\mathbb{M}} =^{?} G_{\mathbb{M}}$
R8=S(R7) $P_{\mathbb{N}\to\mathbb{O}}(\deg(H^5_{\mathbb{M}}) + \deg(H^5_{\mathbb{M}}))^{\mathsf{F}}$
R8=R(R7,+) $\deg(H^5_{\mathbb{M}}) > 0^{\mathsf{F}}$

On the other hand we have

R9=R(>,G2) $P_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}}.X))^{\mathsf{T}} \vee P_{\mathbb{N}\to\mathbb{O}}(s(0)) \neq^{?} S(X) > 0$

With $X =^{?} 0$ and general bindings for $P$ analogous to those above we obtain

R10=U(R9) $\deg(\lambda X_{\mathbb{R}}.X) > 0^{\mathsf{T}}$
R11=R(R10,R8) $H^5 = (\lambda X_{\mathbb{R}}.X)$

which is an empty clause, since the remaining pair is solved as the identity function is a monomial. □

# 7 Conclusion

## 7.1 Applications

This thesis has been motivated essentially by practical practical considerations, i.e. the shortcomings of first-order theorem proving and the lack of expressive power in higher-order logics due to the absence of sorts. However, as it turned out the content of this thesis is rather theoretical in nature. To bridge this gap and to see whether this theoretical system can fulfill its practical expectations, let us have a look at some applications of $\Sigma\mathcal{HOL}$.

We claim that introducing sorts to higher-order logic results in considerably more expressivity, and hence ultimately a practical language to express mathematical facts naturally. In fact, $\Sigma\mathcal{HOL}$ is the logical basis for the higher-order logic $\mathcal{POST}$ (**P**artial **O**rder-**S**orted **T**ype theory) of the $\Omega$-MKRP deduction system [HKK+92, HKK+94], currently under development at the Universität des Saarlandes. The goal of the $\Omega$-MKRP project is to develop an interactive proof development environment which can be used to prove the total contents of a typical mathematical textbook. The extensive experience of the $\Omega$-MKRP group, gained by axiomatizing mathematical theories for automated theorem provers, and the critique of existing input languages, which are considered too weak, have been a major motivation for the work reported in this thesis. The experiments with this new system have verified that sorted higher-order logics indeed offer a sufficiently rich language for naturally specifying a non-trivial fragment of mathematics. These experiments also show that, while term declarations are a desirable feature of a specification language for mathematics, pattern signatures have so far been sufficient for all practical applications. Thus the problems with decidability of sort computation (see the discussion in 4.3) are mainly a theoretical concern.

The resolution calculus $\Sigma\mathcal{HR}$ finds its concretization in the $\mathcal{LEO}$ (**L**ogic **E**ngine for $\Omega$-MKRP) theorem prover also currently under development in Saarbrücken. This implementation is currently been used to test the practical applicability of our calculus and for the development of search strategies specialized to higher-order logic. Naturally unification in $\Sigma\mathcal{HOL}$ is considerably more complex than in the simply typed $\lambda$-calculus, but this complexity is more than compensated by the restriction of search spaces associated with resolution theorem proving.

These applications in higher-order deduction for mathematics are not the only conceivable ones. For instance, $\Sigma\mathcal{HOL}$ can be seen as a logical basis for sorted logic programming languages, such as extensions to $\lambda$-Prolog [MN87, Mil89] in the spirit of TEL [Smo89] or Godel [HL94]. Because of undecidability issues, term declarations would have to be severely restricted, e.g. to higher-order patterns, in order to make sort computation decidable. Even with this restriction $\Sigma$-unification is still undecidable and infinitary (because the first-order subcase is), but it seems probable that there are decidable subcases analogous to those for first-order systems (cf. [Soc93, Uri92]).

Finally, our work can be seen as a guide for adding sort information to other typed $\lambda$-calculi. In this respect our work has applications in the field of meta-logical frameworks ($\lambda$-calculi with strong type systems that are used to formalize logical systems), since the added expressivity makes practical formalizations of logic systems much more convenient. It is still unclear, whether our focus on an extensional partial function semantics is advantageous for logical frameworks. The features of $\Sigma\mathcal{HOL}$ like functional base sorts or term declarations can, however, be adapted to existing sort systems for logical frameworks [Pfe93, KP93] as well.

## 7.2 Sorted Logics: An A-Posteriori View

To get a better intuition of the improvements that $\Sigma\mathcal{HOL}$ has to offer for specifying mathematical theorems and proving them, let us have a closer look at the alternative of using an unsorted higher-order logic. With the relativization technique well-known from first-order logic a well-sorted formulae $\mathbf{A}$ can be coded into an unsorted formula $\mathbf{Rel}(\mathbf{A})$. For instance, a sorted quantification $\forall X_{\mathbb{A}}.\mathbf{A}$ is transformed into $\forall X_{\tau(\mathbb{A})}.(\mathbf{P}_{\mathbb{A}}X) \Rightarrow \mathbf{Rel}(\mathbf{A})$ for some new predicate $\mathbf{P}_{\mathbb{A}}$ that captures the sort information. These new predicates obtain their meaning from an axiomatization $\mathbf{Rel}(\Sigma)$ of the sort information present in a valid signature $\Sigma$, which is provided by the relativization operation. For most sorted first-order logics it is a theorem (see [Wal87, SS89, Wei91] for examples) that a well-sorted formula $\mathbf{A}$ is satisfiable in the class of sorted models, iff $\mathbf{Rel}(\mathbf{A})$ is satisfiable in the class of unsorted models that satisfy the signature axioms $\mathbf{Rel}(\Sigma)$. Thus from a theoretical point of view these sorted first-order logics are not more expressive than unsorted ones. Indeed we conjecture that some kind of sort theorem also holds for $\Sigma\mathcal{HOL}$. This would entail that in theory all theorems of $\Sigma\mathcal{HOL}$ can be proven by coding them into simple type theory and then proving them by simply typed constrained resolution.

Unfortunately, in $\Sigma\mathcal{HOL}$ the situation is not as simple as in the first-order case, where the only binding constructs are quantifications that can be transformed as shown above. In the presence of $\lambda$-abstractions we need some form of conditionals in the target system for coding functional formulae, such as $\lambda_{\mathbb{A}}.\mathbf{A}$, which have to be relativized as partial functions. Conditionals can be realized by description functions (see 5.2.13) as in [Chu40]. Such a system can be obtained from $\Sigma\mathcal{HOL}$ by adding a logical constant $\iota_{(\alpha\to o)\to\alpha}$ for each $\alpha \in \mathcal{T}$ and the inference rule

$$\frac{\Gamma \vdash_\Sigma \mathbf{Q}{::}\mathbb{A} \to \mathbb{O} \quad \Gamma \Vdash_\Sigma \forall X_{\mathbb{A}}.\mathbf{Q}X \Rightarrow .\forall Y_{\mathbb{A}}(\mathbf{Q}Y \Rightarrow X = Y)}{\mathbf{Q}(\iota\mathbf{Q})}\ \Sigma\mathfrak{T}(\iota)$$

to $\Sigma\mathfrak{TE}$. Moreover, in the definition of general $\Sigma$-models we have to specify that the value of $\iota$ is the function that maps singleton sets to their unique member. In this setting we can define a conditional $\mathbf{w}_{\alpha o\alpha}$ as implication, if $\alpha = o$, and otherwise as $(\lambda X_\alpha, P_o.\iota_{\alpha(o\alpha)}.\lambda Y_\alpha.P \wedge Y = X)$. It is easy to see that for any general $\Sigma$-model $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ we have $\mathcal{I}_\varphi(\mathbf{w})(a, \mathtt{T}) = a$.

With these extensions we can now define a relativization operator $\mathbf{Rel}$, and use it to compare the relativization of an example with the sorted version. This comparison will give us a feeling for the advantages of sorts in higher-order deduction.

**Definition 7.2.1 (Relativization)** $\mathbf{Rel}$ is a typed mapping $\mathbf{Rel}{:}\, \mathcal{S} \to \mathit{wff}(\overline{\Sigma} \cup \mathbf{P}_{\mathcal{S}})$, where $\mathbf{P}_{\mathcal{S}} := \{\mathbf{P}_{\mathbb{A}} \mid \mathbb{A} \in \mathcal{S}\}$ is a set of new predicates of type $\tau(\mathbb{A}) \to o$. We now inductively define $\mathbf{Rel}$ by setting

1. $\mathbf{Rel}(\mathbb{A}) := \mathbf{P}_{\mathbb{A}}$ for $\mathbb{A} \in \mathcal{S}^{nf} \cap \mathcal{BS}$,

2. $\mathbf{Rel}(\mathbb{A} \to \mathbb{B}) := (\lambda F_{\tau(\mathbb{A})\to\tau(\mathbb{B})}.\forall X_{\tau(\mathbb{A})}.(\mathbf{Rel}(\mathbb{A})X) \Rightarrow .\mathbf{Rel}(\mathbb{B})(FX))$,

3. $\mathbf{Rel}(\mathbb{A}) := \lambda X_{\tau(\mathbb{A})}.\mathbf{P}_{\mathbb{A}} \wedge \mathbf{Rel}(\mathfrak{d}(\mathbb{A}) \rightarrow \mathfrak{r}(\mathbb{A}))X$ for $\mathbb{A} \in \mathcal{S}_0^f$,

This relativization of sorts allows us to define the full relativization operator on well-sorted formulae, which is a typed function $\mathbf{Rel}$: $wsf(\Sigma, \Gamma) \longrightarrow wff(\overline{\Sigma} \cup \mathbf{P}_{\mathcal{S}})$ such that

4. $\mathbf{Rel}(\mathbf{A}) := \mathbf{A}$, if $\mathbf{A}$ is a constant or variable,

5. $\mathbf{Rel}(\mathbf{AB}) := \mathbf{Rel}(\mathbf{A})\mathbf{Rel}(\mathbf{B})$,

6. $\mathbf{Rel}(\lambda X_{\mathbb{B}}.\mathbf{A}) := (\lambda \overline{X}.\mathbf{w}\mathbf{Rel}(\mathbb{B})\mathbf{A})$.

Finally, we can map valid signatures $\Sigma$ into sets $\mathbf{Rel}(\Sigma)$ of sentences, called **signature axioms**, by defining

7. $\mathbf{Rel}(\Sigma, [\forall\Gamma\mathbf{A}{::}\mathbb{A}]) := \mathbf{Rel}(\Sigma) \wedge \mathbf{Rel}([\forall\Gamma.\mathbf{A}{::}\mathbb{A}])$ where

8. $$\begin{aligned}\mathbf{Rel}([\forall\Gamma.\mathbf{A}{::}\mathbb{A}]) := \forall X^1_{\tau(\mathbb{A}_1)}, \ldots X^n_{\tau(\mathbb{A}_n)}.\quad & \mathbf{Rel}(\mathbb{A}_1)X^1 \Rightarrow \ldots \Rightarrow .\mathbf{Rel}(\mathbb{A}_n)X^n \\ & \Rightarrow \mathbf{Rel}(\mathbb{A})\mathbf{Rel}(\mathbf{A})\end{aligned}$$

   and $\Gamma = [X^1{::}\mathbb{A}^1], \ldots, [X^n{::}\mathbb{A}^n]$.

For instance, in the case of a subsort declaration $[\mathbb{A} \leq \mathbb{B}] \in \Sigma$ we have

$$\begin{aligned}\mathbf{Rel}([\mathbb{A} \leq \mathbb{B}]) &= \mathbf{Rel}([\forall[X{::}\mathbb{A}].X{::}\mathbb{B}]) \\ &= \forall X_{\tau(\mathbb{A})}.(\mathbf{Rel}(\mathbb{A})X) \Rightarrow .\mathbf{Rel}(\mathbb{B})X\end{aligned}$$

as we would have expected from the analogy with first-order sorted logics. In particular, if $\mathbb{A}$ and $\mathbb{B}$ are non-functional sorts, then the sentence $\forall X.(\mathbf{P}_{\mathbb{A}}X) \Rightarrow .\mathbf{P}_{\mathbb{B}}X$ just amounts to the subset definition. To enhance our intuition on relativizations of signatures, let us compare the sorted formulation of the signature in example 4.5.17 with its relativization in the following signature axioms:

$$\begin{aligned}\mathbf{Rel}([\partial{::}\mathbb{D} \rightarrow \mathbb{C}]) &= \forall F.(\mathbf{P}_{\mathbb{D}}F \wedge \forall Y.\mathbf{P}_{\mathbb{R}}Y \Rightarrow \mathbf{P}_{\mathbb{R}}(FY)) \Rightarrow .\mathbf{P}_{\mathbb{C}}(\partial F) \\ &\qquad \wedge(\forall Y.\mathbf{P}_{\mathbb{R}}Y \Rightarrow .\mathbf{P}_{\mathbb{R}}(\partial F)Y) \\ \mathbf{Rel}([\partial{::}\mathbb{P} \rightarrow \mathbb{P}]) &= \forall F.(\mathbf{P}_{\mathbb{P}}F \wedge \forall Y.\mathbf{P}_{\mathbb{R}}Y \Rightarrow \mathbf{P}_{\mathbb{R}}(FY))) \Rightarrow .(\mathbf{P}_{\mathbb{P}}(\partial F)) \\ &\qquad \wedge(\forall Y.\mathbf{P}_{\mathbb{R}}Y \Rightarrow .\mathbf{P}_{\mathbb{R}}(\partial F)Y) \\ \mathbf{Rel}([\lambda X_{\mathbb{R}}.X{::}\mathbb{P}]) &= \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}}XX)) \wedge \forall Y.(\mathbf{P}_{\mathbb{R}}Y) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}}X)Y)) \\ \mathbf{Rel}([\forall[Z_{\mathbb{R}}].\lambda X_{\mathbb{R}}.Z{::}\mathbb{P}]) &= \forall Z.(\mathbf{P}_{\mathbb{R}}Z \Rightarrow \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}}XZ))\forall Y.(\mathbf{P}_{\mathbb{R}}Y)) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}}X)Z)))\end{aligned}$$

$$\begin{aligned}&\mathbf{Rel}([\forall[F{::}\mathbb{P}][G{::}\mathbb{P}].\lambda X_{\mathbb{R}}. + (FY)(GY){::}\mathbb{P}]) \\ &\quad = \forall F, G.(\mathbf{P}_{\mathbb{P}}F) \Rightarrow .(\mathbf{P}_{\mathbb{P}}G) \Rightarrow \\ &\qquad \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}}X. + (FX)(GX)))\forall Y.(\mathbf{P}_{\mathbb{R}}Y) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}}Y) + (FY)(GY))) \\ &\mathbf{Rel}([\forall[F_{\mathbb{P}}][G_{\mathbb{P}}].\lambda X_{\mathbb{R}}. * (FY)(GY){::}\mathbb{P}]) \\ &\quad = \forall F, G.(\mathbf{P}_{\mathbb{P}}F) \Rightarrow .(\mathbf{P}_{\mathbb{P}}G) \Rightarrow \\ &\qquad \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}}X. * (FX)(GX)))\forall Y.(\mathbf{P}_{\mathbb{R}}Y) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}}Y) * (FY)(GY)))\end{aligned}$$

This set of axioms has to be added to the relativization of any $\Sigma\mathcal{HOL}$-sentence that we want to prove in the relativized form. A further effect, which we have not illustrated for lack of space, is that the unification in the sorted setting finds out conflicting taxonomic information for proof objects, and prevent any inference that would yield ill-sorted objects.

These objects arise naturally in the relativized setting, but due to $\mathbf{Rel}(\Sigma)$ they can never contribute to any proof. Thus resolution in $\Sigma\mathcal{HOL}$ gives us the further advantage of cutting off enormous redundant branches in the search space. As a consequence the search spaces are so much smaller that the sorted calculus is clearly practically superior.

In an a posteriori view we can see the generalization of resolution to a sorted setting as the process of building certain classes of axioms, namely, those that correspond to term declarations, into the unification. This process takes axioms of the form $(\forall \overline{X^k}.p^1X^1 \Rightarrow \ldots \Rightarrow p^kX^k \Rightarrow (q\mathbf{A}))$ where the $p^i$ and $q$ are unary predicate constants and $X^i$ are the free variables of $\mathbf{A}$ out of the search, and treats them algorithmically in the unification.

## 7.3 Further Work

Naturally we have not solved all of the problems of sorted $\lambda$-calculi and sorted higher-order logics. On the contrary the investigation of this topic has just started. We now point out some problems left open by our work, and indicate some directions of future research.

### Syntactically Restricted Classes of Formulae

For certain practical applications it is important to find syntactically restricted classes of formulae and signatures that enjoy more tractable unification and sort computation problems. One of the top candidates would be an appropriate generalization of higher-order patterns [Mil92]. In $\Lambda$ this class of syntactically restricted formulae has a decidable unification problem. Miller used higher-order patterns as a basis for the logic programming language $\lambda$-PROLOG [NM88, Mil91], Nipkow in [Nip91] for higher-order rewriting, and Pfenning adapted the results to his logic programming language ELF [Pfe91]. The use of sorted logics in logical frameworks [Pfe93] has lead Pfenning and the author to develop a pattern unification algorithm for a sorted $\lambda$-calculus [KP93].

Unfortunately, we cannot hope that $\Sigma$-unification for is unitary or decidable, since this is not the case for the first-order case [SS89]. Furthermore, the naive generalization of the pattern techniques for the flex/flex case, which call for inference rules like the following will not work.

$$\frac{\langle\Gamma, [F{::}\mathbb{A}]{:}\,\mathcal{R}\rangle.FX^{\varphi(1)}\ldots X^{\varphi(n)} =^? FX^{\psi(1)}\ldots X^{\psi(n)} \wedge \mathcal{E}}{\langle\Gamma, [F^+{::}\mathbb{A}], [H^+{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{r}^l(\mathbb{A})]{:}\,\mathcal{R}\rangle.F =^? \lambda Y^1_{\mathbb{B}^{\varphi(1)}}\ldots X^n_{\mathbb{B}^{\varphi(n)}}.HY^{\rho(1)}\ldots Y^{\rho(l)} \wedge \mathcal{E}}\ \Sigma\mathcal{UP}(same)$$

Here $\rho$ is a partial permutation satisfying: there exists a $k$ such that $\rho(k) = \varphi(i)$ iff $\varphi(i) = \psi(i)$.

$$\frac{\langle\Gamma, [F^+{::}\mathbb{A}]{:}\,\mathcal{R}\rangle.F^+X^{\varphi(1)}\ldots X^{\varphi(n)} =^? GX^{\psi(1)}\ldots X^{\psi(m)} \wedge \mathcal{E}}{\langle\Gamma, [F^+{::}\mathbb{A}], [H^+{::}\mathbb{B}], \overline{[X_n^-{::}\mathbb{B}^n]}{:}\,\mathcal{R}\rangle.F =^? \mathbf{F} \wedge G =^? \mathbf{G}}\ \Sigma\mathcal{UP}(diff)$$

where

1. $\mathbf{F} := \lambda Y^1_{\mathbb{B}^{\varphi(1)}}\ldots X^n_{\mathbb{B}^{\varphi(n)}}.HY^{\varphi'(1)}\ldots Y^{\varphi'(l)}$,

2. $\mathbf{G} := \lambda Y^1_{\mathbb{B}^{\psi(1)}} \ldots X^m_{\mathbb{B}^{\psi(m)}}.HY^{\psi'(1)} \ldots Y^{\psi'(l)}$,

3. $\varphi'$ and $\psi'$ are partial permutations satisfying: there exists a $k$ such that $\varphi'(k) = i$ and $\psi'(k) = j$ iff $\varphi(i) = \psi(i)$,

4. $\Gamma(X^{\varphi(i)}) = \mathfrak{d}^i(\mathbb{A}) \quad \Gamma(X^{\psi(i)}) = \mathfrak{d}^i(\mathbb{A})$.

These inference rules cannot work, since if $\mathbf{ln}(\mathbb{A}) < n$, then we cannot guarantee that the general binding $\mathbf{F}$ really has sort $\mathbb{A}$.

There are syntactic restrictions on the term declarations in the first-order case that make first-order $\Sigma$-unification decidable [Uri92, Soc93]. So there is hope that suitable generalizations of these restrictions to $\Sigma$-patterns yield decidable $\Sigma$-pattern unification problems. For instance, if the signature is **elementary**, i.e. all term declarations contain at most one occurrence of a constant symbol, the sort constraint techniques of [KP93] can be used to obtain a decidable subcase.

### Dynamic Sorts and Partial Functions

In first-order predicate logic the introduction of term declarations has been a major step in the development of dynamic sorted logics [WO90, Wei91, Wei93], where variables are restricted to sorts, but where the sorts can also be treated as unary predicates in the logic allowing the specification of conditioned term declarations; thus the signature is no longer fixed during the search, as sort information can appear in the deduction process. The resolution rule always uses sorted unification with respect to the signature specified by the current state of the proof. Since predicates are primary objects of type theory, a generalization of the resolution system in [Wei91] may yield very powerful calculi for mechanizing mathematics and, in particular, for analysis.

Recently Weidenbach's results have been applied by Kerber in collaboration with the author to obtain an efficient mechanization of Kleene's three-valued approach for partial functions [KK93, KK94]. We believe that this result can be generalized to higher-order logic, and leads to a very natural and powerful logic system for mechanizing informal mathematical practice. Our resolution calculus $\Sigma\mathcal{HR}$ and especially our $\Sigma$-unification algorithms with term declarations are an important foundation for the generalization of these resolution calculi with dynamic sorts to higher-order logic. Thus the work reported here is one key ingredient of $\mathcal{POST}$, a higher-order logic with sorts and partial functions along the lines of our first-order formalization mentioned above. In this direction the work of Farmer [Far93, FGT93] in LUTINS and IMPS has shown that partial functions are a very natural and powerful tool for formalizing mathematics. The author expects that our three-valued approach, which remedies some problems of the simpler two-valued approach, will give an even more powerful framework for deduction systems for mathematics, since the three-valued logic rejects sentences that most mathematicians would deem false whereas LUTINS accepts them as theorems.

### Relativization

The relativization technique indicated in section 7.1 has to be formalized, and the sort theorems in the spirit of [Wal87, SS89, Wei91]) have to be proven. Sort theorems may be more meaningful and natural in extensions of $\Sigma\mathcal{HOL}$ with description functions. In [Far93]

Farmer claims that $\Sigma\Lambda$ can be directly encoded into his system LUTINS that takes the notion of partial functions as primitive [Far90, Far91b] objects. This claim is natural, since sorted logics in some sense formalize the "well-behaved" part of partial functions. On similar grounds the relativization into a higher-order generalization of the three-valued Kleene logic [KK94], which we have discussed above, would be interesting. In fact, these logical system is probably an even more natural target system for relativization than unsorted higher-order logic with description functions. It would be interesting to formalize these encodings, and use them for a comparison between the two-valued and the Kleene approach to partial functions.

Another, perhaps more practical, application of the relativization technique would be to provide the user of a deduction system with a very expressive sort mechanism for specifying mathematics, but then rely on relativization techniques to code this into less expressive sorted logics that have better computational properties. In particular, we think of restrictions of the signatures as discussed above. For such applications it would be fruitful to consider Stickel's technique of term relativization [Sti86].

### Cut Elimination in Extensional Higher-Order Logic

In [And71] Andrews has given a simple cut-elimination proof for a system $\mathcal{G}^+$ of higher-order logic without extensionality by showing that both the system $\mathcal{G}^+$ with cut and the cut-free system $\mathcal{G}$ are complete relative to $\mathfrak{T}$. We conjecture that along these lines it should be easy to construct a cut elimination proof for simple type theory with extensionality. In particular, the method above would lead to a proof of cut-elimination in a formulation of type theory with function symbols. The author only knows of proofs in formulations of classical higher-order logic without function symbols (cf. [Tak87, Tak68, Tak70]). There is a cut-elimination for intuitionistic type theory with extensionality and function symbols in [Autar]. Note that the results in [And71] are abstractions of the cut-elimination proof for simple type theory in [Tak67], which was extended to the extensional case in [Tak68]. Therefore we believe that the unifying principle for general $\Sigma$-models can be used correspondingly.

### Resolution for Extensional Higher-Order Logic

As we have seen in example 6.2.3 $\Sigma\mathcal{HR}$ is not complete with respect to general $\Sigma$-models, since they are fully extensional (5.2.17), and $\Sigma\mathcal{HR}$ cannot cope with the axiom of truth values 5.2.14. This is unfortunate, since this class of models is the most intuitive one that admits complete calculi. In particular, our mathematical intuition would make us believe that a clause set like $\mathcal{C}\{\langle\emptyset\colon\emptyset\rangle.(cb)^{\mathsf{F}}, \langle\emptyset\colon\emptyset\rangle.c(\neg\neg b)^{\mathsf{T}}$ should be refutable, because $\neg\neg b$ is provably equivalent to $b$. This example shows us that in extensional calculi we have to deal with propositions that appear in the arguments of function constants. The simplest approach to build a calculus that can refute $\mathcal{C}$ is to add the equational theory $b = \neg\neg b$ to higher-order unification. Even though this approach is intuitive, it does not solve the general problem of incorporating extensionality into resolution. In fact, we can generalize the formula $\mathbf{C} := (cb) \vee \neg c(\neg\neg b)$ to $\mathbf{C}' := (c\mathbf{A}) \vee \neg(c\mathbf{B})$, where $\mathbf{A}$ and $\mathbf{B}$ are arbitrary propositions. Now $\mathbf{C}'$ is valid in the class of general $\Sigma$-models, iff $\mathbf{A} \Leftrightarrow \mathbf{B}$ is valid. So the approach of enhancing the unification would require augmenting the unification procedure by the theory of logical equivalence, which would enable the unification procedure to prove any theorem by unifying it with $\top_o$.

To make these ideas more precise let us digress to a more general look at automatic theorem proving. Theorem proving is a syntactic process of making judgments about the validity of formulae in all models.

In propositional logic formulae are built up from propositional variables, and the logical connectives $\neg$ and $\vee$. While the variables can be arbitrarily interpreted (to be either $\mathtt{T}$ or $\mathtt{F}$), the connectives $\neg$ and $\vee$ are interpreted to denote the negation and disjunction functions on the set of truth values. Thus the class of models consists only of the $\{\neg, \vee\}$-algebra with carrier set $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$ where $\mathcal{I}(\neg)$ and $\mathcal{I}(\vee)$ are the well-known functions.

In first-order logic there is a clear conceptual distinction between terms (syntactic objects that denote individuals) and formulae (syntactic objects that denote truth values). Formulae are built up from atoms, the symbols $\neg$ and $\vee$, and quantification. Atoms take the place of propositional variables, whereas $\neg$, $\vee$, and quantification have fixed interpretations. Atoms are built up from predicate symbols and terms, which in turn are built up from function symbols, individual constants, and variables, all of which can be freely interpreted. Thus the class of models for first-order logic consists of some universe $\mathcal{D}_\iota$ of individuals and $\mathcal{D}_o$ with a fixed interpretation for $\neg$, $\vee$, and quantification.

Skolemization eliminates the treatment of quantification into a preprocess in refutation-based theorem proving. For instance, resolution-based calculi consist of the propositional rules (computation in the fixed part $\mathcal{D}_o$) and the unification procedure, which amounts to solving term equations in all models. Since the term algebra is the free algebra, it is sufficient to solve the term equations there.

Let us summarize these ideas. Due to the strong division of the model theory into a fixed part $\mathcal{D}_o$ and a free part $\mathcal{D}_\iota$, first-order theorem proving can be divided into a propositional part (acting on formulae) and a term part (unification), which do not interfere.

In higher-order logic (here simple type theory) we do not have this clear division. In particular, there are formulae, where symbols with a fixed interpretation are dominated (in the scope or subterms of arguments) by symbols with a flexible interpretation.

We propose a calculus where the unification procedure calls the theorem proving procedure recursively on demand, i.e. whenever it encounters a propositional pair. This approach makes it necessary to break down the distinction between unification and resolution. It should treat both processes in one uniform calculus.

To account for extensionality we propose the following two rules:

$$\frac{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{C} \vee \mathbf{A} \neq^? \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{O} \quad \Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{O}}{\langle\Gamma\colon\mathcal{R}\rangle.\mathbf{CNF}(\neg(\mathbf{A} \Leftrightarrow \mathbf{B})) \vee \mathbf{C}}\ \mathcal{ER}(Ref)$$

$$\frac{\langle\Gamma, [P{::}(\mathbb{A} \to \mathbb{B}) \to \mathbb{O}]\colon\mathcal{R}\rangle.(P\mathbf{A})^{\mathsf{F}} \vee (P\mathbf{B})^{\mathsf{F}} \vee \mathbf{C}}{\langle\Gamma, [Q{::}\mathbb{A} \to \mathbb{O}], [X^-{::}\mathbb{B}]\colon\mathcal{R}\rangle.(Q(\mathbf{A}X))^{\mathsf{F}} \vee (Q(\mathbf{B}X))^{\mathsf{F}} \vee \mathbf{C}}\ \mathcal{ER}(Ext)$$

Obviously the first rule amounts to the recursive call of the refutation procedure. In our example above we have the following $\Sigma\mathcal{HR}$-derivation $\mathcal{A}$

$$\dfrac{\dfrac{\dfrac{\langle\emptyset\colon\emptyset\rangle.(cb)^{\mathtt{F}} \qquad \langle\emptyset\colon\emptyset\rangle.c(\neg\neg b)^{\mathtt{T}}}{\langle\emptyset\colon\emptyset\rangle.c(\neg\neg b) \neq^{?} (cb)}\,\Sigma\mathcal{HR}(Res)}{\langle\emptyset\colon\emptyset\rangle.(\neg\neg b) \neq^{?} b}\,\mathcal{SIM}}{\langle\emptyset\colon\emptyset\rangle.b}\,\mathcal{ER}(Ref)$$

similarly we have a $\Sigma\mathcal{HR}$-derivation $\mathcal{A}'$ of $\langle\emptyset\colon\emptyset\rangle.b$, since the clause normal form of $\neg(\mathbf{A} \Leftrightarrow \mathbf{B})$ is $\{\langle\emptyset\colon\emptyset\rangle.b^{\mathtt{T}}, \langle\emptyset\colon\emptyset\rangle.b^{\mathtt{F}}\}$. Thus we can complete the refutation with

$$\dfrac{\dfrac{\dfrac{\mathcal{A}}{\langle\emptyset\colon\emptyset\rangle.b} \qquad \dfrac{\mathcal{A}'}{\langle\emptyset\colon\emptyset\rangle.\neg b}}{\langle\emptyset\colon\emptyset\rangle.b \neq^{?} b}\,\Sigma\mathcal{HR}(Res)}{\Box}\,\mathcal{SIM}$$

Even though we do not have a completeness proof, we are confident that the proposed calculus will at least solve the problem of two-valuedness.

### Equality and Higher-Order *RUE*-Resolution

If we consider the inference rules of an unsorted version of $\Sigma\mathcal{HR}$, then we see that they are direct generalizations of the classical *RUE*-resolution calculus of Digricoli [Dig79, Dig81], which also mixes unification with proof search. In particular, if we exchange our symbol $=^{?}$ in $\Sigma$-unification problems for the equality constant $=^{\mathbb{A}}$, then the pairs in unification constraints become proper equality literals, and can be resolved upon like the *RUE*-resolution calculus advocates. Our completeness result for $\Sigma\mathcal{HR}$ (6.4.4) can then be read as partial completeness result for a higher-order *RUE*-resolution calculus for input formulae without equality. It would be interesting to extend this result to full higher-order logic with equality.

We only use this little observation as an example for the real problem of finding specialized mechanizations of higher-order equality. Finding efficient methods for equality will be one of the most critical single problems remaining to be solved in order to make higher-order deduction practical.

# 8 Acknowledgments

I am indebted to my advisor Jörg Siekmann for introducing me to the field of artificial intelligence and giving me a feeling for the big picture of our field. I still feel honored that he accepted me, a total newcomer to artificial intelligence and logic, as a student. I was very lucky that he focused my interest to the fascinating subject of higher-order deduction. His research group at the Universität des Saarlandes has provided the friendly, supportive, and extremely stimulating environment that is necessary for successful research.

During my visit to Carnegie Mellon University, Peter Andrews and Frank Pfenning have answered a great many questions, and have always had an open ear for my ideas; I would like to thank them for their patience and support.

Frank Pfenning has become a second advisor to me, he has even taken time off from his vacation in Germany, when I was hopelessly stuck in the proof of the structure theorem, and pointed me to the technique of logical relations, which finally solved the problem.

The three main components of this thesis can be traced to the influence of these three people: Jörg Siekmann has introduced me to first-order resolution theorem proving and the dream of mechanizing mathematics. Furthermore, he has conveyed the practical importance of sorted logics, that have been significantly influenced by researchers under his supervision. Peter Andrews had an important role in shaping my view of higher-order logic and deduction, while Frank Pfenning introduced me to type theory, and helped me to aquire most of the background that was necessary to complete this thesis.

I would also like to explicitly thank my colleagues Franz Baader, Manfred Kerber, Daniel Nesmith, Christian Prehofer, Zhenyu Qian, Christoph Weidenbach, Ahmet Bozkurt, Gerald Klein, and Ortwin Scheja for valuable discussions and proofreading of earlier drafts of this thesis. Dr. Patricia Johann has read and discussed preliminary versions of this thesis in depth and has spent a lot of time finding formal errors and discussing them with me. Last but not least I want to thank Andrea Kohlhase, who has given me the moral support that is necessary to survive a substantial research project and who has taken up most of the burden of proofreading. More valuable even, she has always insisted that I try harder, when explanations were still unclear.

Finally, the "Studienstiftung des Deutschen Volkes" has had a twofold influence on the work reported in this thesis. At one of their summer academies in Campill, Italy, I had my very first positive experience with the field of artificial intelligence. This course, given by Jörg Siekmann and Wolfgang Wahlster, later brought about my decision to change my research field from mathematics to deduction systems. Much later the Studienstiftung provided the research grant that made it possible to visit the Carnegie Mellon University for a semester without a lot of bureaucratic overhead.

# References


[Aie90] Luigia Carlucci Aiello, editor. *Proceedings of of the 9th European Conference on Artifical Intelligence*, Stockholm, Sweden, 1990. Pitman Publishing, London, England.

[AINP90] Peter B. Andrews, Sunil Issar, Dan Nesmith, and Frank Pfenning. The TPS theorem proving system. In *[Sti90]*, 1990.

[ALCMP84] Peter B. Andrews, Eve Longini-Cohen, Dale Miller, and Frank Pfenning. Automating higher order logics. *Contemp. Math*, 29:169–192, 1984.

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 3(36):414–432, 1971.

[And72] Peter B. Andrews. General models and extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.

[And73] Peter B. Andrews, 1973. letter to Roger Hindley dated January 22, 1973.

[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.

[Autar] Unknown Author. *Unknown Title.* PhD thesis, Worcester College, Unknown Year. I have a manuscript of the thesis (the author is a student advised by Professor Gandy) without a title page.).

[Bar80] Hendrik P. Barendregt. *The Lambda-Calculus: Its Syntax and Semantics.* North-Holland, 1980.

[Bax78] L. D. Baxter. The undecidability of the third order dyadic unification problem. *Information and Control*, 38(2), 1978.

[BEG+64] J. H. Bennet, W. B. Easton, J. R. Guard, D. B. Loveman, and T. H. Mott. Semi-automated mathematics: SAM IV. Scientific Report 64–827, Air Force Cambridge Research Laboratories, October 1964.

[Ber41] Paul Bernays. A system of axiomatic set-theory. *Journal of Symbolic Logic*, 6:1–17, 1941.

[BG90] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In *[Sti90]*, pages 427–441, 1990.

[BG92] Leo Bachmair and Harald Ganzinger. Non-clausal resolution and superposition with selection and redundancy criteria. In *[Vor92]*, pages 273–284, 1992.

[BHP+92] Christoph Beierle, U. Hedtstück, U. Pletat, P. Schmitt, and J. Siekmann. An order sorted logic for knowlege representation. *Journal of Artificial Intelligence*, 55:149–191, 1992.

[Bib82] Wolfgang Bibel. *Automated Theorem Proving*. Vieweg, Braunschweig, 1982.

[Bib83] Wolfgang Bibel. Matings in matrices. *Communications of the ACM*, 26:844–852, 1983.

[BL90] Kim B. Bruce and Giuseppe Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87:196–240, 1990.

[Ble77] W. W. Bledsoe. Set variables. In *[IJC77]*, pages 501–509, 1977.

[Ble79] W. W. Bledsoe. A maximal method for set variables in automatic theorem proving. *Machine Intelligence*, 9:53–99, 1979.

[BLM+86] Robert Boyer, Ewing Lusk, William McCune, Ross Overbeek, Mark Stickel, and Lawrence Wos. Set theory for first-order logic: Clauses for Gödel's axioms. *Journal of Automated Reasoning*, 2:287–327, 1986.

[Boo91] R. Book, editor. *Proceedings of the 4th International Conference on Rewriting Techniques and Applications*, number 488 in LNCS. Springer Verlag, 1991.

[BS94] Franz Baader and Jörg Siekmann. Unification theory. In Dov Gabbay, editor, *Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1994.

[Bun94] Alan Bundy, editor. *Proceedings of the 12th Conference on Automated Deduction*, LNAI, Nancy, France, 1994.

[CAB+86] Robert L. Constable, S. Allen, H. Bromly, W. Cleaveland, J. Cremer, R. Harper, D. Howe, T. Knoblock, N. Mendler, P. Panangaden, J. Sasaki, and S. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[Car84] Luca Cardelli. A semantics of multiple inheritance. In *[KDM84]*, 1984.

[Car88] Luca Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic, Volume 1*. North Holland, 1958.

[CF92] Anthony G. Cohn and Alan M. Frisch. An abstract view of sorted unification. In *[Kap92]*, pages 178–192, 1992.

[CG91] Pierre-Louis Curien and Giorgio Ghelli. Subtyping + extensionality: Confluence of $\beta\eta$-top reduction in $f_{\leq}$. In *[IM91]*, 1991.

[CH85] Thierry Coquand and Gérard Huet. A theory of constructions. In *Semantics of Data Types*. Springer Verlag, 1985.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Coh87] Anthony G. Cohn. A more expressive forumlation of many sorted logic. *Journal of Autmated Reasoning*, 3:113–200, 1987.

[Coh89] Anthony G. Cohn. Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.

[Coh92] Anthony G. Cohn. A many sorted logic with possibly empty sorts. In *[Kap92]*, pages 633–647, 1992.

[CQ94] Régis Curien and Zhenyu Qian. Modular second-order $E$-matchning for regular theories, 1994. draft.

[Cur93] Régis Curien. Second order E-matching as a tool for automated theorem proving. In Miguel Filgueiras and Luís Damas, editors, *Progress in Artificial Intelligence, 6th Protuguese Conference on AI, EPIA '93*, number 727 in Lecture Notes in Artificial Intelligence, pages 242–257, Porto, Portugal, October 1993. Springer Verlag.

[Dar68] J. L. Darlington. Automatic theorem porvind with equality substitutions and mathematical induction. *Machine Intelligence*, 3:113–130, 1968.

[Dar71] J. L. Darlington. A partial mechanization of second order logic. *Machine Intelligence*, 6:91–100, 1971.

[dB72] Nicolaas Govert de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with an application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5):381–392, 1972.

[dB80] Nicolaas Govert de Bruijn. A survey of the project AUTOMATH. In *[HS80]*, pages 579–606. 1980.

[Dig79] Vincent J. Digricoli. Resolution by unificatoin and equality. In *[Joy79]*, 1979.

[Dig81] Vincent J. Digricoli. The efficacy of rue resolution, experimental results and heuristic theory. In *[Dri81]*, pages 539–547, 1981.

[DJ92] Daniel Dougherty and Patricia Johann. A combinatory logic approach to higher-order $E$-unification. In *[Kap92]*, pages 79–93, 1992.

[Dou93] Daniel Dougherty. Higher-order unification using combinators. *Theoretical Computer Science B*, 114(2):273–298, 1993.

[Dow92] Gilles Dowek. Third order matching is decidable. In *[LIC92b]*, pages 2–10, 1992.

[Dri81] Ann Drinan, editor. *Proceedings of the 7th International Joint Conference on Artificial Intelligence (ICJAI)*, Vancouver, Canada, 1981. Morgan Kaufmann, San Mateo, California, USA.

[Ern71] G. W. Ernst. A matching procedure for type theory. Technical report, Case Western Reserve University, 1971.

[Far90] William M. Farmer. A partial-function version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–1291, 1990.

[Far91a] William M. Farmer. Simple second-order languages for which unification is undecidable. *Theoretical Computer Science*, 87(251):25–41, 1991.

[Far91b] William M. Farmer. A simple type theory with partial functions and subtypes. Technical report, MITRE Corporation, Bedford, MA01730 USA, 1991.

[Far93] William M. Farmer. Theory interpretation in simple type theory. In *[HOA93]*, 1993.

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, October 1993.

[Fit90] Melvin Fitting. *First-Order Logic and Automated Threorem Proving.* Springer Verlag, 1990.

[Fra28] Adolf Abraham Fraenkel. Zusatz zu vorstehendem Aufsatz Herrn v. Neumanns. *Mathematische Annalen*, 99:392–393, 1928.

[Fri90] Alan M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1990.

[GM93] M. J. C. Gordon and T. F. Melham. *Introduction to HOL – A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[GOBS69] J. R. Guard, F.C. Oglesby, J.H. Bennet, and L. G. Settle. Semi-automated mathematics. *Journal of the Association of Computing Machinery*, 16(1):49–62, 1969.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360, 1930. English Version 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 Version 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.

[Gor85] Mike Gordon. HOL: a machine oriented formulation of higher-order logic. Technical Report 68, University of Cambridge, Computer Laboratory, July 1985.

[Gou65] William Eben Gould. CRT-aided semi-automated mathematics. Semi-annual report, Applied Logic Corporation, December 1965.

[Gou66] William Eben Gould. A matching procedure for $\omega$-order logic. Technical report, Applied Logic Corporation, One Palmer Square, Princeton, NJ, 1966.

[Gua64] J. R. Guard. Automated logic for semi-automated mathematics. Scientific Report 64–411, Air Force Cambridge Research Laboratories, March 1964.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Her30] Jaques Herbrand. *Recherches sur la théorie de la démonstration*. PhD thesis, Université de Paris, 1930. Englisch translation in [vH67].

[Hib73] Günter Hibsch. Ansatz für ein mechanisches Beweisverfahren für die Prädikatenlogik zweiter Stufe mit Anwendungen auf die Zahlentheorie. Master's thesis, TU-München, 1973.

[HKK+92] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts, and Jörg Siekmann. Ω-MKRP – a proof development environment. Technical Report SR-92-22, Universität des Saarlandes, 1992.

[HKK+94] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts, and Jörg Siekmann. Ω-MKRP a proof development environment. Submitted to the Third International Symposium on Artificial Intelligence and Mathematics, Ft. Lauderdale, USA, 1994.

[HL94] Patricia Hill and John Lloyd. *The Godel Programming Language*. Logic Programming series. MIT Press, 1994.

[HOA93] *HOA'93, an International Workshop on Higher-order Algebra, Logic and Term Rewriting*, Amsterdam, The Netherlands, 1993.

[HRS90] Maritta Heisel, Wolfgang Reif, and Werner Stephan. Tactical Theorem Proving in Program Verification. In *Proceedings of the 10$^{th}$ International Conference on Autom ated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 115–131. Springer Verlag, 1990.

[HRS91] Maritta Heisel, Wolfgang Reif, and Werner Stephan. *Automating Software Design*, chapter 21, pages 547–574. AAAI Press, 1991.

[HS80] R. Hindeley and J. Seldin, editors. *To H.B. Curry: Essays in Combinator Logic, Lambda Calculus and Formalisms*. Academic Press, 1980.

[HS86] J. Hindeley and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, 1986.

[Hue72] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic.* PhD thesis, Case Western Reserve University, 1972.

[Hue73] 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 $\lambda$-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Hue76] Gérard P. Huet. *Résolution d'Équations dans des Langages d'ordre 1,2,...,w.* Thèse d'État, Université de Paris VII, 1976.

[IJC77] *Proceedings of the 5th International Joint Conference on Artificial Intelligence (ICJAI).* Morgan Kaufmann, San Mateo, California, USA, 1977.

[IM91] T. Ito and A. R. Meyer, editors. *Theoretical Aspects of Computer Science*, number 526 in LNCS. Springer Verlag, 1991.

[JK93] Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. SEKI-Report SR-93-14, Universität des Saarlandes, 1993.

[JK94] Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. In *[Bun94]*, pages 620–634, 1994.

[Joh91] Patricia Johann. *Complete Sets of Transformations for Unification Problems.* PhD thesis, Wesleyan University, 1991.

[Joh93] Patricia Johann. A combinator-based order-sorted higher-order unification algorithm. SEKI-Report SR-93-16, Universität des Saarlandes, 1993.

[Joy79] William H. Joyner, editor. *Proceedings of the 4th Workshop on Automated Deduction*, Austin, Texas, USA, 1979.

[JP72] D. C. Jensen and Thomasz Pietrzykowski. A complete mechanization of ($\omega$)-order type theory. In *Proceedings of the ACM annual Conference*, volume 1, pages 82–92, 1972.

[JP73] D. Jensen and T. Pietrzykowski. Mechanizing $\omega$-order type theory through unification. Internal Report CS-73-13, Department of Applied Analysis and Computation, University of Waterloo, 1973.

[JP76] D. C. Jensen and T. Pietrzykowski. Mechanizing $\omega$-order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.

[Jut79] L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH System*, volume 83 of *Mathematical Centre Tracts.* Mathematisch Centrum, Amsterdam, Netherlands, 1979.

[Kap92] D. Kapur, editor. *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, Saratoga Spings, NY, USA, 1992. Springer Verlag.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[KDM84] G. Kahn and G. Plotkin D.G. MacQueen, editors. *Semantics of Data Types*, number 173 in LNCS. Springer Verlag, 1984.

[KK93] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. SEKI-Report SR-93-20 (SFB), Universität des Saarlandes, Saarbrücken, 1993.

[KK94] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. In *[Bun94]*, pages 371–385, 1994.

[Kle52] Stephen C. Kleene. *Introcuction to Meta-Mathematics*. North Holland, 1952.

[Koh92] Michael Kohlhase. Unification in order-sorted type theory. In *[Vor92]*, pages 421–432, 1992.

[Koh93] Michael Kohlhase. Higher-order resolution with combinators. In J. Avenhaus and J. Denzinger, editors, *Informal Proceedings fo the Annual Meeting of "GI-Fachgruppe 'Deduktlionssysteme'" in Kaiserslautern, 1993*, number SR-93-11 (SFB) in SEKI-Report, page 15, 1993.

[Koh94] Michael Kohlhase. Higher-order order-sorted resolution. Seki Report SR-94-1, Fachbereich Informatik, Universität des Sarrlandes, 1994.

[KP93] Michael Kohlhase and Frank Pfenning. Unification in a $\lambda$-calculus with intersection types. In Dale Miller, editor, *Proceedings of the International Logic Programming Sympsion. ILPS'93*, pages 488–505. MIT Press, 1993.

[Lan30] Edmund Landau. *Grundlagen der Analysis*. Wissenschaftliche Buchgesellschaft, Darmstadt, Germany; second edition, 1930. Reprint of the edition, Leipzig, 1970.

[LIC92a] *Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science (LICS-6)*. IEEE Computer Society Press, 1992.

[LIC92b] *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LICS-7)*. IEEE Computer Society Press, 1992.

[LIC94] *Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science (LICS-9)*. IEEE Computer Society Press, 1994.

[LO88] Ewing L. Lusk and Ross A. Overbeek, editors. *Proceedings of the 9th Conference on Automated Deduction*, number 310 in LNCS, Argonne, Illinois, USA, 1988.

[Luc72] Claudio. L. Lucchesi. The undecidability of the unification problem for third order languages. Report CSRR 2059, University of Waterloo, Waterloo, Canada, 1972.

[Mak77] G. S. Makanin. The problem of solvabiliy of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.

[Mil83] Dale Miller. *Proofs in Higher-Order Logic.* PhD thesis, Carnegie-Mellon University, 1983.

[Mil89] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In Peter Schroeder-Heister, editor, *Extensions of Logic Programming: International Workshop, Tübingen FRG, December 1989*, pages 253–281. Springer-Verlag LNCS 475, 1989.

[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.

[MM73] A. Martinelli and U. Montanari. An efficient unification algorithm. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, 1973.

[MN87] Dale Miller and Gopalan Nadathur. A logic programming approach to manipulating formulas and programs. In *IEEE Symposium on Logic Programming*, Salt Lake City, 1987.

[Mor69] James B. Morris. $E$-resolution. In *[WN69]*, pages 287–294, 1969.

[Mül93] Olaf Müller. Optimierung der modularen $E$-Unifikation höherer Stufe. Master's thesis, Universität Karlsruhe, Germany, May 1993.

[MW94] Olaf Müller and Franz Weber. Theory and practice of minimal modular higher-order $E$-unification. In *[Bun94]*, pages 650–677, 1994.

[Nad92] Gopalan Nadathur. A notion of models for an intensional higher-order logic. Unpublished note, 1992.

[Neu28] John von Neumann. Die Axiomatisierung der Mengenlehre. *Mathematische Zeitschrift*, 27:669–752, 1928.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In *[LIC92a]*, pages 342–349, 1991.

[NM88] Gopalan Nadathur and Dale Miller. An overview over $\lambda$PROLOG. Technical Report MS-CIS-88-40, LINC LAB 116, University of Pennsylvania, 1988.

[NQ91] Tobias Nipkow and Zhenyu Qian. Modular higher-order $E$-unification. In *[Boo91]*, pages 200–214, 1991.

[NQ92] Tobias Nipkow and Zhenyu Qian. Reduction and unification in lambda calculi with subtypes. In *[Kap92]*, pages 66–78, 1992.

[Obe62] Arnold Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Mathematische Annalen*, 145:297–333, 1962.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. PVS: a prototype verification system. In *[Kap92]*, pages 748–752, 1992.

[OS89] Hans Jürgen Ohlbach and Jörg Siekmann. The Markgraf Karl Refutation Procedure. In *Computational Logic – Essays in Honor of Alan Robinson*, pages 41–112. MIT Press, Cambridge, 1989.

[Pfe87] F. 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.

[Pfe92] Frank Pfenning. Intersection types for a logical framework. POP-Report 92–106, Carnegie Mellon University, 1992.

[Pfe93] Frank Pfenning. Refinement types for logical frameworks. In Herman Geuvers, editor, *Informal Proceedings of the 1993 Workshop on Types for Proofs and Programs*, pages 285–301, Nijmegen, The Netherlands, May 1993. University of Nijmegen.

[Pie73] Thomasz Pietrzykowski. A complete mechanization of second-order type theory. *Journal of the Association for Computing Machinery*, 20:333–364, 1973.

[Pie91] Benjamin C. Pierce. *Programming with Intersection Types and Bounded Polymorphism.* PhD thesis, School of Computer Science, Carnegie Mellon University, December 1991. Available as Technical Report CMU–CS–91–205.

[Plo72] G. Plotkin. Building in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[PN90] Lawrence C. Paulson and Tobias Nipkow. Isabelle tutorial and user's manual. Technical Report 189, Computer Laboratory, University of Cambridge, January 1990.

[Pre94a] Christian Prehofer. *Higher-Order Equational Reasoning: From Logic to Programming.* PhD thesis, Technische Universität München, 1994. forthcoming.

[Pre94b] Christian Prehofer. Solving higher-order equations. In *[LIC94]*, pages 507–516, 1994.

[Qia91] Zhenyu Qian. *Extensions of Order-Sorted Algebraic Specifications: Parameterization, Higher-Functions and Polymorphism.* PhD thesis, Universität Bremen, März 1991.

[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 Verlag, 1993.

[Qua92] Art Quaife. Automated deduction in von Neumann-Bernays-Gödel set theory. *Journal of Automated Reasoning*, 8(1):91–148, 1992.

[QW94] Zhenyu Qian and Kang Wang. Modular AC unification of higher-order patterns, 1994. draft.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[Rob68] J. A. Robinson. New directions in theorem proving. In *Proceedings of IFIP Congress in Information Processing*, volume 68, pages 63–67. North Holland, Amsterdam, 1968.

[Rob69a] J. A. Robinson. Mechanizing higher order logic. *Machine Intelligence*, 4:151–170, 1969.

[Rob69b] J. A. Robinson. A note on mechanizing higher order logic. *Machine Intelligence*, 5:121–134, 1969.

[Rus08] Bertrand Russell. Mathematical logic as based on the theory of types. *American Jounal of Mathematics*, XXX:222–262, 1908.

[RW69] Arthur Robinson and Larry Wos. Paramodulation and TP in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.

[Sch24] Moses Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. Englisch Version entitled: On the building blocks of mathematical logic in [vH67].

[Sch38] A. Schmidt. Über deduktive Theorien mit mehreren Sorten von Grunddingen. *Mathematische Annalen*, 115, 1938.

[Sch51] A. Schmidt. Die Zulässigkeit der Behandlung mehrsortiger Theorien mittels der üblichen einsortigen prädikatenlogik. *Mathematische Annalen*, 123, 1951.

[Sch60] Kurt Schütte. Semantical and syntactical properties of simple types theory. *Journal of Symbolic Logic*, 25:305–326, 1960.

[SG89] Wayne Snyder and Jean Gallier. Higher-Order Unification Revisited: Complete Sets of Transformations. *J. Symbolic Computation*, 8:101–140, 1989.

[Sie86] J. Siekmann, editor. *Proceedings of the 8th Conference on Automated Deduction*, volume 230 of *LNCS*, Oxford, England, 1986. Springer Verlag.

[Sko19] Albert Thoralf Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze. *Videnskapkasselskapets Skrifter*, I:1–36, 1919. English translation in [vH67].

[Smo89] Gert Smolka. *Logic Programming over Polymorphically Order-Sorted Types.* PhD thesis, Universität Kaiserslautern, 1989.

[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 Verlag, 1968.

[SNGM87] Gert Smolka, Werner Nutt, Joseph A. Goguen, and José Meseguer. Order-sorted equational computation. SEKI-Report SR-87-14, Universität Kaiserslautern, 1987.

[Sny90] Wayne Snyder. Higher order $E$-unification. In *[Sti90]*, pages 573–578, 1990.

[Sny91] Wayne Snyder. *A Proof Theory for General Unification*. Progress in Computer Science and Applied Logic. Birkhäuser, 1991.

[Soc93] Rolf Socher. Unification in order-sorted logic with term declarations. In *[Vor93]*, pages 301–308, 1993.

[SS86] Manfred Schmidt-Schauß. Unification in many-sorted equational theories. In *[Sie86]*, pages 538–552, 1986.

[SS87] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*. PhD thesis, University of Kaiserslautern, 1987. Also [SS89].

[SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *LNAI*. Springer Verlag, 1989.

[Sti86] Mark E. Stickel. Schubert's steamroller problem: Formulations and solutions. *Journal of Automated Reasoning*, 2:89–101, 1986.

[Sti90] Mark Stickel, editor. *Proceedings of the 10th Conference on Automated Deduction*, number 449 in LNCS, Kaiserslautern, Germany, 1990.

[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.

[Tak70] Moto-o Takahashi. A system of simple type theory of Gentzen style with inference on extensionality and the cut-elimination in it. *Commentarii Mathematici Universitatis Sancti Pauli*, XVIII(II):129–147, 1970.

[Tak87] Gaisi Takeuti. *Proof Theory*. North Holland, 1987.

[Tho91] Simon Thompson. *Type Theory and Functional Programming*. International Computer Science Series. Addison-Wesley, 1991.

[Uri92] T. E. Uribe. Sorted unification using set constraints. In *[Kap92]*, pages 163–177, 1992.

[vH67] Jean van Heijenoort, editor. *From Frege to Göel A Soruce Book in Mathematical Logic, 1879-1931.* Source Books in the History of the Sciences. Harvard University Press, 1967.

[Vor92] Andrei Voronkov, editor. *Proceedings of the International Conference on Logic Programming and Automated Reasoning LPAR'92*, volume 624 of *LNAI*, St. Petersburg, Russia, 1992. Springer Verlag.

[Vor93] Andrei Voronkov, editor. *Proceedings of the International Conference on Logic Programming and Automated Reasoning LPAR'93*, volume 698 of *LNAI*, St. Petersburg, Russia, 1993. Springer Verlag.

[Wal83] Christoph Walther. A many–sorted calculus based on resolution and paramodulation. In Alan Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 882–891, Los Altos, California, USA, August 1983. William Kaufmann.

[Wal84] Christoph Walther. Unification in many-sorted theories. In T. O'Shea, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 593–602, Pisa, Italy, 1984.

[Wal85] Christoph Walther. A mechanical solution of Schubert's steamroller by many-sorted resolution. *Artificial Intelligence*, 26(2):217–224, 1985.

[Wal87] Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Pitman, London. Morgan Kaufman Publishers, Inc, 1987.

[Wal88] Christoph Walther. Many-sorted unification. *Journal of the Accociation for Computing Machinery*, 35(1):1–17, January 1988.

[Wan52] Hao Wang. Logic of many-sorted theories. *Journal of Symbolic Logic*, 17, 1952.

[Web93] Franz Weber. *Softwareentwicklung mit Logik höherer Stufe.* PhD thesis, Universität Karlsruhe, Germany, July 1993.

[Wei89] Christoph Weidenbach. A resolution calculus with dynamic sort structures and partial functions. Seki-Report SR-89-23, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1989.

[Wei91] Christoph Weidenbach. A sorted logic using dynamic sorts. Technical Report MPI-I-91-218, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1991.

[Wei93] Christoph Weidenbach. Unification in sort theories and its applications. MPI-Report MPI-I-93-211, Max-Planck-Institut für Informatik, Saarbrücken, Germany, March 1993.

[WN69] Donald E. Walker and Lewis Norton, editors. *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, 1969.

[WO90] Christoph Weidenbach and Hans Jürgen Ohlbach. A resolution calculus with dynamic sort structures and partial functions. In *[Aie90]*, pages 688–693, 1990.

[Wol93] David A. Wolfram. *The Clausal Theory of Types*. Cambridge University Press, 1993.

[WR10] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume I. Cambridge University Press, Cambridge, Great Britain; second edition, 1910.

[Zai87] M. Zaionc. Word operation definable in the typed $\lambda$-calculus. *Theoretical Computer Science*, 52:1–14, 1987.

[Zer08] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre. I. *Mathematische Annalen*, 65:261–281, 1908.

[ZK88] Hantao Zhang and Deepak Kapur. Frist order theorem proving using conditional rewrite rules. In *[LO88]*, pages 1–20, 1988.


# 10 Table of Defined Symbols

currently under construction