A Typed Operational Semantics for Type Theory

Healfdene Goguen

Doctor of Philosophy

University of Edinburgh

1994

Abstract

Untyped reduction provides a natural operational semantics for type theory. Normalization results say that such a semantics is sound. However, this reduction does not take type information into account and gives no information about canonical forms for terms. We introduce a new operational semantics, which we call typed operational semantics, which defines a reduction to normal form for terms which are well-typed in the type theory.

The central result of the thesis is soundness of the typed operational semantics for the original system. Completeness of the semantics is straightforward. We demonstrate that this equivalence between the declarative and operational presentations of type theory has important metatheoretic consequences: results such as strengthening, subject reduction and strong normalization follow by straightforward induction on derivations in the new system.

We introduce these ideas in the setting of the simply typed lambda calculus. We then extend the techniques to Luo's system *UTT*, which is Martin-Löf's Logical Framework extended by a general mechanism for inductive types, a predicative universe and an impredicative universe of propositions. We also give a proof-irrelevant set-theoretic semantics for *UTT*.

Acknowledgements

My supervisors Rod Burstall and Zhaohui Luo have been invaluable, and I greatly admire both of them. Rod was open and welcoming from when I first met him, and his balance between practical and theoretical interests and his ability to see important directions for research are impressive. Luo taught me most of what I know about type theory and more generally gave me an ability to appreciate detail. He was always friendly and open for discussion.

The LFCS provided an excellent environment for doing research, and the LEGO club in particular stimulated my interest in type theory. I had interesting discussions with Matt Fairtlough, Philippa Gardner, Martin Hofmann, John Longley, Randy Pollack and Alex Simpson. Randy Pollack's LEGO proof assistant made learning type theory easier by providing a good implementation of type theory. I would also like to thank Savi Maharaj for submitting the thesis for me.

Two people who had a more substantial influence on this thesis are Thorsten Altenkirch and James McKinna. I had many beneficial discussions with Thorsten about type theory in general and the metatheory of type theories. James explained a broad range of topics to me over innumerable cups of coffee and tea. Both also read a draft version of this thesis and suggested useful improvements.

My step-father gave me my initial enthusiasm for computer science and taught me valuable problem-solving techniques. My mother gave me essential financial support and was encouraging throughout.

My father has had a large influence on this thesis. He first encouraged me to study theoretical computer science and suggested studying at the University of Edinburgh. In addition to letting me live with him for several months during a difficult period of my work, he was supportive of my technical work and sympathetic to the difficulties involved in writing a thesis.

This thesis would almost certainly not have been written without the long-standing support of Ted Kowalski at AT&T Bell Laboratories. Ted hired me for many years, and he taught me valuable technical and research skills as well as having an infectuous enthusiasm for his work and a wonderful ability to work with people. I spent a valuable summer working for Amy Felty and Elsa Gunter at AT&T Bell Laboratories.

I was supported financially for two years by an Overseas Research Students Award. The Science and Engineering Research Council and the European Community TYPES Basic Research Action provided funding for my last six months in Edinburgh. The CROAP project at INRIA Sophia-Antipolis also supported me while I finished the last parts of this thesis.

I would also like to thank my examiners, Alan Smaill and Jan Smith, for their careful reading of the thesis and their helpful comments.

The long time that I took in writing my thesis was made much more pleasant than it might have been because I lived with people who were both patient and supportive. I would like to thank Gail Durie, my parents in Oxford and my sister Alice, Thorsten Altenkirch and the Bath Street Housing Co-op, Sophie and Zazou Smith, and Penny Anderson.

Declaration

I declare that this thesis was composed by myself, and the work contained in it is my own except where otherwise stated.

Healfdene Goguen

Table of Contents

Chapter 1

Introduction

Many specification and verification systems are based on traditional mathematical formalisms such as first-order logic and set theory. However, although these formalisms are well suited to the tasks for which they were designed, they have shortcomings when used in this setting. In recent years type theory, originally designed as a basis for formalizing constructive mathematics, has been put forward as an alternative.

Type theory views computation as fundamental. A type is defined by giving its canonical elements and defining when two canonical elements are equal. We also introduce non-canonical elements corresponding to induction or recursion over the canonical elements of a type. The process of reducing non-canonical elements to canonical ones provides a natural operational semantics for programs written in type theory. Furthermore, the importance of computation in the formulation of type theory leads to implementations that are more straightforward than those of traditional logics.

In type theory, propositions are themselves types whose elements are their proofs. Because proofs are objects in the system, it is easy to check their correctness, which provides a convenient way to study program extraction. Furthermore, because computation and logical inference are dealt with in a unified framework, the gap in other specification languages between the programming language and the specifications is absent. Hence type theory offers an implementable formalism with theoretical benefits over more traditional systems.

One significant shortcoming of type theory from the perspective of computer science has been the lack of a well-behaved, uniform mechanism for understanding datatypes. One system with datatypes that is uniform and intuitively clear is Martin-Löf's extensional type theory. This theory explains datatypes as well-orderings and assumes that functions are extensional—that is, two functions are equal if their domain and range are equal and their result is equal for all elements in the domain. Unfortunately, the system with extensionality fails to have some important properties: for example, type-checking in the system is not decidable.

Zhaohui Luo has proposed a system, which he calls a Unifying Theory of Dependent Types or UTT, that avoids this problem. He uses a logical framework to describe the type theory, much as one uses such frameworks to describe conventional logics. However, instead of adding new types individually, such as the type of pairs, he gives a schema for describing all types of a certain form. These types correspond closely to those used in programming. The schema uses only the syntactic notion of an expression with holes in it that is provided by the logical framework. This system is based on formulations by Backhouse [6], Dybjer [25] and Coquand and Paulin-Mohring [19].

In this thesis we shall show that desirable properties, such as consistency and termination of computation, hold for this calculus. In doing so, we shall introduce a new system to help us understand the computational behavior of type theory. This system is an operational semantics which combines a special reduction relation which is universal with respect to all reductions with type information. We shall see that this new system leads to considerable technical improvements in the study of the relationship between logic and computation. Our primary interest throughout the thesis will be the exploration of the application of this idea to our understanding of the metatheory of type theories.

1.1 A Brief Description of Type Theory

We shall begin by introducing the basic ideas of type theory, giving substance to the general description in the opening paragraphs. We present the kinds of statements that we can make in type theory and give an explanation of how we understand these statements. The explanations of the meanings of the statements justify the formal rules in the system. We also explain the general structure of the presentation of a new type in type theory. Finally, we describe how propositions can be understood within the type-theoretic framework.

Our description here is based on the work of Martin-Löf, in particular Martin-Löf's book [54] and Nördstrom, Petersson and Smith's book describing Martin-Löf's type theory [63]. Martin-Löf calls the entities in his theories "sets" where we have called the entities "types."

1.1.1 The Semantic Explanation of Judgements

Type theory is a formal language based on a conceptual organization of objects. A type is a collection of objects with some common property: we have types of pairs, types of functions and the type of natural numbers. The basic statements we make in type theory are concerned with the validity of some property or whether some object has a property. We are also interested in whether different expressions denote the same object in a type.

Less informally, we have two basic entities, objects and types. We shall say that an object is *canonical* or in canonical form if its outermost constructor is an introduction constant. We make statements or *judgements* about the relationship between objects and types. The type theories we shall be interested in have four forms of judgement, each with a corresponding semantic explanation:

A type This means that we know the canonical objects that are elements of A, and furthermore we have some notion of equality for any two objects in canonical form in A. The equality relation for the type A must be a decidable equivalence relation.

$A = B$ This judgement says that the types A and B have the same canonical objects as elements, and furthermore the equalities associated with A and B are the same.

$M : A$ This says that M is a method or program which when executed yields a canonical object which is an element of the type A. We must already know that A is a type in order for this to make sense.

$M = N : A$ We take this to mean that M and N are methods which when executed yield canonical objects associated with them which are both elements of the type A, and furthermore the canonical objects are equal relative to the notion of equality for A.

We shall call the presentation of the type theory which arises out of these semantic explanations the *semantic presentation* of the type theory.

Hence $\mathrm{succ}(1+1)$ is a canonical element of the natural numbers, because succ is a constructor for the natural numbers and $1+1$ is a natural number, since we can reduce it to a canonical natural number as well. The method of finding a canonical form from a non-canonical form, the notion of computation, is taken as a primitive notion in type theory. We assume that this method of evaluation proceeds from the outside of a term.

1.1.2 Judgements with Hypotheses

We also want to make judgements under assumptions. We explain the meaning of judgements under one assumption using the previously given explanation of judgements with no assumptions. Each of these judgements with a single assumption assumes that the type associated with the assumption is already well-typed, that is we must already know C type below:

$z{:}C \vdash A$ type If we know this judgement then we know that $[P/z]A$ type holds for every P such that $P\colon C$. Furthermore, we know that $[P/z]A = [Q/z]A$ if $P = Q\colon C$, where we take $[P/z]A$ to be the result of substituting P for each free occurrence of z in A.

$z{:}C \vdash A = B$ This judgement means that $[P/z]A = [P/z]B$ for every $P\colon C$.

$z{:}C \vdash M\colon A$ This judgement means that we know that $[P/z]M\colon [c/z]A$ for every $P\colon C$, and furthermore that $[P/z]M = [Q/z]M\colon [P/z]A$ if $P = Q\colon C$.

$z{:}C \vdash M = N\colon A$ This judgement means that $[P/z]M = [P/z]N\colon [P/z]A$ for every $P\colon C$.

This explanation of judgements under a single assumption can be extended to judgements with finite sequences of assumptions in a straightforward manner.

We shall add to these judgement forms another judgement form, that

$$\vdash z_1\colon C_1, \ldots, z_n\colon C_n$$

which we take to mean that

- C_1 type,
- $\ldots$,
- $[P_1, \ldots, P_{n-1}/z_1, \ldots, z_{n-1}]C_n$ type for any
  - $P_1 \in C_1$,
  - $\ldots$,
  - $P_{n-1} \in [P_1, \ldots, P_{n-2}/z_1, \ldots, z_{n-2}]C_{n-1}$.

1.1.3 General Presentation of Types

There is a common pattern in the rules for introducing types in type theory. Each type will be defined by giving rules in each of four general categories:

Formation Rules These rules, one for each type, will dictate how to form the new type from other types, families of types and elements of types. We also need to say under what circumstances the new type is equal to other types.

Introduction Rules These rules introduce the canonical elements of the type being defined and when two canonical elements are equal. It is these rules which give meaning to the type, in the sense that we have fulfilled the requirements for knowing how to judge that we have a type.

Elimination Rules These rules allow us to define functions or programs on the type which has been defined by the introduction rules.

Equality Rules The equality rules relate the introduction and elimination rules. They show how the function defined by the elimination rule behaves on the canonical elements of the type, the canonical elements having been defined by the introduction rules.

In Section 3.3.1 we give an example of how this general pattern is used by explaining in depth the introduction of the type of natural numbers.

1.1.4 The Propositions-as-Types Embedding

A fundamental idea in the justification of type theory as a foundation for constructive mathematics or as a basis for specification and verification of programs is the principle of *propositions as types*. This principle is based on the observation by Curry [20] and Howard [39] of the close correspondence between systems of natural deduction for intuitionistic logical inference and type systems.

The underlying idea of this principle is that we can relate propositions to types by understanding a proposition as the type of proofs of the proposition. We can see the similarity between the traditional reading of the intuitionistic connectives and the canonical objects we have discussed above: to have a proof of $A \wedge B$ we must have a pair of proofs, one of A and one of B; a canonical element of the type of proofs of $A \wedge B$ is a pair of an element of the type of proofs of A and an element of the type of proofs of B. Similarly, to have a proof of $A \supset B$ we must have a map from proofs of A to proofs of B; an element of the type of proofs of $A \supset B$ is a function from elements of the type of proofs of A to elements of the type of proofs of B.

In this context the method of evaluating a term to find a canonical form corresponds to the process of proof normalization or cut elimination in systems of natural deduction. This says that we can remove inessential elimination–introduction pairs in proofs, resulting in a proof which can be recognized directly as a proof according to the intuitionistic reading of the connectives.

Dependent types or families of types allow us to understand quantification under the propositions-as-types paradigm. A proof of the proposition $\forall x\colon A.P$, where x can occur in P, will be a function which for each $M\colon A$ returns an element of the type of proofs of $[M/x]P$. Here we have formed the judgement P type under the assumption that $x\colon A$, a process which was explained above in Section 1.1.2.

The systems studied by Curry and Howard were systems for which there was an equivalence between propositions and types. This equivalence holds for various logics and type theories: for example, an extension of the simply typed lambda calculus corresponds to full intuitionistic first-order logic, as developed by Howard [39]; and System F corresponds to second-order propositional logic, where the former type theory and the equivalence were studied by Girard [32]. For this reason the propositions-as-types embedding is also referred to as an isomorphism.

Second-order logic includes the powerful logical principle of impredicativity. This principle allows us to quantify over all propositions to form a new proposition. The basic type theory with impredicativity or polymorphism, System F, was introduced independently by Girard [32] and Reynolds [67].

The mechanisms of impredicativity and dependent types were combined by Coquand and Huet [18] in the Calculus of Constructions. This system is a type system corresponding to a powerful higher-order logic and provides a strong basis for the formalization of mathematics.

Martin-Löf has extended the equivalence of propositions and types by identifying the two concepts for more complex theories. Hence in Martin-Löf's type theory, the judgement "A is a proposition" is simply understood as another way of saying that A is a type, and to know that a proposition A is true is simply to know that A is inhabited.

An alternative view is that although type theory provides a framework in which to understand both logical inference and computation, we need not identify these two ideas. We can indeed treat propositions as types, but we do not need to see types as propositions as well. Luo's book [47] lists several reasons for viewing the identification of propositions and types as unnatural: first, the logic of our system should be independent of the objects studied in it; secondly, certain types such as the natural numbers do not intuitively correspond to propositions; thirdly, type theory is often considered open to the addition of new types representing new computational or mathematical objects, but the addition of these objects should not change the way we reason in the logic. Furthermore, results about the conservativity of type theories which identify propositions and types over their related logics [9,45] show that this identification does not correspond to the traditional way of formulating logics. Therefore, we shall have a formal distinction between propositions and types in the formulation of the type theory we shall study in this thesis.

An important property of a logic is that we should be able to recognize that a proof does indeed prove the proposition it purports to. According to the propositions-as-types embedding, we have mapped proofs of a proposition to elements of the type of proofs of the proposition. The judgements of our type theory must therefore be decidable, so that we can tell from the form of a judgement $M : A$ that M is indeed an element of the type A. In other words, if we put forward M as a proof of the proposition A then we should be able to recognize that M does prove A.

1.2 Type Theory and Computer Science

The above discussion has described type theory as a general formalism for understanding mathematical structure. There is an alternative interpretation of the judgements of type theory which is more closely related to computer science: we can view the types as specifications of programs and the elements of types as programs. A canonical object under this interpretation of type theory is a program which has been fully evaluated or run. This interpretation of type theory has given rise to a significant amount of interaction between type theory and computer science.

1.2.1 Language Design and Semantics

Type theory is close enough to programming languages and motivations that we can use it as a tool to help us to design and understand programming languages. This is similar to the influence of the untyped lambda calculus on programming languages, leading to the development of LISP and the field of denotational semantics. We give a brief indication of some of the applications here.

Type theory has been used in the design of modern programming languages. A good example is the language Standard ML [37]. Standard ML is a programming language with weak polymorphism, based on the polymorphism of impredicative type theories. The language has type inference, which allows the user to give terms with no type annotations and the type checker finds a most general type. System F or the polymorphic lambda calculus was itself discovered independently as a type theory and as a programming language, as we mentioned above. Furthermore, the modules system in Standard ML has been designed from the basis of a type-theoretic understanding of modularity and abstraction, as discussed by MacQueen [49].

There is considerable interest in using type theories with subtyping to give a semantics for object-oriented programming. We can hope that this effort will lead to a cleaner design of object-oriented languages, similar to the development of the modules system for Standard ML.

1.2.2 Specification and Verification

The problems of modularization and abstraction are major areas of concern in computer science. The conceptual clarity of the type-theoretic approach suggests that type theory may provide a good basis for mechanisms to deal with these problems.

The combined mechanisms of polymorphism and dependent types provide a good environment for studying modularity. Polymorphism allows us to abstract over types, and the dependency allows us to specify program modules over these abstracted types and morphisms between modules.

The internal logic in type theory allows us to consider specification and verification as well. We can formulate predicates over structures, specifying the properties that implementations of the specification should satisfy, as objects within the type theory. Proofs that these predicates hold are also internal to the type theory. Furthermore, the standard operations on specifications can be implemented in type theory. This uniformity of presentation is different from the traditional style of program verification, where a logical language is developed on top of a programming language. There is similarly a gap in the algebraic approach to specification between the operational and algebraic semantics of programs.

1.2.3 Inductive Types

If we are to pursue this relationship between type theory and computing, we must have a mechanism for inductive types in type theory. The ties between programming and type theory are only interesting if the programs we develop in type theory are able to use the types commonly found in programming, such as natural numbers, lists and trees. Also, if we consider type theory as a possible foundation for constructive mathematics, as inspired by Bishop's effort to develop constructive analysis [10], then type theory can only be seen as adequate if it is able to formalize fundamental mathematical concepts captured by inductive types.

We should both be able to write programs with the inductive types and have induction principles which allow us to reason about the programs written using the types. Although the computational principles for inductive types have long been understood, finding a formulation of type theory with a general mechanism for inductive types with an adequate representation of the induction principles has been difficult. We discuss some of the formulations of systems with inductive types in Section 4.1.3.

One of the main contributions of this thesis is that we show that important properties hold for programs written in a type theory with an adequate representation of data types. This provides us with a strong theoretical basis for the study of the practical aspects of the type theory UTT with respect to modularity and the specification and verification of programs.

1.2.4 Implementations of Type Theory

An early implementation of type theory with many important contributions is de Bruijn's Automath project [22]. de Bruijn introduced the idea of using type theory as a system which can serve as a framework for implementing logics, by giving a system which formalizes the underlying principles which mathematicians agree upon. Significant formal developments of mathematics were done in the system, for example the development of Landau's *Grundlagen* by van Bentham Jutting [79].

Coq [23] is an implementation of a system based on the Calculus of Constructions, with a distinction between propositions and types, proof irrelevance and inductive types. ALF [5] is a structure editor for Martin-Löf's type theory in the Logical Framework, including a window-based user interface. Coquand's system of pattern matching [16] is also supported on an experimental basis. NuPRL [12] implements Martin-Löf's extensional type theory.

LEGO [48] is an implementation of several different type theories: the Edinburgh Logical Framework [36]; the Pure Calculus of Constructions [18]; ECC [44]; and one similar to UTT, the formal system we explore in this thesis and also described in Luo's book [47]. This implementation is a type checker and proof assistant for these type theories extended with explicit definitions (discussed further in Section 7.2), existential variables and a mechanism for defining inductive types.

1.3 A Computational View of Type Theory

We have already said that untyped reduction provides a natural operational semantics for type theory. Normalization results say that such a semantics is sound. However, the untyped reduction does not take into account type information, despite the importance of the relationship between computational equality and well-typedness, and furthermore it gives us no information about canonical objects. In this thesis we introduce a new kind of operational semantics, which we call *typed operational semantics*, which defines a reduction to normal form for terms which are well-typed in the type theory. We prove that this new semantics is equivalent to the semantic presentation of type theory, and we shall see that this equivalence between the declarative and operational presentations of type theory will have important metatheoretic consequences.

We shall study a specific typed operational semantics related to the notion of standard reduction in the untyped lambda calculus. However, this is not the only possible operational view of type theory: there are many other reduction strategies for the type theory, and these too may offer useful analyses of the theory. For example, we can easily define a typed operational semantics corresponding to call-by-value reduction, and it seems that the soundness proof for this system would be easier than the proof we are able to use for our system. We have restricted ourselves to studying our particular semantics because it seems to be a good system for studying the most basic metatheoretic properties for a type theory.

Our typed operational semantics will have the following judgement forms:

$\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ **type** Informally this judgement means that A and B are types under the assumptions in Γ and that the type A is associated with the canonical type B. This will mean that we know which canonical objects are elements of the type B.

$\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : B$ This judgement means that the object M is associated with the canonical form P and that P is an element of the type B under the assumptions in Γ, where B is itself a canonical type.

$\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ This judgement means that we can perform a step of outermost computation on the term M, and this computation yields the term N, where both M and N are elements of A under the assumptions in Γ.

We do not intend this explanation of the judgements to justify the presentation of the system in the same way that the explanations in Section 1.1 justify the rules of inference for Martin-Löf's presentation of type theory. We merely give an informal explanation to give some indication of the motivations for the formal system. Our main interest in the system is as a tool for studying the metatheory of the semantic presentation.

However, informally our description of the meaning of these judgements differs from the description of the meaning of the judgements of the semantic presentation in two fundamental ways:

- We have removed the unexplained concept of how to find the canonical form associated with an object. This method is described to us by the rules of inference of the theory.

- We have taken as primitive the notion of a judgement holding under assumptions. Unlike in the semantic presentation of type theory, the rules of substitution cannot be justified in this calculus.

The typed operational semantics will be a powerful tool for studying the relationship between reduction and type information. We shall be able to show all of the important properties about typed terms and reduction by reasoning in this typed operational semantics. Strengthening, often a difficult result for type theories with dependent types, will also be straightforward. Finally, we shall see that our typed operational semantics provides a good induction principle for showing results about reduction in general, related to the notion of type closed predicate introduced by Mitchell [62].

It is straightforward to establish that the typed operational semantics is a complete interpretation of its semantic presentation. We can view this result as saying that the typed reduction or evaluation relation defined by the typed operational semantics is subsumed by the equality of the semantic presentation. Since reduction is essentially the transcription of the rules of equality excluding symmetry, and because we have included type information in our semantics, it is not surprising that this is straightforward.

The most difficult result concerning our typed operational semantics will be to show that it is a sound interpretation of the semantic presentation. The operational semantics, being a system motivated by computation, will have no general rules for application or substitution, and even at the level of judgements it is difficult to see how these rules could be defined. The soundness result will say that the rules for application and substitution are admissible. It is not a coincidence that the standard technique for proving strong normalization is also very well suited to establishing this result—our definition of the typed operational semantics was largely motivated by our study of strong normalization proofs. Through the soundness and completeness results, we can transfer the development of the meta-theory for the typed operational semantics to the semantic presentation of the type theory.

Typed operational semantics are interesting for their applications to the meta-theory of type theories, and in particular in the study of the relationship between reduction and well-typedness, rather than as a practical, implementable system. For implementations, there are other presentations that are considerably more efficient for checking whether a term is well-typed. Also, unlike the operational semantics for programming languages, untyped reduction is considerably more efficient in computing normal forms than the typed operational semantics.

1.4 An Overview of the Thesis

Chapter 2 introduces the idea of typed operational semantics in the context of the simply typed lambda calculus. We present the simply typed lambda calculus and its typed operational semantics, the formal system λ^S, and we prove the important metatheoretic results for these systems. This chapter is intended to be independent of the rest of the thesis, although the structure of the presentation in this chapter is reflected in the thesis as a whole.

In Chapter 3 we introduce the calculus UTT, also presented in Luo's book [47]. We give both a formal presentation of the judgements and rules of inference and a more informal description of the system and how it can be used.

In Chapter 4 we develop the basic metatheory for UTT. We introduce the formal system UTT^S, the typed operational semantics for UTT, and prove basic results about the system. We also give a general discussion of different ways of presenting type theories and how these different presentations lead to different approaches to the metatheory.

We give a set-theoretic semantics for UTT in Chapter 5. The model is proof irrelevant, interpreting propositions as either true or false, and types are interpreted as their intuitive set-theoretic counterparts. The construction used is closely modeled on Dybjer's construction [26].

Chapter 6 is devoted to the central result of the thesis, soundness of UTT^S for UTT. This proof is closely related to traditional proofs of strong normalization, although the complexity of the calculus and the fact that we prove soundness for a typed notion of reduction mean that we introduce some refinements to the usual proof technique.

Finally, in Chapter 7 we summarize the contributions of this thesis and mention some areas for further research.

1.5 Related Work

Much of the discussion in this introduction is given in more detail by Martin-Löf [53,54] and Luo [47]. The latter presents the calculus *UTT* and gives a full account of this system, the metatheory for the simpler *ECC* and a general discussion of type theory. There is also a thorough treatment of *ECC* in Luo's thesis [44]. *UTT* is formulated in Martin-Löf's Logical Framework, as presented by Nördstrom, Petersson and Smith [63]. Harper, Honsell and Plotkin [36] give a different Logical Framework, intended as a framework for presenting traditional logics rather than type theories with added computational types.

Mitchell [62] and Coquand [15] were both influential in our development of a typed operational semantics for type theory. The first of these defines a general class of predicates on typed terms by giving conditions which the predicates must satisfy and shows that any such predicate is true for all well-typed terms. The second paper gives an algorithm for deciding conversion for a particular type theory with $\beta\eta$-equality. We discuss these two papers and their relation to our typed operational semantics in Sections 2.5 and 4.10. Harper and Pollack [38] and Constable *et al.* [12] give explicit definitions of weak head reduction for type systems, a notion which we use in our typed operational semantics.

The basic metatheory for the system UTT^S in Section 4.9 is similar to other such developments. Our work was influenced by Luo [44,47], who gives an excellent treatment of the metatheory for *ECC*, including an early treatment of strengthening for systems with dependent types. The metatheory for Pure Type Systems [8,31,80] has also influenced our work.

Gentzen [29] and Prawitz [66] give early proofs of cut-elimination for the sequent calculus and for systems of natural deduction. The latter corresponds via the principle of propositions as types to a proof of normalization for the corresponding type theory.

There is a large body of work on normalization proofs for type theory. Tait [76] first developed the technique of reducibility to prove strong normalization for Gödel's System T, which is essentially the simply typed lambda calculus with recursion over natural numbers. Girard [32] extended this technique to System F, which includes impredicative quantification. There is also a discussion of these proofs and the relationship between type theory and proof theory in the book by Girard, Lafont and Taylor [33].

Systems with dependent types add another dimension of difficulty to normalization proofs. Martin-Löf [52] gives a proof of strong normalization for one of his early type theories and contains many important insights. This paper presents the type theory using an untyped notion of reduction and equality. His later paper [53] considers a weaker notion of reduction. This paper also uses judgemental equality and gives one of the few proofs of normalization for a typed notion of reduction. More recently, Pottinger [65] and Coquand and Gallier [17] address problems similar to those of the original Martin-Löf paper in the context of the Calculus of Constructions using two different but related techniques based on Kripke-style reasoning. Coquand [15] shows the decidability of type checking for the Logical Framework with a universe and Π-types, and Luo's thesis [44] shows strong normalization for a significant subsystem of UTT. Catarina Coquand [13] gives an excellent explanation of a proof of normalization for the simply typed lambda calculus. We shall discuss many of these works in more detail when we consider our proof of soundness.

The above proofs construct models with well-typed terms. There are also proofs of normalization for the Calculus of Constructions and related systems which interpret types simply as sets of normalizing terms with no type information [30,8,83].

Less work has been done on the metatheory for systems with inductive types. Martin-Löf's original proofs mentioned above are for a theory with his well-ordering type, which allows many inductive types to be defined. Tait's original proof was for Gödel's System T, with primitive recursion over natural numbers. Werner's abovementioned work shows normalization for a type theory with impredicativity, natural numbers and the so-called "large" eliminations. Löfwall and Sjödin [43] prove strong normalization for Martin-Löf's type theory with natural numbers and universes. Mendler [59] gives a proof of strong normalization for inductive definitions in the polymorphic lambda calculus. Werner's recent thesis gives a proof for a system with a general class of inductive types, but this work has not been made available to me.

The present thesis was developed in a time period overlapping with Altenkirch's thesis [4]. This work gives a proof of normalization for the Calculus of Constructions using terms with fully typed application and abstraction under β-reduction. The proof uses a model with untyped terms and is done via an intermediate soundness proof for a modification of a realizability model. There is also an outline of how to extend the proof to a general class of inductive types. Although I benefited considerably from discussions with the author of this work, the approaches taken and the systems considered are significantly different: my work gives a proof for typed terms and uses the new technique of giving a typed operational semantics, and also my result is for an extension of Martin-Löf type theory presented in the Logical Framework with $\beta\eta$-equality, with an impredicative universe of propositions, a general class of inductive types and a single predicative universe.

For $\beta\eta$-equality, van Dalen's thesis [81] is an early proof of the Church–Rosser property for $\beta\eta$-equality in the Automath project. Salvesen [70] uses applications with type labels to show the Church–Rosser and subject reduction properties for Pure Type Systems which are normalizing. Geuver's thesis [30] studies Church–Rosser and strong normalization for various Pure Type Systems and in particular the Calculus of Constructions.

Constructing models for the Calculus of Constructions with $\beta\eta$-equality is discussed in detail in Streicher's book [75], with close attention to the syntax. Salvesen [69] and Dybjer [26] give set-theoretic semantics for Martin-Löf's intensional type theory. Ore [64] and Fu [27] also discuss models for type theories with inductive types. Smith [72] interprets Martin-Löf type theory with judgemental equality and a predicative universe in Aczel's Frege structures [2], and we would need a similar technique if we did not interpret equality as syntactic identity. Allen [3] gives a non-type theoretic interpretation of Martin-Löf's type theory.

Chapter 2

The Simply Typed Lambda Calculus

In this chapter we shall introduce the idea of typed operational semantics for type theory by considering the full metatheoretic development for the simply typed lambda calculus. This chapter can be read separately from the rest of the thesis. However, many of the important ideas for typed operational semantics are illustrated and discussed in this chapter, and at many points later in the thesis we shall refer to discussions here.

As we discussed in the introduction, a typed operational semantics will be a calculus in which only terms that are normalizing via a standard reduction sequence are well-typed, and in which the judgement forms contain the normal forms of all well-typed terms. As such it is closely related to operational semantics for programming languages. It turns out that this system is very useful for reasoning about the relationship between type information and reduction.

We shall not explain the simply typed lambda calculus in detail. We instead refer the reader to Mitchell's thorough introduction to the calculus [62]. He also gives a good account of the usual development of the metatheory for this system, and we shall use both his work and the book by Girard, Lafont and Taylor [33] as sources with which to compare our work.

This chapter is organized as follows. We give a brief introduction to the simply typed lambda calculus and present the syntax of the system in Section 2.1. In

Section 2.2 we introduce the typed operational semantics, give an intuition for the rules and discuss some of the design decisions involved in our choice of rules of the system. Section 2.3 contains the basic development of the metatheory, both for the simply typed lambda calculus and for the typed operational semantics. In Section 2.4 we give a proof of soundness of the typed operational semantics for the simply typed lambda calculus, the most difficult result relating the two presentations of the type theory. This proof is closely related to the proof of strong normalization for the simply typed lambda calculus. Finally, we discuss work related to typed operational semantics in Section 2.5.

2.1 The Proof System

We begin by presenting the simply typed lambda calculus using what we have called the semantic presentation in Section 1.1.1. This presentation follows the semantic explanation of judgements given by Martin-Löf, and hence differs from the usual presentation of the simply typed lambda calculus [62] in having rules for thinning of hypotheses and for substitution. This is similar to the presentation of Martin-Löf's Logical Framework [63].

2.1.1 Types and Terms

We first introduce the types of the simply typed lambda calculus. For simplicity we shall assume that there is only one base type o. We also have the type of functions from A to B, $A \rightarrow B$, for any two types A and B.

We assume the existence of an infinite set V of variables. The terms of the simply typed lambda calculus are defined inductively:

- If $x \in V$ then x is a term.
- If $x \in V$, A is a type and M is a term then the *abstraction* of M by x and A, $\lambda x{:}A.M$, is a term. We call M the *body* of the abstraction.

- If M and N are terms then the *application* of M to N, $M(N)$, is a term.

We identify terms which are equivalent up to the renaming of bound variables. We denote the free variables in a term M, those variables not bound by abstractions, by $\mathrm{FV}(M)$. We denote the substitution of N for the free variable x in M by $[N/x]M$, where this substitution is defined to avoid the capture of free variables.

A sequence $\Gamma \equiv x_1{:}A_1, \ldots, x_n{:}A_n$ of pairs of variables $x_i \in V$ and types A_i is a *pre-context*. We write $dom(\Gamma)$ for the set $\{x_1, \ldots, x_n\}$. The empty pre-context is denoted by $()$.

A *context* is a pre-context where the x_i are distinct.

2.1.2 Judgements and Derivations

Because of the simplicity of the calculus, we can assume that we know what the canonical elements of any type are and when two canonical elements of any type are equal before we define the well-typed terms. This is the same as saying that we have a judgement A type which we know holds for any type.

We have two judgement forms in our presentation of the simply typed lambda calculus, which are informally described as follows:

- $\Gamma \vdash M : A$. We take this to mean that the canonical form associated with M is an element of A. In words we say that "M has type A in context Γ" or "Γ yields M in A."

- $\Gamma \vdash M = N : A$. We take this to mean that we know what the canonical forms of M and N are and that these canonical forms are equal under the equality for A. We say that "M and N are equal objects of type A in context Γ" or "Γ yields M equals N in A." We call this judgement form *judgemental equality.*

The rules of inference for the calculus are given in Figures 2–1 and 2–2.

Notation We shall write $\Gamma \vdash J$ for an arbitrary judgement in the simply typed lambda calculus.

$$(Var)\ \frac{\Gamma\ \text{context} \qquad x{:}A \in \Gamma}{\Gamma \vdash x\ :\ A} \qquad (Thin)\ \frac{\Gamma_0, \Gamma_1 \vdash M\ :\ A \qquad z \notin dom(\Gamma_0, \Gamma_1)}{\Gamma_0, z{:}C, \Gamma_1 \vdash M\ :\ A}$$

$$(\lambda)\ \frac{\Gamma, x{:}A \vdash M\ :\ B}{\Gamma \vdash \lambda x{:}A.M\ :\ A \to B}$$

$$(App)\ \frac{\Gamma \vdash M\ :\ A \to B \qquad \Gamma \vdash N\ :\ A}{\Gamma \vdash M(N)\ :\ B}$$

$$(Subst)\ \frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M\ :\ A \qquad \Gamma_0 \vdash N\ :\ C}{\Gamma_0, \Gamma_1 \vdash [N/z]M\ :\ A}$$

Figure 2–1: Typing Rules

2.1.3 Discussion

The rules of inference (*Thin*), (*Subst*) and the substitution rules for equality are not in the standard presentation of the calculus [62]. We have included these rules in order to follow the semantic explanation of the judgements in Section 1.1, where judgements with hypotheses are explained by substitutions into judgements with fewer hypotheses. However, because these rules are admissible rules in the usual presentation (that is, if there are derivations of the premisses then there is a derivation of the conclusion), adding them does not constitute an essential change to the system.

Derivations of judgemental equality can only follow sequences of well-typed terms, in contrast to conversion, which is defined to be the least equivalence relation containing reduction. In Section 2.3, on the metatheory for the simply typed lambda calculus, we shall define what for us is the auxiliary notion of reduction. We shall also see that using judgemental equality instead of conversion affects the metatheory even for simple calculi. In Section 4.1.2 we shall discuss the consequences of the different presentations for more complex calculi with dependent types.

$$(\mathit{Refl})\quad \frac{\Gamma \vdash M : A}{\Gamma \vdash M = M : A} \qquad (\mathit{Sym})\quad \frac{\Gamma \vdash M = N : A}{\Gamma \vdash N = M : A}$$

$$(\mathit{Trans})\quad \frac{\Gamma \vdash M = N : A \quad \Gamma \vdash N = P : A}{\Gamma \vdash M = P : A}$$

$$(\beta)\quad \frac{\Gamma, x{:}A \vdash M : B \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x{:}A.M)(N) = [N/x]M : B}$$

$$(\eta)\quad \frac{\Gamma \vdash M : A \to B}{\Gamma \vdash \lambda x{:}A.M(x) = M : A \to B}$$

$$(\lambda\text{-}\mathit{Eq})\quad \frac{\Gamma, x{:}A \vdash M = N : B}{\Gamma \vdash \lambda x{:}A.M = \lambda x{:}A.N : A \to B}$$

$$(\mathit{App\text{-}Eq})\quad \frac{\Gamma \vdash M = P : A \to B \quad \Gamma \vdash N = Q : A}{\Gamma \vdash M(N) = P(Q) : B}$$

$$(\mathit{Subst\ Eq})\quad \frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M : A \quad \Gamma_0 \vdash P = Q : C}{\Gamma_0, \Gamma_1 \vdash [P/z]M = [Q/z]M : A}$$

$$(\mathit{Eq\ Subst})\quad \frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M = N : A \quad \Gamma_0 \vdash P : C}{\Gamma_0, \Gamma_1 \vdash [P/z]M = [P/z]N : A}$$

Figure 2–2: Judgemental Equality

The derivations for the judgement form $\Gamma \vdash M = N : A$ are defined after the derivations for the judgement form $\Gamma \vdash M : A$. This will allow us to establish the relationship between the two judgement forms easily. Again, this will be more complicated for calculi with dependent types.

We have followed the conventional presentation by identifying abstractions up to the renaming of bound variables. However, for the formal development of type theory we do not view this as an acceptable presentation, and we would prefer to use one of the presentations of de Bruijn indices [21], Stoughton's α-normal forms [74] or Coquand's distinguishing between parameters and bound variables [15,58]. Our choice of presentation does not affect the correctness of our proofs, but instead we do not believe that the identification of terms which are syntactically different is a reasonable way to present a formal system.

We do not present variables with type labels, differing from other presentations of the simply typed lambda calculus [33]. Not having type labels for variables is the usual practice for systems with dependent types, because having labels on variables in these systems would make the term structure more complex: for example, it would no longer be immediate that variables are normalizing. Because we are interested in studying more complex calculi than the simply typed lambda calculus, we use a term structure which generalizes easily.

2.2 λ^S: A Typed Operational Semantics

A typed operational semantics is a system for reasoning about reduction and normal forms for terms which are well-typed in the type theory. In this section we relate our typed operational semantics to the notion of standard reduction for the untyped lambda calculus and we introduce the calculus λ^S, the typed operational semantics for the simply typed lambda calculus.

2.2.1 Standard Reduction

Standard reduction is an important notion in the untyped lambda calculus (see for example Barendregt's book [7]). The idea is that any reduction sequence can be expressed as a standard, or left to right, reduction sequence. Our typed operational semantics for a type theory will be a synthesis of this notion of standard reduction and the rules of inference of the semantic presentation of the type theory. It is because the reduction used in our typed operational semantics is universal with respect to other reductions that the important results about the relationship between typed terms and reduction follow easily by reasoning in the typed operational semantics.

We can see the typed operational semantics as the type theory presented from the perspective of computation instead of from the perspective of logical inference. We still need the full type information to derive the well-typedness of any term, but we replace the logically clear rules for application and abstraction by rules which instead express the reduction behavior of terms in the calculus. Hence we shall not have one single rule for application but several rules, one for applications with no outermost redex, one for outermost β-redices and one for β-redices on the left side of applications. Similarly we shall have two rules for abstraction, depending on whether the normal form of the body yields an abstraction which is an η-redex.

We shall only be able to give our particular typed operational semantics corresponding to standard reduction for a type theory which is strongly normalizing. This is because while standard reduction for the untyped lambda calculus contracts redices from left to right, which means some redices may not be contracted, we instead always contract the leftmost redex until the process terminates. The property that any reduction sequence can be expressed as a standard one is true for our presentation as well, for any reduction sequence to a normal form. This obviously fails for an arbitrary reduction sequence, since the reduction could simply contract a redex which is not the leftmost one.

Another important property of our typed operational semantics is that it is *syntax directed*, in the sense that the structure of a derivation is determined by the structure of the subject. This property of our typed operational semantics will be important in the treatment of type theories with dependent types.

There is a strong relationship in the untyped lambda calculus between weak head reduction or leftmost reduction and ordinary reduction, which Altenkirch [4] shows for weak head reduction on untyped terms: namely, if M weak head reduces to N and M reduces to M' then $M' \equiv N$ or M' weak head reduces to N' and N reduces to N'. For the untyped lambda calculus this property does not extend to standard reduction, which as we have mentioned does not necessarily contract all redices. We can prove this property for the reduction in our system, but it would require a formal presentation of one-step standard reduction, which is not the reduction we need for our normalization proof. However, this property is used implicitly in the proof of the central result stating the relationship between well-typedness and reduction, Lemma 2.3.27.

Because our reduction is always to a normal form, we shall also have a novel treatment of the η-postponement property. We shall be able to include the postponed η-reductions as part of the inductive definition of standard reduction. The simplicity of this definition allows an elegant and concise study of metatheoretic properties for normalizing type systems with η-equalities.

We have used a different notion of canonical form in defining our typed operational semantics than Martin-Löf uses in explaining his type theory. Martin-Löf only requires that the outermost constructor correspond to an introduction rule for the type, as we described in Section 1.1. In our typed operational semantics we have instead required that each of the subterms be in canonical form as well, so canonical forms must be normal forms with no further reductions. Hence $\mathrm{succ}(1+1)$ is not a canonical natural number but $\mathrm{succ}(\mathrm{succ}(\mathrm{succ}(0)))$ is, the latter being an element which we can directly recognize as a natural number.

We have not modeled this usual explanation of canonical forms and equality in our typed operational semantics only for practical reasons. The system we are studying has simple syntactic identity as the equality on normal forms, rather than a more sophisticated equality involving canonical elements. The metatheoretic proofs for the system are easier if we only consider a relation between terms and types and not the added equivalence relation for judgemental equality. In conjunction with the stricter notion of canonical form, we have not allowed a type to have an equality over its canonical elements.

We can imagine combining the typing judgement with some other notion of reduction, corresponding to another evaluation strategy. For example, it seems that the soundness proof in Section 2.4 might be easier if we were to use call-by-value reduction, a notion of reduction which reduces both terms in an application to normal form and then performs a β-reduction if the resulting term is a redex. We can also prove strengthening by considering this reduction. However, we would not be able to prove strong normalization or subject reduction using this reduction, because it is not canonical with respect to all reductions.

2.2.2 Judgements and Derivations

We now define the calculus λ^S. For our choice of presentation, the term structure for the typed operational semantics will be the same as the term structure for the semantic presentation of the system.

The judgement forms for λ^S and their intuitive meanings will be:

- $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$, meaning that M has canonical form P which is a canonical element of type A in context Γ.

- $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$, meaning that M weak head reduces to N of type A in context Γ. This judgement gives us no indication of what the canonical form of N is, so a derivation of this judgement will only make sense in conjunction with a derivation that $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P : A$ for some P.

We shall also use the abbreviation $\Gamma \vdash^S M : A$ for $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ when the normal form P is not important.

Weak head reduction is defined by the rules of inference for the judgement form $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$. Intuitively weak head redices are outermost redices: either a β-redex or a weak head redex on the left of an application. Weak head reduction is an important notion used implicitly in Martin-Löf's type theory, and for example Harper and Pollack [38] define it as a tool for efficient type checking.

In order to define λ^S, we first need to introduce the auxiliary definition of weak head normal form. A term is weak head normal if it has no weak head reductions, but we define it separately from the judgements of λ^S because we need this property as a side condition in order to test which application rule to use. If a term is in weak head normal form then to find its normal form we only do internal reductions, but if it is not then we must first reduce outermost redices in the definition of the typed operational semantics.

Definition 2.2.1 (Base Term) *A term is a* base term *if it is a variable or it is an application $M(N)$ and M is a base term.*

The base terms are simply the terms with a variable at the head of the term.

Definition 2.2.2 (Pre-redex) *We say that a term M is a* pre-redex *if M is an abstraction.*

The pre-redices are terms which when applied to another term yield a redex.

Definition 2.2.3 (Weak Head Normal) *We say that a term is* weak head normal *or* in weak head normal form *if it is a base term or a pre-redex.*

The typed operational semantics λ^S is defined by the rules of inference in Figures 2–3 and 2–4.

$$(S\text{-}Var) \quad \frac{}{\Gamma \vdash^S x \to^{\mathrm{nf}} x : A} \quad \textit{if } x{:}A \in \Gamma \textit{ and } \Gamma \textit{ is a context}$$

$$(S\text{-}\lambda) \quad \frac{\Gamma, x{:}A \vdash^S M \to^{\mathrm{nf}} P : B}{\Gamma \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} \lambda x{:}A.P : A \to B} \quad \textit{if } P \equiv Q(x) \textit{ implies } x \in \mathrm{FV}(Q)$$

$$(S\text{-}\eta) \quad \frac{\Gamma, x{:}A \vdash^S M \to^{\mathrm{nf}} P(x) : B \qquad \Gamma \vdash^S P \to^{\mathrm{nf}} Q : A \to B}{\Gamma \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} Q : A \to B}$$

$$(S\text{-}App) \quad \frac{\begin{array}{c}\Gamma \vdash^S M \to^{\mathrm{nf}} P : A \to B \\ \Gamma \vdash^S N \to^{\mathrm{nf}} Q : A\end{array}}{\Gamma \vdash^S M(N) \to^{\mathrm{nf}} P(Q) : B} \quad \textit{if } M(N) \textit{ is weak head normal}$$

$$(S\text{-}WH) \quad \frac{\Gamma \vdash^S M \to^{\mathrm{wh}} N : A \qquad \Gamma \vdash^S N \to^{\mathrm{nf}} P : A}{\Gamma \vdash^S M \to^{\mathrm{nf}} P : A}$$

Figure 2–3: Canonical Forms

$$(W\text{-}\beta) \quad \frac{\Gamma \vdash^S \lambda x{:}A.M : A \to B \qquad \Gamma \vdash^S N : A}{\Gamma \vdash^S (\lambda x{:}A.M)(N) \to^{\mathrm{wh}} [N/x]M : B}$$

$$(W\text{-}App) \quad \frac{\Gamma \vdash^S M \to^{\mathrm{wh}} P : A \to B \qquad \Gamma \vdash^S N : A}{\Gamma \vdash^S M(N) \to^{\mathrm{wh}} P(N) : B}$$

Figure 2–4: Weak Head Reduction

Notation We shall write $\Gamma \vdash^S J$ for an arbitrary judgement in λ^S.

2.2.3 Discussion

The relations we define in a typed operational semantics are essentially evaluation relations. The intuitive reading of the judgements is that the subject of the judgement reduces to the normal form in the judgement. Because of the way we have chosen the rules of inference, this system is a good one to use in reasoning about the simply typed lambda calculus in general. It is also specifically designed to give a strong induction principle. We shall discuss the rules of inference of λ^S in order to illustrate the basic idea of the system and to explain some choices in the presentation.

First, the variable rule simply says that variables evaluate to themselves: they are already normal. Because we are interested in a typed evaluation relation, we need to know that x occurs in the context.

The rules (S-λ) and (S-η) depend on the normal forms included in the judgement forms. To know the canonical form of an abstraction, we must first know the canonical form of the body of the abstraction. The rule (S-λ) is purely structural: we know that if the body of an abstraction reduces to some normal form, and if that normal form does not create an η-redex, then the normal form of the whole abstraction is the abstraction of the normal form.

There are two premisses in the rule (S-η). The second premiss, requiring that the canonical form of the body have a canonical form in the context Γ, is unnecessary from the point of view of our reduction, although it removes the need for a side condition requiring x to be distinct from the free variables of P. We have deliberately weakened this rule by strengthening the premisses in order to simplify proofs by induction on derivations of λ^S. Strengthening the premisses corresponds to strengthening the inductive hypothesis for such proofs. As a particular example, to show completeness of this rule for the simply typed lambda calculus, in other words to show that if $\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} P : A \rightarrow B$ where $\Gamma, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} P(x) : B$ and $x \notin \mathrm{FV}(P)$ then $\Gamma \vdash \lambda x{:}A.M = P : A \rightarrow B$, we shall need the normal form P to be well-typed in context Γ instead of $\Gamma, x{:}A$. This is exactly what the second premiss gives us.

Furthermore, our development of the metatheory for λ^S will show that the normal form of a term already in normal form (as P must be if $P(x)$ is normal) is that term, and also we shall be able to show strengthening for λ^S. Using these two results, we shall be able to show the admissibility of the more natural η-reduction rule:

$$(S\text{-}\eta') \quad \frac{\Gamma, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} P(x) \ : \ B}{\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} P \ : \ A \rightarrow B} \quad \textit{if } x \notin \mathrm{FV}(P)$$

The rules $(S\text{-}\lambda)$ and $(S\text{-}\eta)$ for abstraction implicitly introduce η-postponement and preserve normality of the right-hand side of the reduction. The rules first fully reduce the body M of $\lambda x{:}A.M$, and by analysis of the resulting normal form either a final η-reduction takes place or the abstraction is already in normal form. This will mean that when we show that our typed operational semantics is adequate for reduction (in other words, that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P \ : \ A$ implies $M \rhd^* P$), we shall further be able to demonstrate that the reduction sequence factors into β-reductions followed by η-reductions.

We have several rules for application in λ^S. The first rule, $(S\text{-}App)$, is the one which is purely structural. This rule says that if $M(N)$ is weak head normal, which means that M is a variable applied to some sequence of terms, then the way to evaluate $M(N)$ is to evaluate M and N separately. We could have no other way of evaluating such terms.

The second rule for application, the rule $(W\text{-}\beta)$, is as follows:

$$(W\text{-}\beta) \quad \frac{\Gamma \vdash^S \lambda x{:}A.M \ : \ A \rightarrow B \qquad \Gamma \vdash^S N \ : \ A}{\Gamma \vdash^S (\lambda x{:}A.M)(N) \rightarrow^{\mathrm{wh}} [N/x]M \ : \ B}$$

Although we have not used the normal forms of either $[x{:}A_1]M_0$ or N, it is important from the point of view of reduction that we know that the two terms are normalizing, and from the point of view of typing that we know that the two terms are well-typed. By using the second judgement form $\Gamma \vdash^S (\lambda x{:}A.M)(N) \rightarrow^{\mathrm{wh}} [N/x]M \ : \ A$, we also require that the reduct $[N/x]M$ of the redex $(\lambda x{:}A.M)(N)$ is well-typed. This is because the only rule incorporating derivations of this judgement into the main judgement $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ is the rule (*S-WH*). This rule ensures that we know that if $\Gamma \vdash^S (\lambda x{:}A.M)(N) \to^{\mathrm{nf}} P : B$ then $\Gamma \vdash^S (\lambda x{:}A.M)(N) \to^{\mathrm{wh}} [N/x]M : B$ and $\Gamma \vdash^S [N/x]M \to^{\mathrm{nf}} P : B$.

We have not used the rule

$$(S'\text{-}\beta) \quad \frac{\Gamma \vdash^S \lambda x{:}A.M : A \to B \qquad \Gamma \vdash^S N : A \qquad \Gamma \vdash^S [N/x]M \to^{\mathrm{nf}} R : B}{\Gamma \vdash^S (\lambda x{:}A.M)(N) \to^{\mathrm{nf}} R : B}$$

This is because the system corresponding to standard reduction cannot be presented using a rule of this form: this β-rule reduces the outermost β-redex but gives no strategy for reducing leftmost redices. Hence this rule would give a different reduction strategy.

The final rule of the system, (*W-App*), allows us to reduce β-redices which are on the left of applications. This corresponds exactly to the usual definition of weak head reduction.

2.3 Metatheory

In this section we prove the essential properties for derivations of judgements in the simply typed lambda calculus. This presentation is influenced by Luo's thesis [44] and developments of Pure Type Systems [8], although the formal system is considerably different. Since we shall eventually show the equivalence of λ^S and the simply typed lambda calculus, all of the results for λ^S will transfer to the simply typed lambda calculus as well.

2.3.1 Basic Definitions

We first introduce several notions for terms which will be important in the presentation of the typed operational semantics. Some of these definitions will be simple for the simply typed case but more complex for systems with dependent types, which we study later.

Definition 2.3.1 (Compatible Closure) *Let R be a relation on terms. Then the* compatible closure *of R, notation $M \rhd_R N$, is the least relation satisfying the following rules:*

$$(R\text{-}Inc) \qquad \frac{M R N}{M \rhd_R N}$$

$$(\xi) \qquad \frac{M \rhd_R N}{\lambda x{:}A.M \rhd_R \lambda x{:}A.N}$$

$$(App\text{-}L) \quad \frac{M \rhd_R P}{M(N) \rhd_R P(N)} \qquad (App\text{-}R) \quad \frac{N \rhd_R P}{M(N) \rhd_R M(P)}$$

Definition 2.3.2 (Untyped Reduction) *We introduce the following one-step reduction relations:*

$$\begin{array}{llcll} (\beta) & (\lambda x{:}A.M)(N) & \beta & [N/x]M & \\ (\eta) & \lambda x{:}A.M(x) & \eta & M & \text{if } x \notin \mathrm{FV}(M) \end{array}$$

Let untyped reduction, *$M \rhd N$, be $M \rhd_{\beta\eta} N$, the compatible closure of (β) and (η).*

A term M is an R-redex if there is some N such that MRN. A term M is a redex *if M is a $\beta\eta$-redex.*

Notation If R is a relation then we denote the transitive closure of R by R^+ and the reflexive and transitive closure of R by R^*.

Reduction plays no role in the definition of either the semantic presentation of the simply typed lambda calculus or the calculus λ^S. It is instead an auxiliary definition which captures our intuition of how untyped computation operates on terms. Theoretically, we shall use it as a convenient tool to help us with the metatheoretic development of the simply typed lambda calculus. Pragmatically, it is much nicer to implement reduction without reference to type information. For both of these points the subject reduction property, which says that the typing judgement is closed under untyped reduction, will be essential.

Definition 2.3.3 (Normal Form) *We define inductively the terms which are* normal *or* in normal form*:*

- *Variables are normal,*
- $\lambda x{:}A.M$ *is normal if* M *is normal and not of the form* $N(x)$ *with* $x \notin \mathrm{FV}(N)$*, and*
- $M(N)$ *is normal if* M *and* N *are normal and* $M(N)$ *is not a redex.*

Lemma 2.3.4 *If* M *is normal then* M *has no reductions.*

Proof By induction on terms in normal form. □

Definition 2.3.5 (Strongly Normalizing) *A term is* strongly normalizing *if all reduction sequences starting from that term terminate.*

Definition 2.3.6 (Diamond Property) *We say that a term* M *satisfies the* diamond property *if whenever* $M \rhd^* N$ *and* $M \rhd^* P$ *then there is a* Q *such that* $N \rhd^* Q$ *and* $P \rhd^* Q$*.*

The following facts about terms are easy to establish:

Lemma 2.3.7

- *If* M *is normal then* M *is weak head normal.*
- *If* $M(N)$ *is weak head normal and* M *and* N *are normal then* $M(N)$ *is normal.*
- *If* M *is weak head normal and* $M \rhd^* N$ *then* N *is weak head normal.*

Proof Straightforward. □

We shall need a more general notion of substitution, that of parallel substitution, at several points in the proof. We introduce the basic definitions and show some simple properties here.

Definition 2.3.8 (Pre-Substitution) *A* pre-substitution *for a finite set of variables S is a function from S to terms.*

Notation Suppose δ is a pre-substitution for $dom(\Gamma)$. Then:

- We write $\widehat{\delta}(M)$ for the result of simultaneously substituting the values for the variables in the domain of δ:

$$\widehat{\delta}(M) =_{\text{df}} [\delta(x_1), \ldots, \delta(x_n)/x_1, \ldots, x_n]M$$

- We write $\delta[x := M]$ for the extended pre-substitution for $dom(\Gamma, x{:}A)$.

- If Δ has all components of Γ then we write $\text{weak}_\Gamma^\Delta$ for the substitution $\text{weak}_\Gamma^\Delta(x) = x$ for $dom(\Gamma)$.

- $\text{id}_\Gamma =_{\text{df}} \text{weak}_\Gamma^\Gamma$.

- If δ is a pre-substitution for Δ and ϕ is a pre-substitution for Φ then the *composition* of ϕ and δ, $\phi \circ \delta$, is

$$\phi \circ \delta(x) =_{\text{df}} \widehat{\phi}(\delta(x))$$

Definition 2.3.9 (Substitution) *A* substitution *δ from Δ to Γ, where Δ and Γ are contexts, is a pre-substitution for $dom(\Gamma)$ such that for each $x{:}A \in \Gamma$ we have $\Delta \vdash^S \delta(x) \to^{\text{nf}} P : A$ for some P.*

This definition of substitution is for substitutions to λ^S. We could also define substitutions to the simply typed lambda calculus, but these will not be necessary in our metatheoretic development.

We do not know in general that if δ is a substitution from Δ to Γ and ϕ is a substitution from Φ to Δ then $\phi \circ \delta$ is a substitution from Φ to Γ. This will only be the consequence of the final soundness theorem. However, we shall know closure under composition for the following restricted class of substitutions.

Definition 2.3.10 (Renaming) *A* renaming *δ from Δ to Γ is a substitution from Δ to Γ such that $\delta(x) = y$, where $y \in V$, for each $x{:}A \in \Gamma$.*

Lemma 2.3.11

- *If δ is a renaming from Δ to Γ and ϕ is a renaming from Φ to Δ then $\phi \circ \delta$ is a renaming from Φ to Γ.*
- *If M is weak head normal and δ is a renaming from Δ to Γ then $\widehat{\delta}(M)$ is weak head normal.*
- $[N/x]M \equiv \widehat{\mathrm{id}_{\mathrm{FV}(M)-\{x\}}[x := N]}(M)$.
- $\widehat{\delta}(\phi[x := N])M \equiv \widehat{\delta \circ \phi[x := \widehat{\delta}(N)]}(M)$.
- $\widehat{\phi \circ \delta}(M) \equiv \widehat{\phi}(\widehat{\delta}(M))$.

2.3.2 Basic Metatheory

In this section we establish the basic metatheoretic properties we need for the two calculi presenting the simply typed lambda calculus. Most of the results will be for the system λ^S, because we shall transfer the results from this system to the semantic presentation by soundness and completeness results.

An essential technique in showing metatheoretic properties is the principle of induction on derivations. Often this principle is taken as natural number induction on the height of derivation trees, which allows us to know the inductive hypothesis for any shorter derivation tree. We shall instead use complete structural induction, which allows us to know the inductive hypothesis for any subderivation of a derivation.

The definition of the system λ^S is a mutual inductive definition. We shall often want to prove different statements for the different judgement forms. For example, in Free Variables (Lemma 2.3.15), we prove that if $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ then $\mathrm{FV}(M) \subseteq \mathit{dom}(\Gamma)$ and $\mathrm{FV}(P) \subseteq \mathit{dom}(\Gamma)$, and that if $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ then $\mathrm{FV}(M) \subseteq \mathit{dom}(\Gamma)$. In this as in many other cases, we only need to prove the result for the term M in the judgement $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$, because the rule (*S-WH*) is the only one to incorporate this judgement into the judgement $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$, and we know the inductive hypothesis for N in this rule.

Lemma 2.3.12 *If* $\Gamma \vdash M = N : A$ *then* $\Gamma \vdash M : A$ *and* $\Gamma \vdash N : A$.

Proof By induction on derivations of $\Gamma \vdash M = N : A$. □

Lemma 2.3.13 (Generation) *Any judgement which is derivable in* λ^S *has a uniquely determined last rule of inference.*

Proof By induction on derivations. □

We have not written the full information we are interested in from the Generation Lemma. We shall in fact want to know for each judgement what the last rule of inference is and what the forms of the premisses are. For example:

- If $\Gamma \vdash^S M(N) \to^{\mathrm{nf}} P : B$ and $M(N)$ is weak head normal then there are Q, R, and A such that $P \equiv Q(R)$, $\Gamma \vdash^S M \to^{\mathrm{nf}} Q : A \to B$ and $\Gamma \vdash^S N \to^{\mathrm{nf}} R : A$.

- If $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ and M is not weak head normal then there is an N such that $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ and $\Gamma \vdash^S N \to^{\mathrm{nf}} P : A$.

- If $\Gamma \vdash^S M(N) \to^{\mathrm{wh}} P : B$ and M is not weak head normal then there are Q, R and A such that $P \equiv Q(N)$, where $\Gamma \vdash^S M \to^{\mathrm{wh}} Q : A \to B$ and $\Gamma \vdash^S N \to^{\mathrm{nf}} R : A$.

- If $\Gamma \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} P : A \to B$ then $\Gamma, x{:}A \vdash^S M \to^{\mathrm{nf}} Q : B$, and either $Q \equiv R(x)$ with $x \notin \mathrm{FV}(R)$, $\Gamma \vdash^S R \to^{\mathrm{nf}} S : A \to B$ and $P \equiv S$, or $P \equiv \lambda x{:}A.Q$.

Lemma 2.3.14 (Uniqueness of Normal Forms) *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *and* $\Gamma \vdash^S M \to^{\mathrm{nf}} Q : B$ *then* $P \equiv Q$ *and* $A = B$.

Proof By induction on derivations, using Generation and the inductive hypothesis in each case for the second derivation. □

This lemma states that the operational semantics is deterministic. In fact we could have showed the stronger result, which we called "syntax directed" above, that there is a unique derivation associated with any context and term. This is equivalent to saying that the operational semantics is deterministic at the level of derivations. However, we do not need to reason about derivations in this way in our study of the metatheory.

We can show that $A = B$ because there is no notion of α-equivalence for types in the simply typed lambda calculus. This is not the case for systems with dependent types.

Lemma 2.3.15 (Free Variables)

- *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *then* $\mathrm{FV}(M) \subseteq dom(\Gamma)$ *and* $\mathrm{FV}(P) \subseteq dom(\Gamma)$.
- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ *then* $\mathrm{FV}(M) \subseteq dom(\Gamma)$.

Proof By induction on derivations. □

Lemma 2.3.16 (Context) *If* $\Gamma \vdash^S J$ *then* Γ *is a context.*

Proof By induction on derivations. □

Lemma 2.3.17 (Completeness)

- *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *then* $\Gamma \vdash M = P : A$.
- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ *then* $\Gamma \vdash M = N : A$.

Proof By induction on derivations.

We only consider (W-β). By Generation we know that there is a subderivation of $\Gamma, x{:}A \vdash^S M \to^{\mathrm{nf}} R : B$ for some R. Hence by inductive hypothesis we know that $\Gamma, x{:}A \vdash M = R : B$ and $\Gamma \vdash N = Q : A$, so $\Gamma, x{:}A \vdash M : B$ and $\Gamma \vdash N : A$ by Lemma 2.3.12. Hence $\Gamma \vdash (\lambda x{:}A.M)(N) = [N/x]M : B$ by (β). $\square$

We shall now show that any judgement for a context Γ also holds in all contexts which have all components of Γ. We follow Barendregt [8] in calling this property "thinning" and reserving the name "weakening" for adding fresh variables at the end of the context.

We first prove a general lemma using a technique inspired by McKinna and Pollack's treatment [58], where we allow names of variables in the context to change. This statement strengthens the inductive hypothesis for the rules which introduce new bound variables, in particular $(S\text{-}\lambda)$. In such cases, if the bound variable already occurs in the new context then a direct induction on derivations fails. The lemma is only true up to changes in bound variables, as can be seen in our illustration of the rule $(S\text{-}\lambda)$.

We shall have the usual Thinning lemma as a corollary.

Lemma 2.3.18 (Renaming) *If δ is a renaming from Δ to Γ then*

- *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ then $\Delta \vdash^S \widehat{\delta}(M) \rightarrow^{\mathrm{nf}} \widehat{\delta}(P) : A$.*
- *If $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ then $\Delta \vdash^S \widehat{\delta}(M) \rightarrow^{\mathrm{wh}} \widehat{\delta}(N) : A$.*

Proof By induction on derivations. We show several cases:

- $(S\text{-}Var)$. We know that $\delta(x) = y$, so $\Delta \vdash^S y \rightarrow^{\mathrm{nf}} P : A$ for some P. By Generation we know that $P \equiv y$.
- $(S\text{-}\lambda)$. Let δ be a renaming from Δ to Γ, y be fresh in Δ and $w =_{\mathrm{df}} \mathrm{weak}_{\Delta}^{\Delta, y:A}$. Then $w \circ \delta$ is a renaming from $\Delta, y{:}A$ to Γ by Lemma 2.3.11 and $(w \circ \delta)[x := y]$ is a renaming from $\Delta, y{:}A$ to $\Gamma, x{:}A$, so by inductive hypothesis

$$\Delta, y{:}A \vdash^S \widehat{(w \circ \delta)[x := y]}(M) \rightarrow^{\mathrm{nf}} \widehat{(w \circ \delta)[x := y]}(P) : B$$

Hence

$$\begin{array}{lll}\Delta & \vdash^S & \widehat{\delta}(\lambda x{:}A.M) \equiv \lambda y{:}A.(w \circ \widehat{\delta)[x := }y](M) \\ & \to^{\mathrm{nf}} & \lambda y{:}A.(w \circ \widehat{\delta)[x :=} y](P) \equiv \widehat{\delta}(\lambda x{:}A.P) \colon A \to B\end{array}$$

Finally, we need to show for all Q that $w \circ \widehat{\delta[x :=} y](P) \equiv Q(y)$ implies that $y \in \mathrm{FV}(Q)$, where we already know for all Q that if $P \equiv Q(x)$ then $x \in \mathrm{FV}(Q)$. Suppose $w \circ \widehat{\delta[x :=} y](P) \equiv Q(y)$. Then there is a z such that $P \equiv P_1(z)$ by the definition of pre-substitutions and renamings. Because y is fresh in Δ and we know for all $z \in \mathit{dom}(\Gamma)$ that $(w \circ \delta)(z) \neq y$, we have $z = x$. Hence $x \in \mathrm{FV}(P_1)$, and $y \in \mathrm{FV}(\widehat{\delta[x := y]}(P_1)) = \mathrm{FV}(Q)$.

- $(S\text{-}\eta)$. We are able to establish the first premiss as for the case $(S\text{-}\lambda)$, where δ is a renaming from Δ to Γ, y is fresh in Δ and $w =_{\mathrm{df}} \mathrm{weak}_{\Delta}^{\Delta,y:A}$. By inductive hypothesis we also know that $\Delta \vdash^S \widehat{\delta}(P) \to^{\mathrm{nf}} \widehat{\delta}(Q) : A \to B$, so by Contexts $x \notin \mathit{dom}(\Gamma)$ and by Free Variables $x \notin \mathrm{FV}(P) \subseteq \mathit{dom}(\Gamma)$. Hence $\widehat{\delta}(P) \equiv (w \circ \widehat{\delta)[x :=} y](P)$, so

$$(w \circ \widehat{\delta[x :=} y](P(x)) \equiv ((w \circ \widehat{\delta[x :=} y](P))(y) \equiv (\widehat{\delta}(P))(y)$$

Hence $\Delta \vdash^S \lambda y{:}A.(w \circ \widehat{\delta)[x :=} y](M) \equiv \widehat{\delta}(\lambda x{:}A.M) \to^{\mathrm{nf}} \widehat{\delta}(Q) : A \to B$.

- $(S\text{-}App)$. By inductive hypothesis and Lemma 2.3.11.

- $(W\text{-}\beta)$. By inductive hypothesis we know that

$$\Delta \vdash^S \lambda y{:}A.\widehat{\delta[x := y]}(M) \equiv \widehat{\delta}(\lambda x{:}A.M) \to^{\mathrm{nf}} \widehat{\delta}(P) : A \to B$$

and $\Delta \vdash^S \widehat{\delta}(N) \to^{\mathrm{nf}} \widehat{\delta}(Q) : A$. Hence, for y fresh in Δ,

$$\begin{array}{lll}\Delta & \vdash^S & \widehat{\delta}((\lambda x{:}A.M)(N)) \equiv (\lambda y{:}A.\widehat{\delta[x := y]}(M))(\widehat{\delta}(N)) \\ & \to^{\mathrm{nf}} & [\widehat{\delta}(N)/y]\widehat{\delta[x := y]}(M) \equiv \widehat{\delta}([N/x]M) \colon B\end{array}$$

$\square$

Lemma 2.3.19 *If Δ is a context with all components of Γ then* $\mathrm{weak}_{\Gamma}^{\Delta}$ *is a substitution from Δ to Γ.*

Proof By induction on the structure of Γ. $\square$

As we mentioned, Renaming has the following corollary:

Corollary 2.3.20 (Thinning) *If* $\Gamma \vdash^S J$ *and* Δ *is a context with all components of* Γ *then* $\Delta \vdash^S J$.

Proof By Renaming and Lemma 2.3.19. $\square$

Strengthening for the simply typed lambda calculus is a straightforward result. However, we shall see later that proving strengthening for the typed operational semantics rather than the semantic presentation makes the proof much easier for systems with dependent types.

Lemma 2.3.21 (Strengthening)

- *If* $\Gamma_0, z{:}C, \Gamma_1 \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *and* $z \notin \mathrm{FV}(M)$ *then* $\Gamma_0, \Gamma_1 \vdash^S M \rightarrow^{\mathrm{nf}} P : A$.

- *If* $\Gamma_0, z{:}C, \Gamma_1 \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ *and* $z \notin \mathrm{FV}(M)$ *then* $\Gamma_0, \Gamma_1 \vdash^S M \rightarrow^{\mathrm{wh}} N : A$.

Proof By induction on derivations.

We check $(S\text{-}\lambda)$. By Contexts we know that $z \neq x$, so by inductive hypothesis $\Gamma_0, \Gamma_1, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} P : B$. Hence $\Gamma_0, \Gamma_1 \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} \lambda x{:}A.P : A \rightarrow B$ by $(S\text{-}\lambda)$. $\square$

Proving a substitution or cut lemma for λ^S directly seems to be difficult. Indeed, substitution on the judgements of λ^S is difficult to define, because we shall need to know the normal form of the substituted term. For example, if we know that $\Gamma, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} P : B$ and $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} Q : A$ then we shall need to know that there is an R such that $\Gamma \vdash^S [N/x]M \rightarrow^{\mathrm{nf}} R : B$, and we need normalization to know what R is.

Substitution is therefore closely related to soundness for the simply typed lambda calculus. By the soundness result, we shall know that if $\Gamma \vdash M : A$ then there is a P such that $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$, so we can establish the above substitution property by the rule (*Subst*). We shall discuss the relationship between substitution and normalization in more detail in Section 2.4.1.

2.3.3 λ^S and Reduction

The calculus λ^S is a synthesis of standard reduction and the typing rules from the simply typed lambda calculus. We are able to use this to considerable advantage in showing results about the relationship between typing in λ^S and untyped reduction.

Lemma 2.3.22 (Adequacy for Reduction)

- *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *then* $M \rhd^* P$ *and* P *is normal, and furthermore there is an* N *such that* $M \rhd^*_\beta N \rhd^*_\eta P$.

- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ *then* $M \rhd_\beta N$.

Proof By induction on derivations. We consider the two interesting cases:

- (S-η). By Contexts $\Gamma \equiv x_1{:}A_1, \ldots, x_n{:}A_n$ and x_1 through x_n are distinct from each other and from x. By Free Variables $\mathrm{FV}(P) \subseteq dom(\Gamma)$. Hence $x \notin \mathrm{FV}(P)$.

  Then by inductive hypothesis $M \rhd^*_\beta N \rhd^*_\eta P(x)$, so $\lambda x{:}A.M \rhd^*_\beta \lambda x{:}A.N \rhd^*_\eta \lambda x{:}A.P(x)$, and furthermore $\lambda x{:}A.P(x) \rhd_\eta P$. Because $P(x)$ is normal P must be normal as well, so by Lemma 2.3.4 has no reductions. Since by inductive hypothesis $P \rhd^* Q$, we know that $P \equiv Q$.

- (S-*App*). By inductive hypothesis both P and Q are normal and $M \rhd^* P$ and $N \rhd^* Q$, so $M(N) \rhd^* P(Q)$, where $P(Q)$ is both weak head normal and normal by Lemma 2.3.7.

$\square$

Corollary 2.3.23 *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *and* M *is normal then* $M \equiv P$.

Proof By Adequacy for Reduction $M \rhd^* P$. Furthermore M has no reductions by Lemma 2.3.4, so $M \equiv P$. □

We now prove a general lemma about the relationship between untyped reduction and the system λ^S. This lemma will imply the important properties of strong normalization, subject reduction and the diamond property.

Definition 2.3.24 *The predicate* $\mathcal{S}_\Gamma^{P,A}(M)$ *is defined as the least relation such that* $\mathcal{S}_\Gamma^{P,A}(M)$ *holds if:*

- $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$,
- *for all* N *which are subterms of* M *there are* B *and* Q *such that* $\Gamma, \Delta \vdash^S N \to^{\mathrm{nf}} Q : B$, *where* Δ *is the sequence of binders which* N *occurs under in* M, *and*
- $M \rhd N$ *implies* $\mathcal{S}_\Gamma^{P,A}(N)$ *for all* N.

$\mathcal{C}_\Gamma^{P,A}(M)$ *will be the same as* $\mathcal{S}_\Gamma^{P,A}(M)$ *except that in the last clause instead of* $M \rhd N$ *we use* $M \rhd^+ N$.

This definition is closely related to the definition of strong normalization in Altenkirch's thesis [4], although we have added type information. We have used this definition because of the similarity of the proofs of subject reduction, strong normalization and the diamond property in this setting. The predicate is well-defined because it is strictly positive. We shall also have the following induction principle on this predicate: if $\mathcal{S}_\Gamma^{P,A}(M)$, for all N such that $M \rhd N$ we have $\mathcal{P}(\mathcal{N})$, and for all N which are subterms of M we have $\mathcal{P}(\mathcal{N})$, then $\mathcal{P}(\mathcal{M})$ for all M such that $\mathcal{S}_\Gamma^{P,A}(M)$.

Lemma 2.3.25 *The following properties hold for* $\mathcal{S}_\Gamma^{P,A}(M)$*:*

- *If* $\mathcal{S}_\Gamma^{P,A}(M)$ *then* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$.

- *If* $\mathcal{S}_{\Gamma}^{P,A}(M)$ *then* M *is strongly normalizing.*

- *If* $\mathcal{S}_{\Gamma}^{P,A}(M)$ *and* $M \rhd^* N$ *then* $\mathcal{S}_{\Gamma}^{P,A}(N)$*.*

- $\mathcal{S}_{\Gamma}^{P,A}(M)$ *iff* $\mathcal{C}_{\Gamma}^{P,A}(M)$*.*

- *If* $\mathcal{S}_{\Gamma}^{P,A}(M)$ *and* N *is a subterm of* M *then there are* Q *and* B *such that* $\mathcal{S}_{\Gamma}^{Q,B}(N)$*.*

- *If* z *is a variable such that* $z \notin \mathrm{FV}(\Gamma_1)$*,* $\mathcal{S}^{B}_{\Gamma_0,x:C,\Gamma_1}(A)$ *and* $z \notin \mathrm{FV}(A)$ *then* $\mathcal{S}^{P,A}_{\Gamma_0,\Gamma_1}(M)$*.*

Proof By induction on proofs of $\mathcal{S}_{\Gamma}^{P,A}(M)$, using Strengthening for the last property. □

We need one final auxiliary lemma about η-reduction.

Lemma 2.3.26 (Subject Reduction for (η)) *If* $\Gamma \vdash^S \lambda x{:}A.M(x) \rightarrow^{\mathrm{nf}} P : A \to B$ *and* $x \notin \mathrm{FV}(M)$ *then* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A \to B$*.*

Proof By Generation we know that $\Gamma, x{:}A \vdash^S M(x) \rightarrow^{\mathrm{nf}} N : B$, and either $N \equiv P(x)$, $x \notin \mathrm{FV}(P)$ and $\Gamma \vdash^S P \rightarrow^{\mathrm{nf}} P : A \to B$ (by Adequacy for Reduction and Corollary 2.3.23) or $P \equiv \lambda x{:}A.N$ and $N \equiv Q(x)$ implies $x \in \mathrm{FV}(Q)$. By induction on the proof that $\Gamma, x{:}A \vdash^S M(x) \rightarrow^{\mathrm{nf}} N : B$ we show that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A \to B$:

- (S-App). Then, because $M(x)$ is weak head normal, we know by Generation that $\Gamma, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} Q : C \to B$, $\Gamma, x{:}A \vdash^S x \rightarrow^{\mathrm{nf}} x : C$ and $N \equiv Q(x)$. By Generation again $A = C$, and since $x \notin \mathrm{FV}(M)$ we know by Strengthening that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} Q : A \to B$, so $x \notin \mathrm{FV}(Q)$ implies that $P \equiv Q$, because $P(x) \equiv N \equiv Q(x)$.

- (W-β). Then $\Gamma, x{:}A \vdash^S \lambda y{:}C.Q \rightarrow^{\mathrm{nf}} R : C \to B$, $\Gamma, x{:}A \vdash^S x \rightarrow^{\mathrm{nf}} x : C$ and $\Gamma, x{:}A \vdash^S [x/y]Q \rightarrow^{\mathrm{nf}} N : B$. By Generation $A = C$, so $\Gamma \vdash^S \lambda y{:}A.Q \equiv \lambda x{:}A.[x/y]Q \rightarrow^{\mathrm{nf}} P : A \to B$ by either (S-η) or (S-λ).

- (W-App). By inductive hypothesis, Strengthening and (S-WH).

□

Lemma 2.3.27 (Reduction)

- *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *then* $\mathcal{S}_\Gamma^{P,A}(M)$.
- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ *and* $\mathcal{S}_\Gamma^{P,A}(N)$ *then* $\mathcal{S}_\Gamma^{P,A}(M)$.

Proof By induction on derivations. We show two cases:

- (S-λ). We need to show that:
  - $\Gamma \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} \lambda x{:}A.P : A \to B$,
  - there are C and Q such that $\mathcal{S}_{\Gamma,\Delta}^{Q,C}(N)$ for all subterms N of M, and
  - $\lambda x{:}A.M \rhd N$ implies $\mathcal{S}_\Gamma^{P,A\to B}(N)$.

  The first follows directly and the second follows by inductive hypothesis.

  For the last, by induction on $\mathcal{S}_{\Gamma,x:A}^{P,B}(M)$ we know that $M \rhd N$ implies that $\mathcal{S}_{\Gamma,x:A}^{P,B}(N)$ for all N. Hence, suppose $\lambda x{:}A.M \rhd N$. Then:

  - $M \rhd N$ implies that $\lambda x{:}A.M \rhd \lambda x{:}A.N$. By inductive hypothesis $\mathcal{S}_\Gamma^{\lambda x:A.P,A\to B}(\lambda x{:}A.N)$.
  - $\lambda x{:}A.M(x) \rhd M$ by (η), where $x \notin \mathrm{FV}(M)$. Since M is a subterm of $\lambda x{:}A.M(x)$ we know that $\mathcal{S}_{\Gamma,x:A}^{Q,C}(M)$ for some Q and C, and by the Strengthening component of Lemma 2.3.25 we know that $\mathcal{S}_\Gamma^{Q,C}(M)$. Furthermore, by Lemma 2.3.26 we know that $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A \to B$, so by Uniqueness of Normal Forms $\mathcal{S}_\Gamma^{P,A\to B}(M)$.

- (W-β). We know that $\mathcal{S}_\Gamma^{P,A\to B}(\lambda x{:}A.M)$ and $\mathcal{S}_\Gamma^{Q,A}(N)$ by inductive hypothesis. Also $\mathcal{S}_\Gamma^{R,B}([N/x]M)$ by assumption. Hence $\mathcal{C}_\Gamma^{R,B}([N/x]M)$ by Lemma 2.3.25.

The result follows again by induction on the normalization premisses, where if $\lambda x{:}A.M(x) \rhd M$ by (η) with $x \notin \mathrm{FV}(M)$ then $M(N) \equiv [N/x]M(x)$, and $\mathcal{S}_\Gamma^{R,B}([N/x]M(x))$ by assumption.

$\square$

Without η-equalities this presentation would be somewhat simpler. In particular, we would not need Lemma 2.3.26 or the condition for subterms in the definition of $\mathcal{S}_\Gamma^{P,A}(M)$. Lemma 2.3.26 deals with the extra cases we need to verify for the diamond property.

However, the important point is that we have a proof of the diamond property for a system with $\beta\eta$-equality. It is well-known that this property with $\beta\eta$-reduction on non-well-typed terms with type labels fails. The counterexample is that the term $\lambda x{:}A.(\lambda y{:}B.M)(x)$ reduces to either $\lambda x{:}A.[x/y]M$ by (β) or $\lambda y{:}B.M$ by (η), and if A and B are not the same then there is no common reduct for them. On the other hand, this counterexample does not hold for well-typed terms, because by the rules of inference the type labels A and B must be identical for any well-typed term.

It seems that subject reduction cannot be proved straightforwardly by induction on derivations in λ^S, because of the transitive closure in the diagram discussed in Section 2.2.1, which arises from the rule (W-β). A direct proof of subject reduction would have to be proved for the transitive closure of reduction, which is more difficult than one-step reduction. However, strong normalization can in any case be proved directly in the current presentation.

All of the important results about reduction are corollaries to Lemmas 2.3.25, Adequacy for Reduction and Reduction.

Corollary 2.3.28 (Strong Normalization for λ^S) *If $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ then M is strongly normalizing.*

Corollary 2.3.29 (Subject Reduction for λ^S) *If $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ and $M \rhd^* N$ then $\Gamma \vdash^S N \to^{\mathrm{nf}} P : A$.*

Corollary 2.3.30 (Diamond Property for λ^S) *If* $\Gamma \vdash^S M \rightarrow^{\text{nf}} Q : A$*,* $M \rhd^* N$ *and* $M \rhd^* P$ *then* $N \rhd^* Q$ *and* $P \rhd^* Q$*.*

Finally, we show the admissibility of several rules which will be important in showing the soundness of λ^S for the simply typed lambda calculus:

Proposition 2.3.31 (Admissibility of (S-η')) *If* $\Gamma, x{:}A \vdash^S M \rightarrow^{\text{nf}} P(x) : B$ *and* $x \notin \text{FV}(P)$ *then* $\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\text{nf}} P : A \rightarrow B$*.*

Proof By Adequacy for Reduction and Reduction we know that $\Gamma, x{:}A \vdash^S P(x) \rightarrow^{\text{nf}} P(x) : B$ and $P(x)$ is normal. Hence the only rule which applies is (S-App), so by Generation there is a C such that $\Gamma, x{:}A \vdash^S P \rightarrow^{\text{nf}} Q : C \rightarrow B$ and $\Gamma, x{:}A \vdash^S x \rightarrow^{\text{nf}} x : C$. By Generation for the variable we know that $A = C$. We also know by Corollary 2.3.23 that $P \equiv Q$, since P must be normal by definition. Finally, by Strengthening $\Gamma \vdash^S P \rightarrow^{\text{nf}} P : A \rightarrow B$ and $\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\text{nf}} P : A \rightarrow B$ by (S-η). □

We shall also use the lemmas Adequacy for Reduction and Subject Reduction in an essential way in the proof of soundness for the equality rules of the simply typed lambda calculus.

2.4 Soundness

In this section we shall prove the fundamental result of soundness, which relates the simply typed lambda calculus to its typed operational semantics λ^S. Specifically, we show that if $\Gamma \vdash M : A$ then there is a P such that $\Gamma \vdash^S M \rightarrow^{\text{nf}} P : A$. As we have mentioned before, the soundness result is crucial to our treatment of the metatheory of the simply typed lambda calculus. It is through soundness and completeness that we transfer the metatheoretic development of the typed operational semantics to the simply typed lambda calculus.

An important first step in a proof of normalization is to elicit the reduction behavior of the system being studied. Our typed operational semantics serves this purpose: the rules of inference show that the well-typed terms are exactly those constructed with variables and abstraction and a series of weak head expansions. The technique we use to show soundness is closely related to existing proofs of normalization because these proofs do construct such reduction sequences, although this is not always clear. Furthermore, the reasoning used to show strong normalization is applicable to other important results about reduction as well, for example the diamond property as we have seen above, and the rules of inference for the typed operational semantics provide a general structure for showing such results.

2.4.1 Normalization Proofs

There are several techniques for proving normalization for the simply typed lambda calculus. One proof [33] interprets types as sets of well-typed terms and shows that any term is in the interpretation of its type relative to any parallel substitution fulfilling certain conditions.

Although this proof does not discuss variables and contexts in detail, it seems that it relies on an infinite number of variables of each type. The problem is that in the proof we shall often need a variable of some certain type, and it is simplest to assume that this variable always exists in an infinite context. However, this technique seems cumbersome to use for proving soundness of the calculus λ^S, because we are interested in knowing that if $\Gamma \vdash M : A$ then $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ in the same context Γ. If we were instead proving simple normalization this would not be a problem, because untyped reduction is independent of contexts.

Therefore, our proof instead follows Coquand and Gallier's [17] and uses finite contexts and the general notion of substitutions. A substitution from Δ to Γ is an assignment of a term well-typed in Δ to each variable in Γ such that the types are preserved, and the application of a substitution to a term or kind is the parallel substitution of the terms for the variables. This is our Definition 2.3.9. The general soundness result we shall prove says that if a term is well-typed in the simply typed lambda calculus then that term is well-typed in λ^S, relative to all substitutions. This result can also be shown by first showing the soundness of the Kripke interpretation of the simply typed lambda calculus, so we shall call this proof method a Kripke-style model.

The generalization of simple substitution is necessary because of the behavior of β-reduction. For example, for the term $(\lambda x{:}A.\lambda y{:}B.M)(N,P)$ in context Γ, we need to know that M, $[N/x]M$ and $[P/y][N/x]M$ are all normalizing. Intuitively, these three cases are covered by three substitutions to the context $\Gamma, x{:}A, y{:}B$.

For the treatment of abstractions, which are binders of variables, it is important that we can pass from a substitution from Δ to Γ to another from $\Delta, x{:}A$ to Γ for any x fresh in Δ and any type A. This is simply a property of substitutions, because there is always such a weakening of contexts corresponding to the identity substitution on variables. We need this weakening because we can only know that $\Delta \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} P : A \to B$ if $\Delta, x{:}A \vdash^S M \to^{\mathrm{nf}} N : B$, and we shall need to establish this by the inductive hypothesis for the rule (λ) in the semantic presentation.

It is the lack of the general notion of substitution that limits an early presentation of Martin-Löf's type theory [53]. An essential point of his proof of normalization for this system is that the reduction is very restrictive: there is no equality or reduction allowed beneath binders, unlike our rules (λ-*Eq*) in the judgemental equality or (ξ) in untyped reduction. The problem is that his proof is for normalization in the empty context, and there may be no terms of the appropriate type to allow descent beneath the binder in the empty context.

In fact, the need for either infinite contexts or Kripke-style models in normalization proofs seems to relate to the need to prove the general substitution lemma as part of the larger proof. This is related to our inability to prove substitution directly as a property for λ^S, as we discussed in Section 2.3.2. Each of the proof techniques mentioned allows us to choose a fresh free variable and hence to descend beneath binders such as abstractions. This suggests that the substitution lemma will be a corollary to our soundness proof, and indeed the proof of soundness for the system with substitution rules is only a minor modification of the proof for the system without these rules.

Interestingly, the proof of the substitution property embedded in the Kripke-style normalization proof is different from the standard proof of substitution for Pure Type Systems. Omitting the parts of the proof which deal with normalization, we are left with the statement that if ϕ is a substitution from Δ to Γ such that $\delta \circ \phi$ is a substitution from Δ' to Γ for all renamings δ from Δ' to Δ, and if $\Gamma \vdash J$, then $\Delta \vdash \widehat{\phi}(J)$. We can show this directly by induction on derivations in the presentation of the simply typed lambda calculus without rules of inference for thinning or substitution. Furthermore, we do not need to prove the thinning or renaming lemma before showing this, so we can prove this lemma once by induction on derivations, and derive renaming and substitution as corollaries.

Another technique for showing normalization for simple type systems uses terms which need not be well-typed. Gallier [28] discusses the proof of normalization for System F using untyped interpretations, following Tait [77] and Mitchell [61]. However, in our proof we are interested in soundness for the formal system λ^S instead of simple normalization, so by definition we need to construct a model with well-typed terms. In our view, the improvements this approach leads to in the study of the metatheory, as developed in Section 2.3, justify the added complexity of a full soundness proof.

We have mentioned that our proof will be a Kripke-style model construction. Catarina Coquand [13] gives a formal proof of normalization for the simply typed lambda calculus with explicit substitutions, that is substitutions which are in the syntax of the system, via a Kripke model. We feel that our normalization proof could be improved by first proving a soundness result for a more general class of models such as Kripke models and then showing that our semantic objects satisfy these more general conditions.

2.4.2 The Interpretation

We shall prove soundness by interpreting types as sets of terms well-typed in λ^S in a given context and then showing that well-typed terms in the simply typed lambda calculus are in the interpretation of their type.

Definition 2.4.1 (Semantic Object) *A* semantic object *for Δ and A is a term M such that $\Delta \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ for some term P.*

We emphasize that semantic objects are always normalizing. Therefore, the fact that an element of the interpretation or of a saturated set is normalizing will require no proof, unlike many proofs of normalization [33,62,76].

Definition 2.4.2 (Interpretation) *Let Δ be a context. The interpretation of a type A in Δ, $[\![A]\!]\Delta$, is given by induction on the structure of types:*

- $[\![o]\!]\Delta =_{\mathrm{df}} \{M \mid M$ *is a semantic object for* Δ *and* $o\}$.
- $[\![A \rightarrow B]\!]\Delta$ *is the set of semantic objects M for Δ and $A \rightarrow B$ such that for any renaming δ from Δ' to Δ and any $N \in [\![A]\!]\Delta'$ we know that $\widehat{\delta}(M)(N) \in [\![B]\!]\Delta'$.*

Tait's original proof of normalization [76] gives the following interpretations of types:

- $[\![o]\!] =_{\mathrm{df}} \{M \mid M$ is strongly normalizing$\}$.
- $[\![A \rightarrow B]\!] =_{\mathrm{df}} \{M \mid M(N) \in [\![B]\!]$ for all $N \in [\![A]\!]\}$.

Tait's proof shows that terms are normalizing by induction on the structure of types. If M is a term of functional type, he applies it to a constant N. Since by inductive hypothesis the application $M(N)$ is normalizing, M must be normalizing as well.

Martin-Löf [52] instead requires a term to be normalizing as a condition for being in the interpretation. The important difference is that Martin-Löf proves normalization of abstractions by descending beneath the binder. It is clear that this technique is structural, and it is therefore easier to use and more general. Indeed, this seems completely natural in our context of showing a soundness result: we immediately consider the semantic objects to be terms which are well-typed in the typed operational semantics, and the natural way to know that an abstraction is well-typed is to consider typing the body of the abstraction in an extended context.

Using this structural technique makes our soundness proof simpler. We shall also see in Section 2.5 that the rules of inference we have developed for λ^S are more natural than the rules of inference which result from considering Tait's original proof.

Catarina Coquand [13] interprets the base type o at a world w as a family of objects indexed by worlds $w' \geq w$; and $A \to B$ at a world w as a family of functions, again indexed by worlds $w' \geq w$, from the interpretation of A at w' to the interpretation of B at w'. Although in our model we could follow this practice of indexing the objects by worlds, which for us would be contexts, in fact the indexing operation is simply the application of a renaming, as we see above.

Definition 2.4.3 (Interpretation of Contexts) *The interpretation of a context* Γ, $[\![\Gamma]\!]\Delta$, *is a set of substitutions from* Δ *to* Γ *defined by induction on the structure of* Γ*:*

- $[\![()]\!]\Delta =_{\mathrm{df}} \{\epsilon\}$.
- $[\![\Gamma, x{:}A]\!]\Delta =_{\mathrm{df}} \{\rho[x := M] \mid \rho \in [\![\Gamma]\!]\Delta$ *and* $M \in [\![A]\!]\Delta\}$.

2.4.3 Properties of the Interpretation

In this section we show the basic properties of the interpretation of terms and contexts.

Definition 2.4.4 (Saturated Set) *A set S of semantic objects for Δ and A is a* saturated set *for Δ and A if:*

(S1) *If M is a base term then $M \in S$.*

(S2) *If $N \in S$ and $\Delta \vdash^S M \to^{\mathrm{wh}} N : A$ then $M \in S$.*

The reason we need to show that base terms are always in the interpretation of a type is that we shall need to extend a substitution underneath a binder, which we can always do by adding a new variable to the context. We shall want this variable to be in the interpretation of its type.

We emphasize again that the definition of semantic object enforces the condition that elements of saturated sets are normalizing. We therefore do not have the usual condition that if $M \in S$ then M is strongly normalizing as an explicit condition on saturated sets.

Lemma 2.4.5 (Saturated Set) *$[\![A]\!]\Delta$ is a saturated set for any context Δ and type A.*

Proof By induction on the structure of A:

- $A = o$. By the definition of semantic objects and (*S-WH*).

- $A = B \to C$. We need to show conditions *(S1)* and *(S2)*:

  (S1) Suppose M is a base term such that $\Delta \vdash^S M \to^{\mathrm{nf}} P : B \to C$, and let δ be a renaming from Δ' to Δ and $N \in [\![B]\!]\Delta'$ such that $\Delta' \vdash^S N \to^{\mathrm{nf}} Q : B$. By Renaming $\Delta' \vdash^S \widehat{\delta}(M) \to^{\mathrm{nf}} \widehat{\delta}(P) : B \to C$, and $\widehat{\delta}(M)(N)$ is a base term and weak head normal, so $\Delta' \vdash^S \widehat{\delta}(M)(N) \to^{\mathrm{nf}} \widehat{\delta}(P)(Q) : C$ by (*S-App*). Hence $\widehat{\delta}(M)(N) \in [\![C]\!]\Delta'$ by inductive hypothesis and *(S1)*, so $M \in [\![B \to C]\!]\Delta$.

  (S2) Suppose $\Delta \vdash^S M \to^{\mathrm{wh}} N : B \to C$ and $N \in [\![B \to C]\!]\Delta$, and let δ be a renaming from Δ' to Δ and $P \in [\![B]\!]\Delta'$ such that $\Delta \vdash^S P \to^{\mathrm{nf}} R : B$. We know that $\Delta' \vdash^S \widehat{\delta}(M) \to^{\mathrm{wh}} \widehat{\delta}(N) : B \to C$ by Renaming, so $\Delta' \vdash^S \widehat{\delta}(M)(P) \to^{\mathrm{wh}} \widehat{\delta}(N)(P) : C$ by (*W-App*). Finally, $\widehat{\delta}(N)(P) \in [\![C]\!]\Delta'$ by definition, so $\widehat{\delta}(M)(P) \in [\![C]\!]\Delta'$ by inductive hypothesis and *(S2)*. Hence $M \in [\![B \to C]\!]\Delta$.

$\square$

Lemma 2.4.6 (Monotonicity) *If* $M \in [\![A]\!]\Delta$ *and* δ *is a renaming from* Δ' *to* Δ *then* $\widehat{\delta}(M) \in [\![A]\!]\Delta'$.

Proof By induction on the structure of A:

- $A = o$. By Renaming $\widehat{\delta}(M)$ is a semantic object for Δ' and A.
- $A = B \to C$. We again know by Renaming that $\widehat{\delta}(M)$ is a semantic object for Δ' and $B \to C$. Furthermore, suppose δ' is a renaming from Δ'' to Δ' and $N \in [\![B]\!]\Delta''$. Then by definition

$$\widehat{\delta'}(\widehat{\delta}(M))(N) \equiv \widehat{\delta' \circ \delta}(M)(N) \in [\![C]\!]\Delta''$$

□

Lemma 2.4.7 (Monotonicity for Contexts) *If* $\rho \in [\![\Gamma]\!]\Delta$ *and* δ *is a renaming from* Δ' *to* Δ *then* $\delta \circ \rho \in [\![\Gamma]\!]\Delta'$.

Proof By induction on the structure of Γ, using Lemma 2.4.6. □

2.4.4 Soundness

Finally, we can show the soundness result.

Lemma 2.4.8 (Soundness) *If* $\Gamma \vdash M : A$ *and* $\rho \in [\![\Gamma]\!]\Delta$ *then* $\widehat{\rho}(M) \in [\![A]\!]\Delta$.

Proof By induction on derivations of $\Gamma \vdash M : A$:

- (Var). By definition $\rho(x) \in [\![A]\!]\Delta$.
- $(Thin)$. Suppose $\rho \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ and $z \notin \mathrm{FV}(M)$. Then $\rho = \rho_0[z := N]\rho_1$ for some $\rho_0 \in [\![\Gamma_0]\!]\Delta$, $N \in [\![C]\!]\Delta$ and $\rho_1 \in [\![\Gamma_1]\!]\Delta$, so $\rho_0\rho_1 \in [\![\Gamma_0, \Gamma_1]\!]\Delta$. Hence $\widehat{\rho_0\rho_1}(M) \in [\![A]\!]\Delta$ by inductive hypothesis, and $\widehat{\rho_0[z := N]\rho_1}(M) \equiv \widehat{\rho_0\rho_1}(M)$ because $z \notin \mathrm{FV}(M)$.

- (λ). By inductive hypothesis we know that $\widehat{\rho'}(M) \in [\![B]\!]\Delta'$ for any $\rho' \in [\![\Gamma, x{:}A]\!]\Delta'$.

  We first need to show that $\lambda x{:}A.M$ is a semantic object for Δ and $A \to B$, in other words that there is a Q such that $\Delta \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} Q : A \to B$. Let $\rho \in [\![\Gamma]\!]\Delta$ and y be fresh in Δ. Then $\mathrm{weak}_{\Delta}^{\Delta,y:A}$ is a renaming from $\Delta, y{:}A$ to Δ, and $y \in [\![A]\!]\Delta, y{:}A$ by Lemma 2.4.5 and *(S1)*, so $\mathrm{weak}_{\Delta}^{\Delta,y:A} \circ \rho \in [\![\Gamma]\!]\Delta, y{:}A$ by Lemma 2.4.7 and

  $$(\mathrm{weak}_{\Delta}^{\Delta,y:A} \circ \rho)[x := y] \in [\![\Gamma, x{:}A]\!]\Delta, y{:}A$$

  by definition. We therefore know that

  $$\widehat{(\mathrm{weak}_{\Delta}^{\Delta,y:A} \circ \rho)[x := y]}(M) \in [\![B]\!]\Delta, y{:}A$$

  which implies that $\Delta, y{:}A \vdash^S \widehat{\rho[x := y]}(M) \to^{\mathrm{nf}} P : B$ for some P. Hence if $P \equiv P'(y)$ with $y \notin \mathrm{FV}(P')$ then $\Delta \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} P' : A \to B$ by $(S\text{-}\eta')$, and otherwise $\Delta \vdash^S \widehat{\rho}(\lambda x{:}A.M) \equiv \widehat{\rho}(\lambda y{:}A.[y/x]M) \to^{\mathrm{nf}} \lambda y{:}A.P : A \to B$ by $(S\text{-}\lambda)$. Let Q be the normal form in either case.

  We then need to show that $\widehat{\delta}(\lambda x{:}A.M)(N) \in [\![B]\!]\Delta'$ for any renaming δ from Δ' to Δ and any $N \in [\![A]\!]\Delta'$. We can show by reasoning similar to the above that $(\delta \circ \rho)[x := N] \in [\![\Gamma, x{:}A]\!]\Delta'$. This means that $\widehat{(\delta \circ \rho)[x := N]}(M) \in [\![B]\!]\Delta'$, so $\Delta' \vdash^S \widehat{(\delta \circ \rho)[x := N]}(M) \to^{\mathrm{nf}} R : B$ for some R. Furthermore, we know that $\Delta' \vdash^S \widehat{\delta \circ \rho}(\lambda x{:}A.M) \to^{\mathrm{nf}} \widehat{\delta}(Q) : A \to B$ by Renaming, and also $\Delta' \vdash^S N \to^{\mathrm{nf}} S : A$ for some S by definition of semantic objects, so

  $$\begin{aligned} \Delta' \quad \vdash^S \quad & \widehat{\delta \circ \rho}(\lambda x{:}A.M)(N) \equiv (\lambda z{:}A.\widehat{\delta \circ \rho[x := z]}(M))(N) \\ \to^{\mathrm{wh}} \quad & [N/z]((\widehat{\delta \circ \rho)[x := z]}(M)) \equiv \widehat{(\delta \circ \rho)[x := N]}(M) : B \end{aligned}$$

  by $(W\text{-}\beta)$ for any z fresh in Δ'. Hence $\widehat{\delta \circ \rho}(\lambda x{:}A.M)(N) \in [\![B]\!]\Delta'$ by *(S2)*, so $\widehat{\rho}(\lambda x{:}A.M) \in [\![A \to B]\!]\Delta$ by definition.

- (App). By inductive hypothesis we know that $\widehat{\rho}(M) \in [\![A \to B]\!]\Delta$ and $\widehat{\rho}(N) \in [\![A]\!]\Delta$. Hence by definition of $[\![A \to B]\!]\Delta$

  $$\widehat{\mathrm{id}_{\Delta}}(\widehat{\rho}(M))(\widehat{\rho}(N)) \equiv \widehat{\rho}(M(N)) \in [\![B]\!]\Delta$$

- (*Subst*). Suppose $\rho \in [\![\Gamma_0, \Gamma_1]\!]\Delta$. Then we know that $\rho = \rho_0\rho_1$ where $\rho_0 \in [\![\Gamma_0]\!]\Delta$ and $\rho_1 \in [\![\Gamma_1]\!]\Delta$. Hence by inductive hypothesis $\widehat{\rho_0}(N) \in [\![C]\!]\Delta$, so $\rho_0[z := \widehat{\rho_0}(N)]\rho_1 \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$. Finally, by inductive hypothesis again $\widehat{\rho_0[z := \widehat{\rho_0}(N)]\rho_1}(M) \in [\![A]\!]\Delta$, and

$$\widehat{\rho_0[z := \widehat{\rho_0}(N)]\rho_1}(M) \equiv \widehat{\rho_0[z := \widehat{\rho_0\rho_1}(N)]\rho_1}(M) \equiv \widehat{\rho_0\rho_1}([N/z]M)$$

□

Lemma 2.4.9 *If* Γ *is a context then* $\mathrm{id}_\Gamma \in [\![\Gamma]\!]\Gamma$.

Proof By induction on the structure of Γ:

- $\Gamma \equiv ()$. Immediate.

- $\Gamma \equiv \Gamma_0, x{:}A$. By inductive hypothesis $\mathrm{id}_{\Gamma_0} \in [\![\Gamma_0]\!]\Gamma_0$, and $\mathrm{weak}^{\Gamma}_{\Gamma_0} \circ \mathrm{id}_{\Gamma_0} \in [\![\Gamma_0]\!]\Gamma$ by Lemma 2.4.7. Finally

$$(\mathrm{weak}^{\Gamma}_{\Gamma_0} \circ \mathrm{id}_{\Gamma_0})[x := x] \in [\![\Gamma]\!]\Gamma$$

  by Lemma 2.4.5, *(S1)* and the definition of the interpretation of contexts.

□

Corollary 2.4.10 *If* $\Gamma \vdash M : A$ *then there is a* P *such that* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$.

Proof By Soundness, Lemma 2.4.9 and the definition of $[\![A]\!]\Delta$. □

Lemma 2.4.11 (Soundness for $\Gamma \vdash M = N : A$) *If* $\Gamma \vdash M = N : A$ *then there is a* P *such that* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *and* $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P : A$.

Proof By induction on derivations that $\Gamma \vdash M = N : A$. We give several interesting cases:

- (*Trans*). By inductive hypothesis and Uniqueness of Normal Forms.

- (β). By Soundness we know that $\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} P : A \rightarrow B$, $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} Q : A$ and $\Gamma \vdash^S [N/x]M \rightarrow^{\mathrm{nf}} R : B$. Hence $\Gamma \vdash^S (\lambda x{:}A.M)(N) \rightarrow^{\mathrm{nf}} R : B$ by (W-β) and (S-WH).

- (η). By Soundness we know that $\Gamma \vdash^S \lambda x{:}A.M(x) \rightarrow^{\mathrm{nf}} P : A \rightarrow B$, so $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A \rightarrow B$ by Lemma 2.3.26.

- (*App-Eq*). By inductive hypothesis we know that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} R : A \rightarrow B$, $\Gamma \vdash^S P \rightarrow^{\mathrm{nf}} R : A \rightarrow B$, $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} S : A$ and $\Gamma \vdash^S Q \rightarrow^{\mathrm{nf}} S : A$. By Soundness we know that $\Gamma \vdash^S M(N) \rightarrow^{\mathrm{nf}} T : B$ and $\Gamma \vdash^S P(Q) \rightarrow^{\mathrm{nf}} U : B$. Hence by Adequacy for Reduction we know that $M(N) \rhd^* R(S)$ and $P(Q) \rhd^* R(S)$, and by Subject Reduction we have $\Gamma \vdash^S R(S) \rightarrow^{\mathrm{nf}} T : B$ and $\Gamma \vdash^S R(S) \rightarrow^{\mathrm{nf}} U : B$. Hence $T \equiv U$ by Uniqueness of Normal Forms.

$\square$

2.4.5 Consequences of Soundness

We now transfer the metatheoretic study of λ^S in Section 2.3 to the simply typed lambda calculus by using Corollary 2.4.10 and Completeness.

Corollary 2.4.12 (Strong Normalization) *If $\Gamma \vdash M : A$ then M is strongly normalizing.*

Corollary 2.4.13 (Subject Reduction) *If $\Gamma \vdash M : A$ and $M \rhd N$ then $\Gamma \vdash N : A$.*

Corollary 2.4.14 (Church–Rosser) *If $\Gamma \vdash M = N : A$ then there is a P such that $M \rhd^* P$ and $N \rhd^* P$.*

We could also show the properties of Free Variables, Renaming and Strengthening for the simply typed lambda calculus.

2.5 Discussion and Related Work

As we have seen, we are able to give simpler proofs for important properties of type theory by using the derivations of λ^S instead of those of the simply typed lambda calculus. The soundness and completeness results are essential to our study of the metatheory for simply typed lambda calculus, to transfer the results from the typed operational semantics to the semantic presentation. In particular, we emphasize that the Church–Rosser and subject reduction properties follow straightforwardly for λ^S, and that the Church–Rosser property does not follow by simple induction on terms for $\beta\eta$-reduction and terms with type labels.

Several ideas have been influential in the development of typed operational semantics. Martin-Löf[53] gives an early proof of Church–Rosser and normalization for a typed notion of reduction: the reduction relation there is defined to have the same rules of inference as the judgemental equality excluding symmetry. Although this is not a formal system, it is one of the few proofs in the literature which proves normalization (our soundness) for a typed reduction, and typed operational semantics can be seen as a restriction of this more general reduction.

In another direction, Mitchell's notion of type closed predicate for the simply typed lambda calculus [62] is close to a formal system. A type closed predicate is a predicate $\mathcal{P}$ on terms which satisfies the following conditions:

$$(\mathcal{P}\text{-}Var) \quad \frac{\mathcal{P}(M_1) \wedge \ldots \wedge \mathcal{P}(M_n)}{\mathcal{P}(x(M_1, \ldots, M_n))} \quad \textit{for } x \textit{ of appropriate type}$$

$$(\mathcal{P}\text{-}App) \qquad \frac{\forall N.\mathcal{P}(N) \supset \mathcal{P}(M(N))}{\mathcal{P}(M)}$$

$$(\mathcal{P}\text{-}\beta) \qquad \frac{\mathcal{P}(N) \wedge \mathcal{P}(([N/x]M)(N_1, \ldots, N_n))}{\mathcal{P}((\lambda x{:}A.M)(N, N_1, \ldots, N_n))}$$

The conditions Mitchell gives are similar to our rules of inference for the typed operational semantics, but there are several important differences:

- Normal forms are not included in the formulation of type closed predicates. Having the extra information of the normal form of a term is useful in proving properties about reduction. For example, our use of normal forms in the definition of our typed operational semantics led us to a proof of weak Church–Rosser, whereas Mitchell states that there is no obvious type closed predicate to prove this property.

- There is no explicit relationship between the type closed predicates and reduction. Our definition instead makes it clear that we are defining a special reduction relation with type information. This is a benefit when we are studying the reduction behavior of well-typed terms, because the reduction relation we define is related to standard reduction.

- Mitchell's rule ($\mathcal{P}$-*App*) is not structural. This seems to be a consequence of the usual proof of soundness, as we have discussed above in Section 2.4.2. Using the rule ($\mathcal{P}$-*App*) instead of ours both leads to a strange formal system and makes using the predicates more difficult. For example, Church–Rosser is difficult to prove using type closed predicates, because it is not immediate that M is Church–Rosser if $M(N)$ and N are.

- The definition of type closed predicate does not contain information about contexts, which limits the generality of the predicates. For example, there is no type closed predicate which proves strengthening.

- The definition of type closed predicate is informal with respect to applications to sequences of terms, which makes it appear slightly simpler than it is.

The above points suggest that although our definition of typed operational semantics appears more complicated than Mitchell's type closed predicates, our approach is in fact simpler because it provides structure which is helpful in studying reduction.

The relationship between type closed predicates and typed operational semantics makes clear the view of λ^S as an alternative induction principle for well-typed terms in the simply typed lambda calculus: the rules of inference can instead be seen as conditions on predicates. One way of thinking of λ^S is as a formal system expressing general properties of predicates which need to be established in order to know that the predicate holds for judgements in the simply typed lambda calculus. The soundness result says that such predicates are a model for the simply typed lambda calculus.

Typed operational semantics can also be seen as presenting a modification of Coquand's algorithm [15], described as a formal system. He presents a reduction relation to weak head normal form and a bisimilarity relation which compares these weak head normal forms for $\beta\eta$-convertibility. These relations are untyped and therefore need to be annotated with type information.

Our work offers the following technical improvements over the approach in Coquand's paper:

- Our typed operational semantics is itself a formal system. In addition to its conceptual simplicity, this means that we can prove metatheoretic results by showing the equivalence of formal systems instead of considering an untyped reduction relation.

- Similarly, our derivations define a general class of predicates which hold for typed terms in the simply typed lambda calculus, as we discussed above, and we can establish results such as strong normalization by induction on the derivations of our typed operational semantics. Coquand can prove certain predicates, such as strengthening, but there is no obvious generalization in his presentation.

- We give a direct proof of normalization instead of using a bisimilarity relation for testing conversion. In addition to being a much simpler algorithm, because conversion is tested by reducing two terms to normal form and testing whether they are syntactically identical, this allows a considerable simplification of the soundness proof. Because we exploit the full power of standard reduction to normal form, we only need to consider predicates about well-typedness and not relations for bisimilarity.

The simply typed lambda calculus is a good system for explaining many important points about typed operational semantics, but there is one important aspect of typed operational semantics which is only brought out in the study of systems with dependent types. In these systems, with type equalities that allow a term to have syntactically different but judgementally equal types, proving the subject reduction property can be difficult. The typed operational semantics presentation makes these proofs easier by giving a presentation without an explicit type equality rule coercing a term from one type to another. This will be essential to our treatment of the Logical Framework and the calculus UTT.

Chapter 3

UTT

In this chapter we present the calculus *UTT*. This system is formulated in Martin-Löf's Logical Framework [63], a formal system which Martin-Löf intended to be used as a framework for defining type theories. The Logical Framework is a weak theory that has functional types and dependency, and we define new type theories by introducing constants and equalities into this theory. These constants represent the terms and rules of inference of the type theory we are representing in the Logical Framework.

The Logical Framework used in *UTT* differs from Martin-Löf's by having labels on the framework-level abstractions. Including the labels makes the study of the metatheory for the system more difficult.

3.1 Structure and Intended Use

The type theory *UTT* incorporates the aspects of type theory that we have discussed in the introduction. We shall give a brief overview of the type theory and some motivations for the constructs included in the type theory.

UTT has an impredicative universe of propositions which is distinct from the collection of all types. This reflects the view that propositions are indeed mathematical objects which can be understood in the type-theoretic framework, but that our logical reasoning should be confined to an independent universe which does not change as we alter the overall type theory to incorporate different applications. Furthermore, we can establish that type-checking is decidable as a consequence of the results later in this thesis, so we shall be able to know whether a proof object is an element of the type of proofs of a particular proposition.

We also have a predicative universe of computational types in *UTT*. This universe includes inductive types which are important for programming and mathematics, such as natural numbers, product types, function spaces, sum types, and dependent product and sum types. We also include the type of propositions and the type of proofs of each proposition in this universe.

UTT provides an environment which is well-suited to studying the specification and verification of programs. The dependent product and sum types allow us to represent specifications, refinements or implementations of specifications and realizations adequately in the type theory. A specification in type theory is represented as a pair of a type, which contains the possible structures which realize the specification, and a predicate over that type, which specifies the properties that any structure should satisfy. Luo discusses this approach iextensively [46, 47]. Also, the notion of deliverable [57,56] is studied in the context of *ECC*, a subsystem of *UTT*.

The structure of *UTT* may be inappropriate for particular applications in programming or mathematics. For example, it seems unnatural to consider the type of propositions and the types of proofs of propositions to be computational types, although including them in the computational universe is useful in studying specification and verification. The Logical Framework, the independence of the logic and the conceptual openness of the type theory make it possible to consider the type-theoretic formulation best suited to a given application domain.

3.2 The Logical Framework

Martin-Löf's Logical Framework is intended to be used as a meta-language for introducing new type theories. There are several reasons for using such a framework to present a type theory. First, the Logical Framework allows us to form hypothetical judgements parameterized over types. As Nordström, Petersson, and Smith explain, most text books on logic have few completely formal derivations. For example, the statement "A implies A" involves a meta-level use of the variable A, which need not be a formal entity in the theory we are studying. The Logical Framework allows us to make assumptions that A is a type and to reason about arbitrary types. Secondly, for reasons which we shall discuss in detail later, inductive types can be presented effectively in the Logical Framework. Lastly, implementing type theories using the Logical Framework is easier, because the framework deals with issues such as binders once and for all.

We shall distinguish the collections in the type theory being formulated, the object language, from the collections in the Logical Framework, the meta-language, by referring to the collections in the framework as *kinds*. Extending our semantic explanation of the judgements in Section 1.1, we say that if A is a kind then we know what the elements of A are and when two elements of A are equal, the equality being a decidable equivalence relation. The notion of kind is more basic than the concept of type, because kinds remain conceptually open: we only need to be able to recognize that an object is an element of the kind. In contrast, as we explained in Section 1.1, if we know that A is a type then we must know what the canonical elements of A are, where these canonical elements are formed according to the introduction rules for the type. Corresponding to our change in terminology in Section 1.1, where we used "type" for his "set," what we call a "kind" he calls a "type."

We have the following forms of judgement in the Logical Framework and the corresponding semantic explanations:

- A kind. This means that we know what the elements of the kind A are, and we also know when two elements of A are equal. The equality must be a decidable equivalence relation.

- $A = B$. This means that A and B are kinds that they have the same elements, and furthermore that the equalities on elements for A and B are the same.

- $M: A$. This means that M is an element of the kind A.

- $M = N: A$. This means that M and N are elements of the kind A and that they are equal under the equality for A.

Judgements with hypotheses in the Logical Framework are explained in the same way as judgements with hypotheses in type theory, as we have described in Section 1.1. We therefore introduce an infinite set V of names for the hypotheses, and another judgement form, $\vdash\Gamma$, which means that Γ represents a valid sequence of hypotheses. We also parameterize each of the four judgements above by sequences of hypotheses Γ.

Notation We write $\Gamma \vdash J$ to mean any of the five judgement forms under the sequence of hypotheses Γ.

We use the following conventions for variable names:

- $\Gamma, \Delta, \Phi, \ldots$ denote arbitrary sequences of hypotheses,

- $A, B, C, \ldots$ denote arbitrary kinds,

- $M, N, P, \ldots$ denote arbitrary terms, and

- $x, y, z, \ldots$ denote arbitrary variables.

3.2.1 Contexts and Substitution Rules

Because of the explanation of judgements with hypotheses, we can introduce general rules which correspond to our understanding of what it means to know a judgement with hypotheses. For example, the judgement $x{:}A \vdash B$ kind was explained to mean that we know

- $() \vdash [M/x]B$ kind for any element $() \vdash M : A$ and
- $() \vdash [M/x]B = [N/x]B$ for any elements $() \vdash M = N : A$.

This explanation, generalized to arbitrary finite sequences of hypotheses, justifies the rules (*Kind Subst*) and (*Kind Subst-Eq*) below.

We first introduce the syntactic construct for representing sequences of hypotheses.

Definition 3.2.1 (Pre-Context) *A sequence $x_1{:}A_1, \ldots, x_n{:}A_n$ of pairs of variables $x_i \in V$ and kinds A_i is a* pre-context. *The empty context is denoted by* $()$.

The domain *of a pre-context Γ, $dom(\Gamma)$, is the set of variables $\{x_1, \ldots, x_n\}$.*

We have several rules which allow us to judge that a sequence of hypotheses is well-formed. We shall call a pre-context Γ such that $\vdash\Gamma$ a *valid context.*

$$(Emp)\ \frac{}{\vdash ()} \qquad (Weak)\ \frac{\Gamma \vdash A \text{ kind} \quad x \notin dom(\Gamma)}{\vdash \Gamma, x{:}A} \qquad (Var)\ \frac{\vdash \Gamma_0, x{:}A, \Gamma_1}{\Gamma_0, x{:}A, \Gamma_1 \vdash x : A}$$

Figure 3–1: Contexts and Assumptions

Corresponding to the above description of the meaning of judgements with hypotheses, we have the following formal rules for substitution given in Figure 3–2.

3.2.2 The Kind Type

The kind Type represents the kind of all types which can be formed in the type theory being represented in the Logical Framework. In order to know that A is

(*Ctxt Subst*) $$\frac{\vdash \Gamma_0, z{:}C, \Gamma_1 \quad \Gamma_0 \vdash P : C}{\vdash \Gamma_0, [P/z]\Gamma_1}$$

(*Kind Subst*) $$\frac{\Gamma_0, z{:}C, \Gamma_1 \vdash A \text{ kind} \quad \Gamma_0 \vdash P : C}{\Gamma_0, [P/z]\Gamma_1 \vdash [P/z]A \text{ kind}}$$

(*Term Subst*) $$\frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M : A \quad \Gamma_0 \vdash P : C}{\Gamma_0, [P/z]\Gamma_1 \vdash [P/z]M : [P/z]A}$$

(*Kind Subst-Eq*) $$\frac{\Gamma_0, z{:}C, \Gamma_1 \vdash A \text{ kind} \quad \Gamma_0 \vdash P = Q : C}{\Gamma_0, [P/z]\Gamma_1 \vdash [P/z]A = [Q/z]A}$$

(*Term Subst-Eq*) $$\frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M : A \quad \Gamma_0 \vdash P = Q : C}{\Gamma_0, [P/z]\Gamma_1 \vdash [P/z]M = [Q/z]M : [P/z]A}$$

(*Kind-Eq Subst*) $$\frac{\Gamma_0, z{:}C, \Gamma_1 \vdash A = B \quad \Gamma_0 \vdash P : C}{\Gamma_0, [P/z]\Gamma_1 \vdash [P/z]A = [P/z]B}$$

(*Term-Eq Subst*) $$\frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M = N : B \quad \Gamma_0 \vdash P : C}{\Gamma_0, [P/z]\Gamma_1 \vdash [P/z]M = [P/z]N : [P/z]B}$$

Figure 3–2: Substitution Rules

a type, we shall need to know exactly what we prescribed in Section 1.1: what the canonical elements of A are and when two canonical elements of A are equal. Similarly, two types A and B are equal if they have the same canonical elements and the equalities for A and B are the same.

Introducing a kind of all types does not make our theory inconsistent because kinds themselves are not a part of the object language. Reflecting the conceptual openness of the universe of types, in order to know that Type is a kind we do not need to know what all types in the type theory are, we only need to know how to recognize what a type is and when two types are equal.

There will also be associated with each type A in the object language a kind $\mathrm{El}(A)$ of the elements of A. To know that an object is in the kind $\mathrm{El}(A)$ we must know that it evaluates to a canonical form which is in A, and to know that two objects are equal objects in $\mathrm{El}(A)$ we must know that they evaluate to equal canonical forms in A.

$$(\mathrm{Type}) \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathrm{Type\ kind}}$$

$$(\mathrm{El}) \quad \frac{\Gamma \vdash A : \mathrm{Type}}{\Gamma \vdash \mathrm{El}(A)\ \mathrm{kind}} \qquad (\mathrm{El}\text{-}Eq) \quad \frac{\Gamma \vdash A = B : \mathrm{Type}}{\Gamma \vdash \mathrm{El}(A) = \mathrm{El}(B)}$$

Figure 3–3: The Kind Type

Notation In writing kinds we shall often omit the El operator.

3.2.3 Judgemental Equality

The rules in Figure 3–4 stipulate that the judgemental equalities of the Logical Framework for terms and for kinds are equivalence relations.

$$(KRefl)\ \frac{\Gamma \vdash A\ \mathrm{kind}}{\Gamma \vdash A = A} \qquad (KSym)\ \frac{\Gamma \vdash A = B}{\Gamma \vdash B = A}$$

$$(KTrans)\ \frac{\Gamma \vdash A = B \quad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

$$(Refl)\ \frac{\Gamma \vdash M : A}{\Gamma \vdash M = M : A} \qquad (Sym)\ \frac{\Gamma \vdash M = N : A}{\Gamma \vdash N = M : A}$$

$$(Trans)\ \frac{\Gamma \vdash M = N : A \quad \Gamma \vdash N = P : A}{\Gamma \vdash M = P : A}$$

Figure 3–4: General Equality Rules

In Figure 3–5 we have the rules that reflect our understanding of the meaning of the judgement $\Gamma \vdash A = B$: that the kinds A and B have the same elements and that the equalities for the two kinds are identical.

$$(=T)\ \frac{\Gamma \vdash M : A \quad \Gamma \vdash A = B}{\Gamma \vdash M : B} \qquad (=R)\ \frac{\Gamma \vdash M = N : A \quad \Gamma \vdash A = B}{\Gamma \vdash M = N : B}$$

Figure 3–5: Equality Typing Rules

3.2.4 Dependent Product Kinds

As we have said, we shall define the object-level type theory through the use of the meta-language of the Logical Framework. The fundamental tool we use in the Logical Framework is the mechanism of a kind of higher-order dependent functions. These functions will correspond to expressions with holes in them in the object language. As we discussed at the beginning of this chapter, the higher-order functions allow us to explain the binding mechanism of the object-level through the binding mechanism of the Logical Framework.

The β and η-equalities for dependent products express two basic operations on expressions with holes. The rule (β) allows us to fill a hole in an expression, which involves using the meta-level substitution. The rule (η) states that two different terms denote the same expression with a hole. The equality on terms induced by the equalities of the Logical Framework is a *definitional equality*: it merely says that two terms in the Logical Framework denote the same expression, with or without holes. When we describe new types in the framework we shall need to introduce equalities to express the object-level equality on elements of the types.

We have differed from Martin-Löf's presentation of the terms of the Logical Framework by including type labels in abstractions. This is also the presentation used in the Edinburgh Logical Framework. The constructs $(x{:}A_1)A_2$ and $[x{:}A_1]M_0$ are binders.

Notation We use the convention of writing $(A)B$ for $(x{:}A)B$ if x does not occur in the free variables of B.

We shall also write $f(M_1, \ldots, M_n)$ for $f(M_1) \ldots (M_n)$.

3.3 Inductive Data Types in *UTT*

So far we have only introduced a calculus for notation, the Logical Framework. An essential feature of programming languages is the ability to define datatypes and to perform recursion over the elements. One datatype which is included in almost all programming languages is the type of numbers, so we shall demonstrate the type-theoretic approach to inductive types by beginning with this example.

3.3.1 An Example: The Natural Numbers

As we have discussed in Section 1.1, the first step in defining a new type in type theory is to show under what circumstances that type is well-formed. Here this is

$$(\Pi)\quad \frac{\Gamma, x{:}A_1 \vdash A_2 \text{ kind}}{\Gamma \vdash (x{:}A_1)A_2 \text{ kind}} \qquad (\Pi\text{-}Eq)\quad \frac{\Gamma \vdash A_1 = B_1 \quad \Gamma, x{:}A_1 \vdash A_2 = B_2}{\Gamma \vdash (x{:}A_1)A_2 = (x{:}B_1)B_2}$$

$$(\lambda)\quad \frac{\Gamma, x{:}A_1 \vdash M_0 : A_2}{\Gamma \vdash [x{:}A_1]M_0 : (x{:}A_1)A_2}$$

$$(\lambda\text{-}Eq)\quad \frac{\Gamma \vdash A_1 = B_1 \quad \Gamma, x{:}A_1 \vdash M_0 = N_0 : A_2}{\Gamma \vdash [x{:}A_1]M_0 = [x{:}B_1]N_0 : (x{:}A_1)A_2}$$

$$(App)\quad \frac{\Gamma \vdash M_1 : (x{:}A_1)A_2 \quad \Gamma \vdash M_2 : A_1}{\Gamma \vdash M_1(M_2) : [M_2/x]A_2}$$

$$(App\text{-}Eq)\quad \frac{\Gamma \vdash M_1 = N_1 : (x{:}A_1)A_2 \quad \Gamma \vdash M_2 = N_2 : A_1}{\Gamma \vdash M_1(M_2) = N_1(N_2) : [M_2/x]A_2}$$

$$(\beta)\quad \frac{\Gamma, x{:}A_1 \vdash M_0 : A_2 \quad \Gamma \vdash M_2 : A_1}{\Gamma \vdash ([x{:}A_1]M_0)(M_2) = [M_2/x]M_0 : [M_2/x]A_2}$$

$$(\eta)\quad \frac{\Gamma \vdash M : (x{:}A_1)A_2}{\Gamma \vdash [x{:}A_1]M(x) = M : (x{:}A_1)A_2}$$

Figure 3–6: Dependent Product Kinds

simple: the natural numbers are a well-formed type in any valid context.

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathrm{N} : \mathrm{Type}}$$

From now on, we shall assume that each of the rules introducing a new constant has the premiss that Γ is a valid context, and that its conclusion is that the constant is well-typed in Γ.

We also need to say what canonical objects are elements of the new type. For the natural numbers, 0 is always a natural number, and if n is a natural number then $\mathrm{succ}(n)$ is also a natural number. In our presentation in the Logical Framework, succ will be a meta-level function from N to N, instead of a new constant symbol which has a hole in it. In other words, we could instead have said that if n is a natural number then succ$[n]$ is a natural number, where succ$[n]$ is simply a new term, but instead we use the meta-level functional type to express this.

The rules of inference for these canonical elements are what we would expect:

$$0\colon \mathrm{N} \quad \text{and} \quad \mathrm{succ}\colon (\mathrm{N})\mathrm{N}$$

Non-canonical elements of type N will be variables or terms which have not been evaluated to a form where the outermost term constructor is either 0 or succ. For example, $([x{:}\mathrm{N}]x)(0)$ is not canonical, although it evaluates to the canonical form 0 by β-reduction. We can think of terms which have not been evaluated to a canonical form as programs which have not yet been run.

In order to write programs with the natural numbers, we shall want an induction principle over the elements. This induction principle will be a non-canonical object whose kind will reflect the usual induction principle on natural numbers. The induction principle, because it should be able to prove predicates over natural numbers, will make essential use of the dependency of the Logical Framework.

$$\begin{aligned}
\mathrm{E_N} \quad : \quad & (C{:}(\mathrm{N})\mathrm{Type}) \\
& (C(0)) \\
& ((n{:}\mathrm{N})(C(n))C(\mathrm{succ}(n))) \\
& \quad (z{:}\mathrm{N})C(z)
\end{aligned}$$

Because this induction principle allows us to defined dependent functions to any type, rather than just to propositions, we shall call it an elimination constant. We can define the usual recursion for natural numbers using the elimination constant corresponding to induction:

$$\begin{aligned}
R_{\mathrm{N}} \quad =_{\mathrm{df}} \quad & [C{:}\mathrm{Type}] \\
& [c{:}C] \\
& [f{:}(\mathrm{N})(C)C] \\
& \quad \mathrm{E_N}([n{:}\mathrm{N}]C, c, f)
\end{aligned}$$

Finally, we need to introduce equality rules. These rules relate the introduction rules, the canonical elements of the natural numbers, with the elimination constant or induction principle, by showing how a function defined by the elimination constant behaves on the canonical elements. There will be a separate such rule for each canonical element, and these are what we expect:

$$(\mathrm{E_N}-0) \qquad \frac{C\colon (\mathrm{N})\mathrm{Type} \quad c\colon C(0) \quad f\colon (n{:}\mathrm{N})(C(n))C(\mathrm{succ}(n))}{\mathrm{E_N}(C,c,f,0) = a\colon C(0)}$$

$$(\mathrm{E_N}-\mathrm{succ}) \quad \frac{C\colon (\mathrm{N})\mathrm{Type} \quad c\colon C(0) \quad f\colon (n{:}\mathrm{N})(C(n))C(\mathrm{succ}(n) \quad n\colon \mathrm{N}}{\mathrm{E_N}(C,c,f,\mathrm{succ}(n)) = f(n,\mathrm{E_N}(C,c,f,n))\colon C(\mathrm{succ}(n))}$$

These rules correspond to the usual definition of functions over the natural numbers by primitive recursion. We can write programs or functions on the natural numbers using the elimination constant and derive the expected equalities:

$$\begin{aligned} m+n &=_{\mathrm{df}} R_{\mathrm{N}}(\mathrm{N}, m, [x, v{:}\mathrm{N}]\mathrm{succ}(v), n) \\ m+0 &= m \\ m+\mathrm{succ}(n) &= \mathrm{succ}(m+n) \end{aligned}$$

We can furthermore use the elimination constant to show that expected properties hold for $+$, such as associativity and commutativity.

We shall see when we define the propositional equality that we can derive the Peano postulates using the elimination constant.

3.3.2 Schemas for Inductive Types

We need more than just a few basic types in a programming language. Older languages have certain basic types such as integers and then allow new types to be built up with records. Languages such as SML make it easy to create new types tailored to the programming task at hand. In order to capture this feature in type theory, we need a general way of introducing new inductive types. In type theory, because it is a logical language, we have the further problems that we need to put restrictions on this general mechanism to ensure the consistency of the system with the new types: in other words, we should not be able to write programs which never terminate.

Above, we introduced the natural numbers by giving each new constant explicitly. The induction principle we gave is clearly related to the introduction rules, and we would like to have a general way of generating an induction principle given reasonable introduction rules for a new type. The introduction of kind schemas to the Logical Framework, and the definitions which support this, will allow us to achieve exactly this.

Definition 3.3.1 (Small Kinds) *We say that a kind A is a* small kind *if*

- $A \equiv \mathrm{El}(M)$ *or*
- $A \equiv (x{:}A_1)A_2$*, where A_1 and A_2 are small kinds.*

The small kinds are kinds in which Type does not occur. The intention is that inductive types will be formulated as a sequence of kinds which represent the kinds of the introduction rules. Because it is inconsistent to have an embedding of the universe of types into itself, we need to ensure that the kinds representing the introduction rules have no occurrences of the kind Type.

Definition 3.3.2 (Schema Families) *Let Γ be a valid context and X be a variable.*

- *We say that a kind Φ is a* strictly positive operator in Γ with respect to X*, notation $\mathrm{P{\small OS}}_{\Gamma;X}(\Phi)$, if $\Gamma, X{:}\mathrm{Type} \vdash \Phi$ kind and*
  - $\Phi \equiv X$ *or*
  - $\Phi \equiv (x{:}A)\Phi_0$*, where A is a small kind and $\mathrm{P{\small OS}}_{\Gamma;X}(\Phi_0)$.*
- *We say that a kind Θ is an* inductive schema in Γ with respect to X*, notation $\mathrm{S{\small CH}}_{\Gamma;X}(\Theta)$, if*

- $\Theta \equiv X$,
- $\Theta \equiv (x{:}A)\Theta_0$, *where* A *is a small kind and* $\textsc{Sch}_{\Gamma;X}(\Theta_0)$, *or*
- $\Theta \equiv (\Phi)\Theta_0$, *where* $\textsc{Pos}_{\Gamma;X}(\Phi)$ *and* $\textsc{Sch}_{\Gamma;X}(\Theta_0)$.

- *We say that a finite sequence of kinds* $\Theta_1, \ldots, \Theta_n$ *is a* schema family in Γ with respect to X, *notation* $\textsc{Sch}_{\Gamma;X}(\Theta_1, \ldots, \Theta_n)$, *if* $\textsc{Sch}_{\Gamma;X}(\Theta_i)$ *for* $1 \leq i \leq n$.

Notation

- We write $\Phi(M)$ for $[M/X]\Phi$ if $\textsc{Pos}_{\Gamma;X}(\Phi)$, and similarly for Θ such that $\textsc{Sch}_{\Gamma;X}(\Theta)$ and $\Theta_1, \ldots, \Theta_n$ such that $\textsc{Sch}_{X;\Gamma}(\Theta_1, \ldots, \Theta_n)$.

- If Θ is a schema family in Γ with respect to X then we write $\textsc{Arity}_X(\Theta)$ for the sequence $\Phi_1, \ldots, \Phi_m$ of strictly positive operators in Γ with respect to X in Θ. We also extend this to schema families in the obvious way.

- We use the notation $F[x_1, \ldots, x_n] =_{\mathrm{df}} M$ to introduce definitions with variables $x_1, \ldots, x_n$. The notation $F[N_1, \ldots, N_n]$ then means the term or kind $[N_1, \ldots, N_n/x_1, \ldots, x_n]M$.

Definition 3.3.3

- *Assume that* $\textsc{Pos}_{\Gamma;X}(\Phi)$, *where* $\Phi \equiv (x_1{:}A_1)\ldots(x_n{:}A_n)X$, *and suppose that* $\Gamma \vdash A : \mathrm{Type}$, $\Gamma \vdash C : (A)\mathrm{Type}$ *and* $\Gamma \vdash z : \Phi(A)$. *Then define*

$$\Phi^\circ[A, C, z] =_{\mathrm{df}} (x_1{:}A_1)\ldots(x_n{:}A_n)C(z(x_1, \ldots, x_n))$$

- *Suppose* $\textsc{Sch}_{\Gamma;X}(\Theta)$, *where* $\Theta \equiv (x_1{:}M_1)\ldots(x_n{:}M_n)X$, *let* A *and* C *be as above and let* $\Gamma \vdash z : \Phi(A)$. *Then define*

$$\begin{aligned}\Theta^\circ[A, C, z] \quad =_{\mathrm{df}} \quad & (x_1{:}M_1(A))\ldots(x_n{:}M_n(A)) \\ & (\Phi^\circ_{i_1}[A, C, x_{i_1}])\ldots(\Phi^\circ_{i_k}[A, C, x_{i_k}])C(z(x_1, \ldots, x_n))\end{aligned}$$

- *Suppose* $\textsc{Pos}_{\Gamma;X}(\Phi)$, *where* $\Phi \equiv (x_1{:}A_1)\ldots(x_n{:}A_n)X$, *let* A, C *and* z *be as above, and let* $\Gamma \vdash f : (x{:}A)C(x)$. *Then define*

$$\Phi^\natural[A, C, f, z] =_{\mathrm{df}} [x_1{:}A_1]\ldots[x_n{:}A_n]f(z(x_1, \ldots, x_n))$$

We then have the following introduction rules for the constants associated with inductive types, where we again consider these introductions relative to a valid context Γ and $\bar{\Theta}$ a schema family in Γ with respect to X:

$$\begin{array}{rcl}
\mathcal{M}^X[\bar{\Theta}] & : & \mathrm{Type} \\
\iota_i^X[\bar{\Theta}] & : & \Theta_i(\mathcal{M}^X[\bar{\Theta}]) \quad (1 \leq i \leq n) \\
\mathrm{E}^X[\bar{\Theta}] & : & (C{:}(\mathcal{M}^X[\bar{\Theta}])\mathrm{Type}) \\
& & (f_1{:}\Theta_1^\circ[\mathcal{M}^X[\bar{\Theta}], C, \iota_1^X[\bar{\Theta}]]) \\
& & \ldots \\
& & (f_n{:}\Theta_n^\circ[\mathcal{M}^X[\bar{\Theta}], C, \iota_n^X[\bar{\Theta}]]) \\
& & \quad (z{:}\mathcal{M}^X[\bar{\Theta}])C(z)
\end{array}$$

We have the associated equality rule:

$$\begin{array}{rl}
& \mathrm{E}^X[\bar{\Theta}](C, \bar{f}, \iota_i^X[\bar{\Theta}](\bar{a})) \\
= & f_i(\bar{a}, \Phi_1^\natural[\mathcal{M}^X[\bar{\Theta}], C, \mathrm{E}^X[\bar{\Theta}](C, \bar{f}), a_1], \ldots, \Phi_k^\natural[\mathcal{M}^X[\bar{\Theta}], C, \mathrm{E}^X[\bar{\Theta}](C, \bar{f}), a_k]) \\
: & C(\iota_i^X[\bar{\Theta}](\bar{a}))
\end{array}$$

for $1 \leq i \leq n$, where $\bar{f}$ stands for $f_1, \ldots, f_n$, $\bar{a}$ for $a_1, \ldots, a_m$ and $\textsc{Arity}_X(\Theta_i) \equiv \Phi_1, \ldots, \Phi_k$.

The kinds of the constants for the introduction and elimination of inductive types require explicit information about the inductive type involved: we need to provide the full schema. The extra information is needed in order for it to be decidable whether a term has a given kind. These operations are therefore monomorphic, in contrast to the polymorphic application in the Logical Framework, which requires no type information, and the framework abstraction which only has partial type information.

We can define many interesting inductive types using this general schema:

$$
\begin{array}{rcl}
\mathbf{0} & =_{\mathrm{df}} & \mathcal{M}^X[] \\
\mathbf{1} & =_{\mathrm{df}} & \mathcal{M}^X[X] \\
\mathbf{2} & =_{\mathrm{df}} & \mathcal{M}^X[X, X] \\
\mathrm{N} & =_{\mathrm{df}} & \mathcal{M}^X[X, (X)X] \\
\mathit{list} & =_{\mathrm{df}} & [A{:}\mathrm{Type}]\mathcal{M}^X[X, (A)(X)X] \\
\rightarrow & =_{\mathrm{df}} & [A, B{:}\mathrm{Type}]\mathcal{M}^X[((A)B)X] \\
\Pi & =_{\mathrm{df}} & [A{:}\mathrm{Type}][B{:}(A)\mathrm{Type}]\mathcal{M}^X[((x{:}A)B(x))X] \\
\Sigma & =_{\mathrm{df}} & [A{:}\mathrm{Type}][B{:}(A)\mathrm{Type}]\mathcal{M}^X[(x{:}A)(B(x))X] \\
+ & =_{\mathrm{df}} & [A, B{:}\mathrm{Type}]\mathcal{M}^X[(A)X, (B)X] \\
W & =_{\mathrm{df}} & [A{:}\mathrm{Type}][B{:}(A)\mathrm{Type}]\mathcal{M}^X[(x{:}A)((B(x))X)X]
\end{array}
$$

Finally, we need to have a rule of inference which equates constant expressions which have equal schema families.

$$
(\kappa\text{-}Eq) \quad \frac{\begin{array}{c}\textsc{Sch}_{\Gamma;X}(\bar{\Theta}) \;\; \textsc{Sch}_{\Gamma;X}(\bar{\Theta}') \;\; \Gamma \vdash \kappa^X[\bar{\Theta}] : A \quad \Gamma \vdash \kappa^X[\bar{\Theta}'] : A \\ \Gamma, X{:}\mathrm{Type} \vdash \Theta_i = \Theta_i' \;\; 1 \leq i \leq n\end{array}}{\Gamma \vdash \kappa^X[\bar{\Theta}] = \kappa^X[\bar{\Theta}'] : A}
$$

Luo [47] calls the Logical Framework with the rule (κ-*Eq*) and no added constants *the Logical Framework with inductive schema.*

We give an example of the need for the rule (κ-*Eq*). Suppose we consider the type of lists over a type A, which we have represented above as $\mathcal{M}^X[X, (A)(X)X]$. Furthermore, suppose A' is some type which is judgementally equal to A, $\vdash A = A' : \mathrm{Type}$. We certainly then intend that the type of lists over A is equal to the type of lists over A', and furthermore that the term constructors for these two types are equal as well. We need the rule (κ-*Eq*) to show that $\vdash \mathcal{M}^X[X, (A)(X)X] = \mathcal{M}^X[X, (A')(X)X] : \mathrm{Type}$, and similarly for ι_0 and ι_1 and E (and μ if it is in the universe, as we define it below).

3.4 An Impredicative Universe of Propositions

We now introduce the universe of propositions. *UTT* follows the idea, as we discussed in Section 1.1.4, that the propositions-as-types principle gives an embedding from propositions to types but does not necessarily identify propositions and types. We therefore introduce the universe of propositions as a type and a coercion operator from this universe to types.

The impredicative universe of propositions is introduced by the following constant declarations:

$$
\begin{array}{rcl}
\mathrm{Prop} & : & \mathrm{Type} \\
\mathrm{Prf} & : & (\mathrm{Prop})\mathrm{Type} \\
\forall & : & (A{:}\mathrm{Type})((A)\mathrm{Prop})\mathrm{Prop} \\
\Lambda & : & (A{:}\mathrm{Type}) \\
 & & (P{:}(A)\mathrm{Prop}) \\
 & & ((x{:}A)\mathrm{Prf}(P(x))) \\
 & & \quad \mathrm{Prf}(\forall(A,P)) \\
\mathrm{E}_\forall & : & (A{:}\mathrm{Type}) \\
 & & (P{:}(A)\mathrm{Prop}) \\
 & & (R{:}(\mathrm{Prf}(\forall(A,P)))\mathrm{Prop}) \\
 & & ((g{:}(x{:}A)\mathrm{Prf}(P(x)))\mathrm{Prf}(R(\Lambda(A,P,g)))) \\
 & & \quad (z{:}\mathrm{Prf}(\forall(A,P)))\mathrm{Prf}(R(z))
\end{array}
$$

with the equality rule

$$\mathrm{E}_\forall(A,P,R,f,\Lambda(A,P,g)) = f(g) : \mathrm{Prf}(R(\Lambda(A,P,g)))$$

with the appropriate types for the variables.

Again, the constants we define here are monomorphic, in that they require full information about the proposition being proved.

As we have discussed earlier, kinds are conceptually open in the sense that we only need to recognize the elements of a kind, rather than needing to know all of the elements of the kind inductively. The introduction of new types into the Logical Framework using the dependent product kind and the kind Type reflects this openness: we are saying that we can recognize these new objects to be elements of their respective kinds.

The universe of propositions is impredicative because the universal quantification is allowed to range over arbitrary types and because we have a coercion from propositions to types: the expression $\forall(\mathrm{Prop}, [X{:}\mathrm{Prop}]\mathrm{Prf}(X))$ quantifies over all propositions and is itself a proposition.

We have linked the type theory and the logic intimately by allowing the quantifier to range over all types. This adheres to the view that the type theory should be open and that the logic in the system is universal: if we add new types to the type theory we can form new propositions over these types, but we do not need to alter the ways of forming propositions.

We could have used more restrictive quantifiers if we were interested in using the logic for specific types. For example, instead of the constant $\forall$ with a domain of all types we could have introduced the constants:

$$\begin{aligned} \forall_{\mathrm{Prop}} &: (X{:}\mathrm{Prop})\mathrm{Prop} \\ \forall_{\mathrm{Prf}} &: (X{:}\mathrm{Prop})(p{:}\mathrm{Prf}(X))\mathrm{Prop} \end{aligned}$$

plus separate introduction and elimination constants for these. This would restrict our logic so that it could only form statements about propositions and proofs of propositions.

We can define the usual application operator using the dependent elimination operator we have introduced:

$$\begin{aligned}
\mathrm{app}(A, P, M, N) \;&=_{\mathrm{df}}\; \mathrm{E}_\forall(A, P, [G{:}\mathrm{Prf}(\forall(A, P))]P(N), \\
&\qquad [g{:}(x{:}A)\mathrm{Prf}(P(x))]g(N), M) \\
&: \;(A{:}\mathrm{Type}) \\
&\quad (P{:}(A)\mathrm{Prop}) \\
&\quad (\mathrm{Prf}(\forall(A, P))) \\
&\quad (N{:}A)\mathrm{Prf}(P(N))
\end{aligned}$$

We can derive the usual equality rule corresponding to β-reduction for propositional redices:

$$\mathrm{app}(A, P, \Lambda(A, P, g), N) = g(N) : \mathrm{Prf}(P(a))$$

Impredicativity is a very strong principle. It allows us to define many of the logical connectives which are taken as primitive in weaker logics. For example, we can encode the following connectives:

$$\begin{aligned}
A \supset B \;&=_{\mathrm{df}}\; \forall(\mathrm{Prf}(A), [x{:}\mathrm{Prf}(A)]\mathrm{Prf}(B)) \\
\mathbf{true} \;&=_{\mathrm{df}}\; \forall(\mathrm{Prop}, [X{:}\mathrm{Prop}]X \supset X) \\
\mathbf{false} \;&=_{\mathrm{df}}\; \forall(\mathrm{Prop}, [X{:}\mathrm{Prop}]X) \\
A \wedge B \;&=_{\mathrm{df}}\; \forall(\mathrm{Prop}, [X{:}\mathrm{Prop}]A \supset B \supset X) \\
A \vee B \;&=_{\mathrm{df}}\; \forall(\mathrm{Prop}, [X{:}\mathrm{Prop}](A \supset X) \to (B \supset X) \supset X) \\
\neg A \;&=_{\mathrm{df}}\; A \supset \mathbf{false} \\
\exists x{:}A.P[x] \;&=_{\mathrm{df}}\; \forall(\mathrm{Prop}, [X{:}\mathrm{Prop}](\forall(A, [x{:}A]P[x] \supset X)) \supset X)
\end{aligned}$$

We can also define equality using an impredicative encoding of the Leibniz principle:

$$\begin{aligned}
M =_A N \;&=_{\mathrm{df}}\; \forall(A \to \mathrm{Prop}, \\
&\qquad [P{:}A \to \mathrm{Prop}] \\
&\qquad \mathrm{App}(A, \mathrm{Prop}, P, M) \supset \\
&\qquad \mathrm{App}(A, \mathrm{Prop}, P, N))
\end{aligned}$$

where App is the application operator for the inductive type $\rightarrow$, defined in the same way as the above app.

We can justify these encodings of logical connectives using the propositions-as-types embedding and the metatheoretic properties of the system *UTT* as we develop them in Chapters 4 and 6. We shall also know that in addition to being an equivalence relation and having the substitution property, the Leibniz equality defined also reflects the judgemental equality: two objects are Leibniz equal if and only if they are judgementally equal.

Theorem 3.4.1 (Equality Reflection) *$() \vdash M = N : A$ if and only if there is a term p such that $() \vdash p : \mathrm{Prf}(M =_A N)$.*

Proof Following the technique of Martin-Löf [51] or Luo [47]. $\square$

A thorough discussion of the internal logic of *UTT* and the adequacy of the encodings of the logical operators can be found in Luo's book [47].

We can use recursion on the inductive types to define higher-order predicates for induction principles. Smith [73] shows that $\neg(\mathrm{Eq}(\mathrm{N}, 0, \mathrm{succ}(0))$, cannot be shown in Martin-Löf type theory with no universes, where Eq is Martin-Löf's intensional equality, but in the theory with a predicative universe it can be shown. However, it is not the predicativity of the universes which is essential but the existence of any universe: we can use the same style of prove in *UTT* to show that $\neg(0 =_{\mathrm{N}} \mathrm{succ}(0))$ in the impredicative universe. This is done by defining the predicate $P\colon \mathrm{N} \rightarrow \mathrm{Prop}$ which for 0 returns **true** and for any term of the form $\mathrm{succ}(n)$ returns **false**. Then by definition of Leibniz equality it is clear that if $\mathrm{App}(A, \mathrm{Prop}, P, 0)$ implies $\mathrm{App}(A, \mathrm{Prop}, P, \mathrm{succ}(n))$ then there must be a proof of **false**.

Similarly, to prove that if $s(m) =_{\mathrm{N}} s(n)$ then $m =_{\mathrm{N}} n$, we can use the predicate $P\colon \mathrm{N} \rightarrow \mathrm{Prop}$ which for 0 returns any proposition and for $\mathrm{succ}(x)$ returns the proposition that $m =_{\mathrm{N}} x$. It is clear by definition of Leibniz equality that if there is a proof of $\mathrm{App}(A, \mathrm{Prop}, P, \mathrm{succ}(m)) \Rightarrow \mathrm{App}(A, \mathrm{Prop}, P, \mathrm{succ}(n))$, which is judgementally equal to $(m =_{\mathrm{N}} m) \Rightarrow (m =_{\mathrm{N}} n)$, then there is a proof of $m =_{\mathrm{N}} n$.

We can also prove the equalities which correspond to the uniqueness of the constructors in the internal logic using the induction principle which is defined for the inductive types. For example, for the type $\Sigma x{:}A.B$ as we defined it above, we can show that

$$\mathbf{pair}(\pi_1(A, B, p), \pi_2(A, B, p)) =_{\Sigma x:A.B} p$$

where **pair** is the only constructor for the type and π_1 and π_2 are the projections defined using the elimination constant.

If we do not have the rule of η-equality for the Logical Framework then some of the uniqueness properties cannot be proved, for example for the Π-type. Luo [47] shows that if this rule is present then the uniqueness properties hold for all inductive types. This is part of the motivation for formulating *UTT* in the Logical Framework: the original proposal [19] formulated the inductive schema in the predicate universes of a type theory, altering the structure of the theory by having (η) as an equality in the type theory.

3.5 Universes

An important aspect of Martin-Löf's type theory is the predicative universes. We have already introduced an impredicative universe of propositions, which allows impredicative quantification. The predicative universe contains names of types, and allows us to quantify over types and write functions in the type theory which return types. This is important if we want to use type theory as a language for specification, in order to be able to express abstract data types and specifications.

The system we study in this thesis has only one predicative universe. It is therefore a subsystem of full UTT [47].

The declarations which define the predicative universe are:

$$\begin{array}{rcl} \mathrm{U} & : & \mathrm{Type} \\ \mathrm{T} & : & (\mathrm{U})\mathrm{Type} \\ \mathrm{prop} & : & \mathrm{U} \\ \mathrm{prf} & : & (\mathrm{Prop})\mathrm{U} \end{array}$$

The associated rules of eqaulity are:

$$(\text{T-prop}) \quad \frac{}{\mathrm{T}(\mathrm{prop}) = \mathrm{Prop} \colon \mathrm{Type}} \qquad (\text{T-prf}) \quad \frac{P \colon \mathrm{Prop}}{\mathrm{T}(\mathrm{prf}(P)) = \mathrm{Prf}(P) \colon \mathrm{Type}}$$

In order to introduce inductive types in the universe, we need an auxiliary operator which collects the types used in inductive schemas. These operations are defined only on the syntax of schemas.

Definition 3.5.1 ($\textsc{Types}_\Gamma(\bar{\Theta})$) *Let Γ be a pre-context and let A, Φ and Θ be a small kind, a strictly positive operator with respect to X and a schema with respect* *to X. The sets* $\textsc{Types}_\Gamma(A)$, $\textsc{Types}_\Gamma(\Phi)$ *and* $\textsc{Types}_\Gamma(\Theta)$ *are defined as follows:*

$$\textsc{Types}_\Gamma(A) \quad =_{\mathrm{df}} \quad \begin{cases} \{(\Gamma, A)\} & \textit{if } A \textit{ is a type} \\ \textsc{Types}_\Gamma(A_1) \cup \textsc{Types}_{\Gamma,x:A_1}(A_2) & \textit{if } A \equiv (x{:}A_1)A_2 \end{cases}$$

$$\textsc{Types}_\Gamma(\Phi) \quad =_{\mathrm{df}} \quad \begin{cases} \emptyset & \textit{if } \Phi \equiv X \\ \textsc{Types}_\Gamma(A_1) \cup \textsc{Types}_{\Gamma,x:A_1}(\Phi_0) & \textit{if } \Phi \equiv (x{:}A_1)\Phi_0 \end{cases}$$

$$\textsc{Types}_\Gamma(\Theta) \quad =_{\mathrm{df}} \quad \begin{cases} \emptyset & \textit{if } \Theta \equiv X \\ \textsc{Types}_\Gamma(A_1) \cup \textsc{Types}_{\Gamma,x:A_1}(\Theta_0) & \textit{if } \Theta \equiv (x{:}A_1)\Theta_0 \\ \textsc{Types}_\Gamma(\Phi) \cup \textsc{Types}_\Gamma(\Theta_0) & \textit{if } \Theta \equiv (\Phi)\Theta_0 \end{cases}$$

For $\bar{\Theta} \equiv \Theta_1, \ldots, \Theta_n$*,*

$$\textsc{Types}_\Gamma(\bar{\Theta}) =_{\mathrm{df}} \bigcup_{1 \le i \le n} \textsc{Types}_\Gamma(\Theta_i)$$

We now consider the last rules of inference for *UTT*. Suppose $\textsc{Sch}_{\Gamma;X}(\bar{\Theta})$, and assume that there is a $\bar{\Theta}'$ such that $\textsc{Sch}_{\Gamma;X}(\bar{\Theta}')$, $\Gamma, X{:}\mathrm{Type} \vdash \Theta_i = \Theta_i'$ for $1 \le i \le n$, and for each $(\Gamma', A) \in \textsc{Types}_\Gamma(\bar{\Theta}')$ there is an a such that $\mathrm{T}(a) \equiv A$. Then we have that $\mu^X[\bar{\Theta}]$ is an element of the universe:

$$\Gamma \vdash \mu^X[\bar{\Theta}] \,:\, \mathrm{U} \quad \text{and} \quad \Gamma \vdash \mathrm{T}(\mu^X[\bar{\Theta}]) = \mathcal{M}^X[\bar{\Theta}] \,:\, \mathrm{Type}$$

This definition is formally different from Luo's [47], but we can show that it is equivalent. Our definition is easier to reason with when we consider the syntactic properties of the system.

Chapter 4

Basic Metatheory for *UTT*

In this chapter we shall consider the basic metatheory for the type theory UTT. We begin by discussing different aspects of the metatheory for type theories. We follow this by giving several definitions which will be used later in our understanding of the type theory. These include the definition of reduction and an equivalent presentation of UTT which omits some of the rules of inference.

Because of our approach of introducing a typed operational semantics, the focus of the chapter will be on the system UTT^S, the typed operational semantics for UTT. Nearly all of the metatheory will be developed for UTT^S, and we shall transfer these results to UTT via soundness and completeness results. We shall only need a few auxiliary lemmas about UTT to be able to prove the soundness theorem.

The fundamental result about the type theory in our development, that the typed operational semantics UTT^S is a sound interpretation of UTT, requires a complicated proof. We therefore leave this result for Chapter 6.

4.1 Metatheory for Type Theories

This section explains several aspects of the metatheory of type theory which are relevant to our treatment of the system UTT.

4.1.1 Presentations of Type Theory

We can have many different motivations for considering a type theory, and each of these motivations can give rise to a different formal presentation of the type theory. We use soundness and completeness results to show that these different systems really are just different views of the same underlying type theory.

We have described what we called the semantic presentation in Section 1.1. We can give an explanation of the judgements and rules of inference in this system in a way that justifies our belief in the correctness of the system. This presentation includes substitution or cut rules, judgemental equality and a judgement for valid contexts.

Unfortunately, the semantic presentation is very inconvenient to reason about, although it is the system we want to reason in. We have used typed operational semantics as the formal system to reason about, transferring the results about the typed operational semantics to the semantic presentation by soundness and completeness results. The typed operational semantics is a convenient system for metatheoretic study because it has a close correspondence between judgements and derivations.

Our interest in using type theory as a formalism implementable on the computer leads to another presentation of the system. For this purpose we want to use the minimal amount of computation necessary to determine whether a term is well-typed. Huet's constructive engine [40] is an excellent choice of presentation from this perspective for the Calculus of Constructions, and van Bentham Jutting, McKinna and Pollack [80] explore this problem for Pure Type Systems, a general class of type theories. Furthermore, the user should be able to give the minimal information necessary in the most convenient form, which leads to implicit arguments and type synthesis, universe polymorphism, closure under α-equivalence and other such features.

Because there are many ways of viewing a type theory, it is not clear exactly what we mean when we speak of a particular type theory. There are several syntactic ways in which we can vary the presentation of a type theory while still considering the type theory to be essentially the same:

- A simple way in which we can present a type theory in different ways is to have different judgement forms. For example, the system with both a valid context judgement and a typing judgement for terms can be shown equivalent with the system with a single typing judgement for terms.

- There can be presentations of the theory which are equivalent at the level of judgements but not at the level of derivations. For example, because the cut rule is admissible in many minimal presentations of type theories, adding this rule will not change the derivable judgements but will change the derivations.

- The syntax of terms can vary between presentations while we still might maintain some notion of the equivalence of the systems. One example of this is the presentation of calculi with named variables or with de Bruijn indices, where we introduce a syntax which is less convenient for the use of the theory but is considerably easier to reason about.

  A second example is the syntax of application and abstraction. Certainly the representation using polymorphic application and abstraction with a type label for the domain is the intended syntax, because it is most convenient for using type theory. However, as Streicher [75] notes, using a polymorphic application in systems which fail to have the Church–Rosser property introduces significant problems, and it is unlikely that the systems with and without explicit type labels on applications are equivalent in the sense which we intend.

It is not a new idea that different presentations of a theory emphasize different aspects of the theory. However, the study of syntax is fundamental to our understanding of type theory, and small changes to the syntax can significantly alter the behavior of a system. In the remainder of this section we shall discuss some variations in syntax and how these variations affect the metatheory for the type theories.

4.1.2 Reduction and Equality

One fundamental property for a type theory is the Church–Rosser property, which essentially says that equal terms have a common reduct. However, the name "Church–Rosser" has come to represent two different but related theorems in type theory, each associated with a different perspective on the relative importance of the notions of reduction and equality. These differing perspectives affect the presentation of the type theory because of the principle of replacing a type by an equal one: in each system we can say that if a term has a certain type, and another type is equal to that type, then the term has the second type.

The first view, influential in Barendregt's presentation of Pure Type Systems [8], is that reduction is simply a relation on untyped terms. Conversion is then defined to be the least equivalence relation containing reduction, and there is one rule incorporating the conversion, which works on untyped terms, into the type theory. In this setting the Church–Rosser property is viewed as a property of the untyped reduction, by analogy with the untyped lambda calculus, so we say that if a term has two reducts then those two reducts themselves have a common reduct. This property is also called the "diamond property," because of the diagrammatic representation of the theorem with reduction depicted as arrows, and the property can be formulated as a property for arbitrary relations rather than only reduction.

Investigations of metatheoretic properties for systems with conversion also require the subject reduction property, which says that if a term is well typed then any reduct of that term has the same type. Indeed, the properties of Church–Rosser and subject reduction will be necessary in the construction of almost any interesting model for systems with conversion. This is because proofs of conversion may involve terms which are not well-typed, because the proofs can be from expansions as well as reductions. In showing the correctness of a model construction, we need to know that the interpretation respects conversion for types, and it is possible that the interpretation may only be defined for well-typed terms and types. The argument for correctness, using the Church–Rosser and subject reduction properties, is as follows: suppose that these two properties hold, and that A and B are convertible types. Then by Church–Rosser we know that there is a common reduct for A and B, and by subject reduction we know that each type in these reduction sequences is well-typed. Thus all we need to show is that the interpretation respects reduction for types.

The other view is that equality between terms and types is central to the type theory, leading to a separate judgement form to express this equality. In presentations with judgemental equality, the rules of inference for equality and well-typedness are defined simultaneously, and a proof of equality will involve only well-typed terms or types between the two equal objects. In this presentation reduction is defined by analogy with the rules of inference which define the equality, and the Church–Rosser property is expressed by saying that if we can judge that two objects are equal then those two objects have a common reduct. Again, in this style of presentation reduction is not used in the actual type theory but is instead an auxiliary notion.

We shall follow the presentation with judgemental equality largely for practical reasons. It seems to be easier to use these systems to construct models for all but the simplest type theories. The important distinction between conversion and judgemental equality is that with judgemental equality we always know that a proof of equality involves only well-typed terms or types between the two equal objects. Because reduction is not a fundamental notion in the presentation of the system, it is not necessary to know any properties of reduction, such as Church–Rosser and subject reduction, in order to construct models. The untyped reduction is only an auxiliary notion which we consider for practical reasons. The main role of the subject reduction property will now be to explain the relationship between the untyped reduction and judgemental equality, and the relationship between systems with conversion and systems with judgemental equality.

As a consequence of our choice of presentation, we shall mean by the Church–Rosser property what we called the Church–Rosser property for systems with judgemental equality, that if two terms are judgementally equal then they have a common reduct. By the diamond property we shall mean that if a term reduces to two terms then those two terms have a common reduct. Each of these properties is of course sensitive to the definition of reduction: for example, it could be the case that a one-step reduction does not have the diamond property but its transitive closure does.

The decision between systems presented with conversion or judgemental equality in general has important consequences for the development of the metatheory of type theories. For β-equality this has been less evident, because we can give syntactic proofs of the diamond property for β-reduction. The proof essentially follows the proof for the untyped lambda calculus, where we consider a parallel one-step reduction which we can show to have the diamond property, and the transitive closure of the one-step reduction is equivalent to the full reduction. The subject reduction property can also be shown relatively straightforwardly for the system with only β-conversion. As a consequence, models can be constructed for these systems without any real difficulty, using the two important syntactic results.

However, the diamond property is not true for untyped $\beta\eta$-reduction on the usual term structure used for type theories, as we have already discussed in Section 2.3.3. This problem is more difficult in the context of type theories with dependent types, because for the same counterexample $[x{:}A]([y{:}B]M)(x)$, even if the two labels A and B are judgementally equal it could take an arbitrary number of reduction steps to find a common reduct for them.

The difficulties with $\beta\eta$-equality are important in our understanding of type theory. Although the role of η-equalities is not clear in systems considered as logics, such as the Calculus of Constructions, in the Logical Framework not having (η) as an equality influences the object logic: some propositions which we expect to be true in object logics are unprovable. We discussed this problem briefly in Section 3.3.2.

More generally, the failure of the diamond property for $\beta\eta$-reduction reinforces the belief that judgemental equality is preferable for the semantic presentation of type theories. If we are viewing the type theory as expressing a logical system then we should be more concerned with the well-typedness of terms than with their computational behavior. As we have discussed above, other presentations of the type theory may well use the notion of reduction instead of judgemental equality, for example the presentation which we use for the implementation of the type theory.

From the perspective of the metatheory of type theories, we shall see that using judgemental equality for the system with $\beta\eta$-equality leads to an elegant proof of the soundness of the typed operational semantics for UTT. As we have mentioned, the proof will not rely on the syntactic properties Church–Rosser and subject reduction being established beforehand. This is in contrast to Geuver's proof of Church–Rosser, subject reduction and strong normalization for the Calculus of Constructions with $\beta\eta$-conversion [30], which involves a complicated dependency between these syntactic results, for different reduction relations.

4.1.3 Syntax for Inductive Types

As we have mentioned in the introduction, there is a long-standing interest in the presentation of inductive types in type theory. We shall give here an outline of some of the more influential presentations and discuss the advantages and disadvantages of the different systems from the perspective of the metatheory.

The computational aspect of inductive types in type theory has been understood for a long time. Gödel's System T [34] is a type theory with computation over the natural numbers. Martin-Löf's well-ordering type [50] captures many inductive types in an elegant way. Böhm and Berarducci [11] formulate inductive datatypes using encodings in type theories with impredicativity.

The problem has been to give a formulation which includes not only the computation rules but also an adequate induction principle. An early presentation of inductive types is Martin-Löf's well-ordering or W-type [50,54]. The idea behind this type is that inductive types are well-ordered sets, and given enough basic types many interesting inductive types can be defined as well-orderings over the basic types. If we also include the internal equality type which reflects judgemental equality then this system is sufficiently powerful to provide a good basis for constructive mathematics, including as it does adequate representations of the important logical connectives, natural numbers, and a wide class of other inductive types.

Unfortunately, type-checking in the extensional theory is undecidable, which means that the theory fails to meet the criterion that we should be able to recognize proofs, where by proofs we mean proof objects rather than derivations. This is also a problem in computer science applications, since we would like the computational equality to be decidable. Another serious problem is that if an empty type and a universe are included then the type theory is no longer strongly normalizing. These problems are discussed in Nördstrom, Petersson and Smith's book [63].

Goguen and Luo [35] study the W-type in a system with added η-rules or filling-up rules. We can choose to have an extensional equality for particular types by adding these rules, which correspond to uniqueness conditions, only for those types. The filling-up rules are weaker than full extensionality, and therefore they are slightly more flexible than the extensional equality type, However, the essential problems remain for the empty type, and the problems with the filling-up rules seem to be similar to those with the extensional equality type.

Another proposal was to add fixed point operators to the type theory. Early papers [60,24] used the subset type, a significant addition to type theory. Coquand and Paulin-Mohring [19] introduce two different presentations of inductive types. One of these presentations requires the η-rules in order to derive the induction principles and so has the problems we have discussed above. The other is similar to the earlier proposals but avoids the need for the subset type.

UTT uses a mechanism for inductive types similar to the proposal of Coquand and Paulin-Mohring [19], which uses the existing Π-type in the type theory to define a general class of strictly positive schemas. However, one important difference between their system and UTT is that UTT is presented in the Logical Framework. This allows us to use the η-equality for the Π-type in the Logical Framework to gain an adequate induction principle for all of the encoded inductive types. In contrast, Coquand and Paulin-Mohring rely on η-equality for the

Π-type of the type theory being defined, which then changes our understanding of the type theory.

Dybjer [25] introduces a similar schematic method of introducing inductive types, based on a scheme introduced by Backhouse [6]. This schema also includes inductive families of types and simultaneous induction.

Coquand [16] formulates a system with inductive types and pattern matching. The pattern matching mechanism is similar to that of functional programming languages such as SML, but it is generalized to dependent types so pattern matching can be done on proofs. This system seems to offer an excellent environment for readable proofs, and the mechanism is implemented on an experimental basis in the ALF system mentioned in the introduction.

4.1.4 Metatheory for Inductive Types

There are several problems to consider in our understanding of the type theory once we have a formulation of the syntax which adequately represents inductive types.

The interaction between inductive types and universes leads to a system which is proof-theoretically very strong. Giving a hierarchical explanation of the type theory, in other words explaining how each type is built from types which are simpler according to some measure, is consequently more difficult. The measures for understanding the hierarchy of types in less powerful systems, for example natural number measures for the Calculus of Constructions or for *ECC*, no longer apply in systems with inductive types. We construct a complexity measure for types which gives a hierarchical understanding of the type theory in Appendix A.

Syntactically, many representations of inductive types use monomorphic constructors, as we have described in Section 3.3. This will mean that we will have some form of label on both the introduction and elimination constants, and we need to consider whether the reduction relation requires the labels to be syntactically identical or not. This is also a problem in defining the impredicative universe when we formulate the type theory in the Logical Framework. We discuss our solution to this problem in Section 4.7.2.

A further complication is introduced by the more general induction principles associated with inductive types. The Calculus of Constructions as formulated by Coquand and Huet does not have a general induction principle over impredicative quantification: the only elimination principle is the non-dependent application, corresponding to recursion rather than induction. Such induction principles have already been considered in the early proofs of normalization by Martin-Löf[52,53].

4.2 Definitions for the Logical Framework

In Section 3.2 we said that the Logical Framework is an open theory and that we can add new constants to extend the type theory being defined in the framework. In order to study the type theory metatheoretically, however, we need to fix it as a particular theory. In this section we make the relevant definitions to be able to reason about the Logical Framework and the type theory UTT.

In Chapter 3 we referred to objects which are well-typed as terms or kinds, but in studying the metatheory we are also interested in the underlying syntactic objects. We introduce the syntactic objects of the Logical Framework inductively, the terms and kinds, and we shall explicitly mention when we require them to be well-typed.

Definition 4.2.1 (Terms and Kinds) *We have several basic constructors for terms and kinds in the Logical Framework:*

- *Let V be an infinite collection of names. Then there is a* variable *for each element of V.*
- *If $x \in V$, A is a kind and M is a term then* abstraction*, $[x{:}A]M$, is a term.*
- *If M and N are terms then* application*, $M(N)$, is a term.*

- Type *is a kind.*

- *If M is a term then* $\mathrm{El}(M)$ *is a kind.*

- *If $x \in V$ and A and B are kinds then the* Π-kind, *$(x{:}A)B$, is a kind.*

- *The constants introduced in Chapter 3, Sections 3.4 and 3.5, are terms.*

- *If $\bar{\Theta} \equiv \Theta_1, \ldots, \Theta_n$ is a sequence of kinds then $\kappa^X[\bar{\Theta}]$ is a term, for $\kappa \in \{\mathcal{M}, \iota_i, \mathrm{E}, \mu\}$, where $1 \leq i \leq n$.*

Definition 4.2.2 (Syntactic Equivalence) *Two terms are* syntactically equivalent, *$M \equiv N$, if they are equal up to renaming of bound variables.*

We identify terms which are syntactically equivalent.

Definition 4.2.3 (Free Variables) *The* free variables *of a term M,* $\mathrm{FV}(M)$, *are defined as usual to be the variables which are not in the scope of a binder.*

Definition 4.2.4 (Contexts) *Let $\Gamma \equiv x_1{:}A_1, \ldots, x_n{:}A_n$ be a pre-context.*

We define the domain *of Γ, notation $dom(\Gamma)$, as $\{x_1, \ldots, x_n\}$. The free variables of Γ, notation* $\mathrm{FV}(\Gamma)$, *are defined as*

$$\mathrm{FV}(\Gamma) =_{\mathrm{df}} dom(\Gamma) \cup \bigcup_{1 \leq i \leq n} \mathrm{FV}(A_i)$$

Γ *is a* context *if the x_i are distinct and* $\mathrm{FV}(A_i) \subseteq \{x_1, \ldots, x_{i-1}\}$ *for $1 \leq i \leq n$.*

We say that x is fresh *in Γ if $x \notin dom(\Gamma)$.*

Definition 4.2.5 (Pre-Schema Families) *Let X be a variable.*

- *We say that a kind Φ is a* pre-positive operator with respect to X, *notation* $\mathrm{PPos}_X(\Phi)$, *if*

  - $\Phi \equiv X$ *or*

  - $\Phi \equiv (x{:}A)\Phi_0$, *where A is a small kind and* $\mathrm{PPos}_X(\Phi_0)$.

- *We say that a kind* Θ *is a* pre-inductive schema with respect to X, *notation* $\mathrm{PS{\small CH}}_X(\Theta)$, *if*
  - $\Theta \equiv X$,
  - $\Theta \equiv (x{:}A)\Theta_0$, *where* A *is a small kind and* $\mathrm{PS{\small CH}}_X(\Theta_0)$, *or*
  - $\Theta \equiv (\Phi)\Theta_0$, *where* $\mathrm{PP{\small OS}}_X(\Phi)$ *and* $\mathrm{PS{\small CH}}_X(\Theta_0)$.
- *We say that a finite sequence of kinds* $\Theta_1, \ldots, \Theta_n$ *is a* pre-schema family with respect to X, *notation* $\mathrm{PS{\small CH}}_X(\Theta_1, \ldots, \Theta_n)$, *if* $\mathrm{PS{\small CH}}_X(\Theta_i)$ *for* $1 \leq i \leq n$.

A strictly positive operator Φ in Γ with respect to X is a pre-positive operator Phi with respect to X such that $\Gamma, X{:}\mathrm{Type} \vdash \Phi$ kind, and similarly for inductive schemas and schema families.

4.3 An Alternative Induction on Terms

We use an alternative induction principle on terms and kinds in order to ensure that we have the correct premisses for constants. The idea is that we want to know the inductive hypothesis for the argument kinds for constants which have functional kinds. Also, for constants which will be types, we need information about the kinds for the constructors for those types: for example, for the type $\mathrm{El}(\mathrm{Prf}(\forall(A, P)))$, whose interpretation will be defined in the definition for $\forall$, we need to know that the inductive hypothesis is valid for the kind $(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))$, which is an argument kind for the constructor Λ.

Specifically, if P is a predicate on terms and Q is a predicate on kinds, then the following will be several of the clauses for the induction principle:

- if Q holds for Type and P holds for M then Q holds for $\mathrm{El}(M)$.
- if Q holds for (Prop)Type then P holds for Prf.
- if Q holds for $(A{:}\mathrm{Type})((A)\mathrm{Prop})\mathrm{Prop}$ and Q holds for $(x{:}A)\mathrm{Prf}(P(x))$ then P holds for $\forall$.

- if Q holds for the kind of $\mathrm{E}^X[\bar{\Theta}]$, P holds for $\mathcal{M}^X[\bar{\Theta}]$, P holds for $\iota_i^X[\bar{\Theta}]$ and P holds for $\Phi^{\natural}_{i_j}[A, C, f, z]$, where A, C, f and z are variables and i_j ranges over the i for the strictly positive operators, then P holds for $\mathrm{E}^X[\bar{\Theta}]$.

Given that these clauses and those for the other constants hold then P holds for all terms and Q holds for all kinds. We can show the soundness of this induction principle simply by using the usual induction principle and the proofs which we are given for the basic constructors. A simple way of verifying that this principle is sound is to notice that the kind for each constant only depends on constants defined before it.

4.4 *UTT*$^-$

The rules of inference we have introduced for *UTT* are not the most convenient to reason about, because the substitution rules which we have included in *UTT* are admissible in the theory without them. We therefore introduce another calculus, *UTT*$^-$, whose judgements are the same as *UTT* but which does not include the substitution rules: specifically, we do not include the rules in the table for substitution rules on page 68. We indicate that the judgement has been derived in *UTT*$^-$ by writing $\vdash^-$ instead of $\vdash$. We shall use *UTT*$^-$ to construct a set-theoretic model of the calculus.

We shall assume for the systems *UTT* and *UTT*$^-$ that in each of the rules for constants, there is a subderivation of the type of the constant. Furthermore, in the rule ($\bar{\Theta}$-*Eq*), we shall assume that each $\Phi^{\natural}[A, C, f, z]$ is well-typed. It is clear that this in no way changes the derivable judgements, because the new rules are derivable from the existing ones, so the equivalence of the systems is trivial.

The relationship between *UTT*$^-$ and *UTT* in one direction is easy to establish:

Lemma 4.4.1 (Completeness of *UTT*$^-$ for *UTT*) *If* $\Gamma \vdash^- J$ *then* $\Gamma \vdash J$.

Proof By induction on derivations of *UTT*$^-$. Each rule of inference in *UTT*$^-$ is also a rule of inference in *UTT*. $\square$

For the other direction, we shall rely on the soundness of UTT^S for UTT and completeness of UTT^S for UTT^-.

A better approach to organizing our proof might be to prove the soundness and completeness of the categorical model of UTTfirst, which is a more abstract presentation of the calculus. Constructing the set-theoretic model and the normalization proof via this intermediate semantics might be easier than constructing the models directly.

4.5 Metatheory for *UTT*

We shall need some simple lemmas to construct the normalization proof and the set-theoretic model.

Definition 4.5.1 (Subderivation) *For any two derivations J_1 and J_2, J_1 is a* subderivation *of J_2 if J_1 occurs as a subtree of J_2.*

Lemma 4.5.2 (Context Validity for *UTT*) *Any derivation of $\Gamma_0, \Gamma_1 \vdash J$ has a subderivation of $\vdash \Gamma_0$, if $\Gamma_0, \Gamma_1 \vdash J$ is not $\vdash \Gamma_0$.*

Proof By induction on derivations in UTT. □

The notation for the above lemma may be confusing. We note that the notation $\Gamma \vdash J$ can represent any of the judgements of UTT, including $\vdash \Gamma$.

Lemma 4.5.3 (Contexts for *UTT*) *If $\Gamma \vdash J$ then Γ is a context. Furthermore:*

- *If* $\Gamma \vdash A$ kind *then* $\mathrm{FV}(A) \subseteq dom(\Gamma)$.
- *If* $\Gamma \vdash M : A$ *then* $\mathrm{FV}(M) \cup \mathrm{FV}(A) \subseteq dom(\Gamma)$.
- *If* $\Gamma \vdash A = B$ *then* $\mathrm{FV}(A) \cup \mathrm{FV}(B) \subseteq dom(\Gamma)$.
- *If* $\Gamma \vdash M = N : A$ *then* $\mathrm{FV}(M) \cup \mathrm{FV}(N) \cup \mathrm{FV}(A) \subseteq dom(\Gamma)$.

Proof By induction on derivations, using Context Validity for UTT and the fact that $\vdash \Gamma, x{:}A$ implies that $\Gamma \vdash A$ kind for (Π) and (λ). $\square$

These lemmas also follow for UTT^{-}.

4.6 Environments

Environments are simply assignments of values for a set of variables. We introduce the definition in order to fix notation.

Definition 4.6.1 (Environment) *An* environment *for S and Γ, where S is a set and Γ is a finite set of variables, is a function from Γ to S.*

Notation If ρ is an environment for S and Γ, $x \notin \Gamma$ and $v \in S$ then we write $\rho[x := v]$ for the environment ρ' such that:

$$\rho'(y) \quad =_{\text{df}} \quad \begin{cases} \rho(y) & \text{if } y \neq x \\ v & \text{if } y = x \end{cases}$$

4.7 Definitions for the Metatheory

In this section we give the basic definitions we shall need for the metatheoretic development of UTT.

4.7.1 Reduction and Normal Forms

We first introduce several notions for terms which will be important in the presentation of UTT^S.

Definition 4.7.1 (Compatible Closure) *Let R be a relation on terms. Then the* compatible closure *of R, notation $M \rhd_R N$, is the least relation satisfying the* *following rules:*

$$\text{(R-Inc)} \qquad \frac{MRN}{M \rhd_R N}$$

$$(\Pi\text{-}L) \quad \frac{A_1 \rhd_R B_1}{(x{:}A_1)A_2 \rhd_R (x{:}B_1)A_2} \qquad (\Pi\text{-}R) \quad \frac{A_2 \rhd_R B_2}{(x{:}A_1)A_2 \rhd_R (x{:}A_1)B_2}$$

$$(\lambda\text{-}L) \quad \frac{A_1 \rhd_R B_1}{[x{:}A_1]M_0 \rhd_R [x{:}B_1]M_0} \qquad (\xi) \quad \frac{M_0 \rhd_R N_0}{[x{:}A_1]M_0 \rhd_R [x{:}A_1]N_0}$$

$$(App\text{-}L) \quad \frac{M_1 \rhd_R N_1}{M_1(M_2) \rhd_R N_1(M_2)} \qquad (App\text{-}R) \quad \frac{M_2 \rhd_R N_2}{M_1(M_2) \rhd_R M_1(N_2)}$$

$$(\mathrm{El}) \qquad \frac{M \rhd_R N}{\mathrm{El}(M) \rhd_R \mathrm{El}(N)}$$

Definition 4.7.2 (Untyped Reduction) *We introduce the following one-step reduction relations:*

$$\begin{array}{llcll}
(\beta) & ([x{:}A_1]M_0)(M_2) & \beta & [M_2/x]M_0 & \\
(\eta) & [x{:}A]M(x) & \eta & M & x \notin \mathrm{FV}(M) \\
(\mathrm{E}_\forall) & \mathrm{E}_\forall(A, P, R, f, \Lambda(A', P', g)) & o & f(g) & \\
(\mathrm{E}^X[\bar\Theta]) & \mathrm{E}^X[\bar\Theta](C, \bar f, \iota_i^X[\bar\Theta'](\bar a)) & o & \quad f_i(\bar a, & \\
 & & & \quad \Phi^\natural_{i_1}[\mathcal{M}^X[\bar\Theta], C, \mathrm{E}^X[\bar\Theta](C, \bar f), a_{i_1}], & \\
 & & & \quad \ldots, & \\
 & & & \quad \Phi^\natural_{i_k}[\mathcal{M}^X[\bar\Theta], C, \mathrm{E}^X[\bar\Theta](C, \bar f), a_{i_k}]) & \\
(\mathrm{prop}) & \mathrm{T}(\mathrm{prop}) & o & \mathrm{Prop} & \\
(\mathrm{prf}) & \mathrm{T}(\mathrm{prf}(P)) & o & \mathrm{Prf}(P) & \\
(\mu^X[\bar\Theta]) & \mathrm{T}(\mu^X[\bar\Theta]) & o & \mathcal{M}^X[\bar\Theta] &
\end{array}$$

Let untyped reduction, $M \rhd N$, *be the compatible closure of all of the above rules. Let* object-level reduction *be the relation* $\rhd_o$.

A term M *is a* redex *if there is an* N *such that* $M\beta\eta o N$.

We do not require labels to be identical in the rules for $\mathrm{E}_\forall$ and $\mathrm{E}^X[\bar{\Theta}]$ in the above reduction: for example, in the rule $\mathrm{E}_\forall$ the terms A and A' are distinct, although we shall know that if the whole term is well-typed then A and A' are judgementally equal. From the point of view of implementations this is clearly the preferable definition of reduction, because we do not want to have to consider equivalence of labels. We discuss the two possible choices, either requiring the labels to be identical or allowing them to be different, further in Section 4.7.2.

Definition 4.7.3 *We write $M \rhd^+ N$ for the transitive closure of reduction and $M \rhd^* N$ for the reflexive, transitive closure of reduction.*

Lemma 4.7.4 (Substitution for Reduction) *If $M_1 \rhd N_1$ and $M_2 \rhd N_2$ then $[M_2/x]M_1 \rhd^* [N_2/x]N_1$.*

Proof We show that if $M_1 \rhd N_1$ then $[M_2/x]M_1 \rhd [M_2/x]N_1$ by induction on the proof of $M_1 \rhd N_1$, and that if $M_2 \rhd N_2$ then $[M_2/x]M_1 \rhd^* [N_2/x]M_1$ by induction on the structure of M_1. □

Definition 4.7.5 (Strongly Normalizing) *A term is* strongly normalizing *if all reduction sequences starting from that term terminate.*

Definition 4.7.6 (Pre-redex) *We say that a term M is a* full pre-redex *if M is an abstraction or of the forms $\mathrm{E}_\forall(A, P, R, f)$, $\mathrm{E}^X[\bar{\Theta}](C, \bar{f})$ or* T*.*

If M is a full pre-redex and M is not an abstraction then we say that M is an object-level full pre-redex*.*

A term which is a full pre-redex or a subterm $\mathrm{E}_\forall$, $\mathrm{E}_\forall(A)$, $\mathrm{E}_\forall(A, P)$ and so on is a pre-redex*.*

Definition 4.7.7 (Base Term) *We define the notion of* base term *inductively by the following rules:*

- *Variables are base terms.*

- *If M_1 is a base term then $M_1(M_2)$ is a base term.*

- *If M_2 is a base term and M_1 is an object-level full pre-redex then $M_1(M_2)$ is a base term.*

The base terms are terms where a variable is in the position of an introductory term, that is an abstraction or a canonical object in the type theory.

Definition 4.7.8 (Weak Head Normal) *A term M is* weak head normal *if it is a base term or a pre-redex.*

Definition 4.7.9 (Normal Form) *A term M or kind A is* normal *or* in normal form *if*

- *M is a variable,*

- *$A \equiv (x{:}A_1)A_2$ and A_1 and A_2 are normal,*

- *$M \equiv [x{:}A_1]M_0$, A_1 is normal and M_0 is normal and not of the form $N(x)$ with $x \notin \mathrm{FV}(N)$,*

- *$M \equiv M_1(M_2)$, M_1 and M_2 are normal and $M_1(M_2)$ is not a redex,*

- *$A \equiv \mathrm{Type}$,*

- *$A \equiv \mathrm{El}(M)$ and M is normal,*

- *M is a constant, or*

- *$M \equiv \kappa^X[\bar{\Theta}]$, where Θ_i is normal for $1 \leq i \leq n$.*

Lemma 4.7.10 *If M is normal then M has no reductions.*

Proof By induction on terms in normal form. □

The following facts about terms are easy to establish:

Lemma 4.7.11

- *If $M_1(M_2)$ is weak head normal and M_1 and M_2 are normal then $M_1(M_2)$ is normal.*
- *If M is weak head normal and $M \rhd^* N$ then N is weak head normal.*

Proof Straightforward. □

4.7.2 Labels and Untyped Reduction

In Definition 4.7.2, we defined an untyped reduction relation which allowed the labels of the constants to be different in the rules for $\mathrm{E}_\forall$-reduction and $\mathrm{E}^X[\bar{\Theta}]$-reduction. There is another formulation of untyped reduction, which Altenkirch [4] calls "tight" reduction in contrast to the reduction which ignores labels called "loose" reduction, that requires the labels for the introduction and elimination constants to be the same. This is closer to the presentation of equalities in UTT: the rule of tight reduction for $\mathrm{E}_\forall$ is

$$\mathrm{E}_\forall(A, P, R, f, \Lambda(A, P, g)) \rhd f(g)$$

and similarly for $\mathrm{E}^X[\bar{\Theta}]$. In systems like this with non-left linear reductions, we cannot prove the diamond property simply by induction on possible reductions: we can only prove the weak diamond property, that if a term has two single-step reducts then those two reducts have a common multi-step reduct. For example, suppose that we take the definition of reduction above for $\mathrm{E}_\forall$, and suppose $\mathrm{E}_\forall(A, P, R, f, \Lambda(A, P, g)) \rhd f(g)$ and $\mathrm{E}_\forall(A, P, R, f, \Lambda(A, P, g)) \rhd \mathrm{E}_\forall(A, P, R, f, \Lambda(A', P, g))$. We then need two reduction steps to reach the common reduct $f(g)$,

$$\mathrm{E}_\forall(A, P, R, f, \Lambda(A', P, g)) \rhd \mathrm{E}_\forall(A', P, R, f, \Lambda(A', P, g)) \rhd f(g)$$

One reason for considering tight reduction is that it is easier to show the property of subject reduction, which says that if $\Gamma \vdash M : A$ and $M \rhd N$ then

$\Gamma \vdash N : A$, where $\vdash$ represents judgements for some type theory. For example, Salvesen [70] shows that certain Pure Type Systems with $\beta\eta$-equality have the properties of Church–Rosser and closure using terms with labeled application and a tight reduction, relying on strong normalization. In our formulation we can prove the properties of subject reduction and Church–Rosser for well-typed terms for either notion of reduction by induction on derivations in UTT^S. We choose to do the development for loose reduction because it is easier to use practically: we do not need to look at labels at all in untyped reduction.

4.7.3 Pre-Substitutions

Definition 4.7.12 (Pre-Substitution) *We define a* pre-substitution *to be an environment for terms.*

Notation

- We write $\widehat{\delta}(M)$ for the result of simultaneously substituting the values for the variables in the domain of δ:

$$\widehat{\delta}(M) =_{\mathrm{df}} [\delta(x_1), \ldots, \delta(x_n)/x_1, \ldots, x_n]M$$

- If Δ has all components of Γ then we write $\mathrm{weak}_\Gamma^\Delta$ for the substitution $\mathrm{weak}_\Gamma^\Delta(x) = x$ from Δ to Γ.

- $\mathrm{id}_\Gamma =_{\mathrm{df}} \mathrm{weak}_\Gamma^\Gamma$.

Lemma 4.7.13

- $[N/x]M \equiv \widehat{\mathrm{id}_{\mathrm{FV}(M)-\{x\}}[x := N]}(M)$.

- $\delta \circ \widehat{\phi[x := N]}(M) \equiv \widehat{\delta \circ \phi[x := \widehat{\delta}(N)]}(M)$.

- $\widehat{\phi}(\widehat{\delta}(M)) \equiv \widehat{\phi \circ \delta}(M)$.

Lemma 4.7.14 *Suppose δ is a pre-substitution such that $\delta(X) \equiv Y$. Then:*

- *If* $\mathrm{PP{\small OS}}_X(\Phi)$ *then* $\mathrm{PP{\small OS}}_Y(\widehat{\delta}(\Phi))$, *and similarly for* $\mathrm{PS{\small CH}}_X(\Theta)$ *and* $\mathrm{PS{\small CH}}_X(\bar{\Theta})$.

- *If* $\mathrm{PP{\small OS}}_X(\Phi)$ *then*

$$\widehat{\delta}(\Phi^\circ[A, C, z]) \equiv (\widehat{\delta}(\Phi))^\circ[\widehat{\delta}(A), \widehat{\delta}(C), \widehat{\delta}(z)]$$

and similarly for $\mathrm{PS{\small CH}}_X(\Theta)$ *and* $\mathrm{PS{\small CH}}_X(\bar{\Theta})$.

- *If* $\mathrm{PP{\small OS}}_X(\Phi)$ *then*

$$\widehat{\delta}(\Phi^\natural[A, C, f, z]) \equiv (\widehat{\delta}(\Phi))^\natural[\widehat{\delta}(A), \widehat{\delta}(C), \widehat{\delta}(f), \widehat{\delta}(z)]$$

Proof By induction on the proofs that $\mathrm{PP{\small OS}}_X(\Phi)$, $\mathrm{PS{\small CH}}_X(\Theta)$ or $\mathrm{PS{\small CH}}_X(\bar{\Theta})$. □

4.8 UTT^S

: A Typed Operational Semantics for UTT

We now introduce UTT^S, the typed operational semantics for UTT. We have already discussed the motivations behind this idea in Section 1.3 and Chapter 2.

4.8.1 Judgements and Derivations

The judgement forms for UTT^S and their intuitive meanings will be:

- $\vdash^S \Gamma$, meaning that Γ is a valid sequence of assumptions,

- $\Gamma \vdash^S A \to^{\mathrm{nf}} B$, meaning that the kind A has normal form B which is a valid kind in Γ,

- $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$, meaning that M has normal form P which is a canonical form in the kind A in context Γ, and

- $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$, meaning that M has an outermost reduct which when contracted yields N, and both M and N are elements of kind A in context Γ.

We shall use the following abbreviations:

- $\Gamma \vdash^S M : B$ if $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : B$ for some P,

- $\Gamma \vdash^S A$ kind if $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ for some B,

- $\Gamma \vdash^S M \downarrow N\ [P] : B$ if $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : B$ and $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P : B$,

- $\Gamma \vdash^S A \downarrow B\ [C]$ if $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} C$ and $\Gamma \vdash^S B \rightarrow^{\mathrm{nf}} C$,

- $\textsc{Sch}^S_{\Gamma;X}(\bar{\Theta})$ if $\vdash^S \Gamma$, $\mathrm{P}\textsc{Sch}_X(\bar{\Theta})$ and $\Gamma, X{:}\mathrm{Type} \vdash^S \Theta_i \rightarrow^{\mathrm{nf}} \Theta'_i$ for $1 \leq i \leq n$ and

- $\Gamma \vdash^S \bar{\Theta} \downarrow \bar{\Theta}'\ [\bar{\Theta}'']$ if $\vdash^S \Gamma$, $\mathrm{P}\textsc{Sch}_X(\bar{\Theta})$, $\mathrm{P}\textsc{Sch}_X(\bar{\Theta}')$ and $\Gamma, X{:}\mathrm{Type} \vdash^S \Theta_i \downarrow \Theta'_i\ [\Theta''_i]$ for $1 \leq i \leq n$ and

UTT^S is defined to be the least relations defined by the rules of inference in Figures 4–1 to 4–4.

$$(S\text{-}Emp) \quad \frac{}{\vdash^S ()} \qquad (S\text{-}Weak) \quad \frac{\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B \qquad x \notin dom(\Gamma)}{\vdash^S \Gamma, x{:}A}$$

Figure 4–1: Contexts

4.8.2 Discussion

We have already discussed the motivation for typed operational semantics in the context of the simply typed lambda calculus in Section 2.1.3. The ideas underlying

$$(S\text{-}Var)\quad \frac{\Gamma_0, x{:}A, \Gamma_1 \vdash^S A \rightarrow^{\mathrm{nf}} B}{\Gamma_0, x{:}A, \Gamma_1 \vdash^S x \rightarrow^{\mathrm{nf}} x \,:\, B} \qquad (S\text{-}\Pi)\quad \frac{\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1 \quad \Gamma, x{:}A_1 \vdash^S A_2 \rightarrow^{\mathrm{nf}} B_2}{\Gamma \vdash^S (x{:}A_1)A_2 \rightarrow^{\mathrm{nf}} (x{:}B_1)B_2}$$

$$(S\text{-}\lambda)\quad \frac{\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1 \quad \Gamma, x{:}A_1 \vdash^S M_0 \rightarrow^{\mathrm{nf}} N_0 \,:\, B_2}{\Gamma \vdash^S [x{:}A_1]M_0 \rightarrow^{\mathrm{nf}} [x{:}B_1]N_0 \,:\, (x{:}B_1)B_2} \qquad N_0 \not\equiv N(x) \textit{ with } x \notin \mathrm{FV}(N)$$

$$(S\text{-}\eta)\quad \frac{\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1 \quad \Gamma, x{:}A_1 \vdash^S M_0 \rightarrow^{\mathrm{nf}} N(x) \,:\, B_2 \quad \Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P \,:\, (x{:}B_1)B_2}{\Gamma \vdash^S [x{:}A_1]M_0 \rightarrow^{\mathrm{nf}} P \,:\, (x{:}B_1)B_2}$$

$$(S\text{-}App)\quad \frac{\Gamma \vdash^S M_1 \rightarrow^{\mathrm{nf}} N_1 \,:\, (x{:}B_1)B_2 \quad \Gamma \vdash^S M_2 \rightarrow^{\mathrm{nf}} N_2 \,:\, B_1 \quad \Gamma \vdash^S [M_2/x]B_2 \rightarrow^{\mathrm{nf}} C}{\Gamma \vdash^S M_1(M_2) \rightarrow^{\mathrm{nf}} N_1(N_2) \,:\, C} \qquad M_1(M_2) \textit{ weak head normal}$$

$$(S\text{-Type})\quad \frac{\vdash^S \Gamma}{\Gamma \vdash^S \mathrm{Type} \rightarrow^{\mathrm{nf}} \mathrm{Type}} \qquad (S\text{-El})\quad \frac{\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} N \,:\, \mathrm{Type}}{\Gamma \vdash^S \mathrm{El}(M) \rightarrow^{\mathrm{nf}} \mathrm{El}(N)}$$

$$(S\text{-}C)\quad \frac{\vdash^S \Gamma}{\Gamma \vdash^S c \rightarrow^{\mathrm{nf}} c \,:\, A} \qquad c \textit{ a constant of type } A$$

$$(S\text{-}WH)\quad \frac{\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N \,:\, B \quad \Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P \,:\, B}{\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P \,:\, B}$$

Figure 4–2: Canonical Forms

$$(S\text{-}\mathcal{M}) \qquad \frac{\textsc{Sch}^S_{\Gamma;X}(\bar{\Theta})}{\Gamma \vdash^S \mathcal{M}^X[\bar{\Theta}] \to^{\mathrm{nf}} \mathcal{M}^X[\bar{\Theta}'] \ : \ \mathrm{Type}}$$

$$(S\text{-}\iota) \qquad \frac{\textsc{Sch}^S_{\Gamma;X}(\bar{\Theta})}{\Gamma \vdash^S \iota^X_i[\bar{\Theta}] \to^{\mathrm{nf}} \iota^X_i[\bar{\Theta}'] \ : \ \Theta'_i(\mathcal{M}^X[\bar{\Theta}'])} \quad 1 \le i \le n$$

$$(S\text{-E}) \qquad \frac{\textsc{Sch}^S_{\Gamma;X}(\bar{\Theta})}{\begin{array}{lll} \Gamma \vdash^S \mathrm{E}^X[\bar{\Theta}] \to^{\mathrm{wh}} \mathrm{E}^X[\bar{\Theta}'] & : & (C{:}(\mathcal{M}^X[\bar{\Theta}'])\mathrm{Type}) \\ & & (f_1{:}(\Theta'_1)^\circ[\mathcal{M}^X[\bar{\Theta}'], C, \iota^X_1[\bar{\Theta}']]) \\ & & \quad \ldots \\ & & (f_n{:}(\Theta'_n)^\circ[\mathcal{M}^X[\bar{\Theta}'], C, \iota^X_n[\bar{\Theta}']])(z{:}\mathcal{M}^X[\bar{\Theta}'])C(z) \end{array}}$$

$$(S\text{-}\mu) \qquad \frac{\Gamma \vdash^S \bar{\Theta} \downarrow \bar{\Theta}' \ [\bar{\Theta}'']}{\Gamma \vdash^S \mu^X[\bar{\Theta}] \to^{\mathrm{nf}} \mu^X[\bar{\Theta}''] \ : \ \mathrm{Type}} \quad (\Gamma', A) \in \textsc{Types}_\Gamma(\bar{\Theta}') \textit{ implies } A \equiv \mathrm{T}(a)$$

Figure 4–3: Canonical Forms–Schemas

$$(W\text{-}\beta)\quad \frac{\Gamma \vdash^S [x{:}A_1]M_0 : (x{:}B_1)B_2 \quad \Gamma \vdash^S M_2 : B_1 \quad \Gamma \vdash^S [M_2/x]B_2 \rightarrow^{\mathrm{nf}} C}{\Gamma \vdash^S ([x{:}A_1]M_0)(M_2) \rightarrow^{\mathrm{wh}} [M_2/x]M_0 : C}$$

$$(W\text{-}\mathrm{E}_\forall)\quad \frac{\begin{array}{c}\Gamma \vdash^S \mathrm{E}_\forall(A_1,P_1,R,f) \rightarrow^{\mathrm{nf}} \mathrm{E}_\forall(A_3,P_3,R',f') : (M{:}\mathrm{Prf}(\forall(A_3,P_3)))S \\ \Gamma \vdash^S \Lambda(A_2,P_2,g) \rightarrow^{\mathrm{nf}} \Lambda(A_3,P_3,g') : \mathrm{Prf}(\forall(A_3,P_3,g')) \\ \Gamma \vdash^S \mathrm{Prf}(R(\Lambda(A_2,P_2,g))) \rightarrow^{\mathrm{nf}} B\end{array}}{\Gamma \vdash^S \mathrm{E}_\forall(A_1,P_1,R,f,\Lambda(A_2,P_2,g)) \rightarrow^{\mathrm{wh}} f(g) : B}$$

$$(W\text{-}\mathrm{E}^X[\bar\Theta])\quad \frac{\begin{array}{c}\Gamma \vdash^S \mathrm{E}^X[\bar\Theta_1](C,\bar f) \rightarrow^{\mathrm{nf}} \mathrm{E}^X[\bar\Theta_3](C',\bar f') : (M{:}\mathcal{M}^X[\bar\Theta_3])D \\ \Gamma \vdash^S \iota_i^X[\bar\Theta_2](\bar a) \rightarrow^{\mathrm{nf}} \iota_i^X[\bar\Theta_3](\bar a') : \mathcal{M}^X[\bar\Theta_3] \\ \Gamma \vdash^S C(\iota_i^X[\bar\Theta_2](\bar a)) \rightarrow^{\mathrm{nf}} B\end{array}}{\begin{array}{l}\Gamma \;\; \vdash^S \;\; \mathrm{E}^X[\bar\Theta_1](C,\bar f,\iota_i^X[\bar\Theta_2](\bar a)) \rightarrow^{\mathrm{wh}} \\ \qquad\qquad f_i(\bar a, \Phi^\natural_{i_1}[\mathcal{M}^X[\bar\Theta_3], C, \mathrm{E}^X[\bar\Theta_3](C,\bar f), a_{i_1}], \ldots) : B\end{array}}$$

$$(W\text{-}\mathrm{prop})\quad \frac{\Gamma \vdash^S \mathrm{prop} : \mathrm{El}(\mathrm{U})}{\Gamma \vdash^S \mathrm{T}(\mathrm{prop}) \rightarrow^{\mathrm{wh}} \mathrm{Prop} : \mathrm{Type}}$$

$$(W\text{-}\mathrm{prf})\quad \frac{\Gamma \vdash^S \mathrm{prf}(P) : \mathrm{El}(\mathrm{U})}{\Gamma \vdash^S \mathrm{T}(\mathrm{prf}(P)) \rightarrow^{\mathrm{wh}} \mathrm{Prf}(P) : \mathrm{Type}}$$

$$(W\text{-}\mu)\quad \frac{\Gamma \vdash^S \mu^X[\bar\Theta] : \mathrm{El}(\mathrm{U})}{\Gamma \vdash^S \mathrm{T}(\mu^X[\bar\Theta]) \rightarrow^{\mathrm{wh}} \mathcal{M}^X[\bar\Theta] : \mathrm{Type}}$$

$$(W\text{-}App)\quad \frac{\Gamma \vdash^S M_1 \rightarrow^{\mathrm{wh}} N_1 : (x{:}A_1)A_2 \quad \Gamma \vdash^S M_2 : A_1 \quad \Gamma \vdash^S [M_2/x]A_2 \rightarrow^{\mathrm{nf}} B}{\Gamma \vdash^S M_1(M_2) \rightarrow^{\mathrm{wh}} N_1(M_2) : B}$$

$$(W\text{-}Obj)\quad \frac{\begin{array}{c}\Gamma \vdash^S M_1 : (x{:}B_1)B_2 \\ \Gamma \vdash^S M_2 \rightarrow^{\mathrm{wh}} N_2 : B_1 \\ \Gamma \vdash^S [M_2/x]B_2 \rightarrow^{\mathrm{nf}} C\end{array}}{\Gamma \vdash^S M_1(M_2) \rightarrow^{\mathrm{wh}} M_1(N_2) : C} \quad \textit{M_1 a full object-level pre-redex}$$

Figure 4–4: Weak Head Reduction

our presentation of a typed operational semantics for UTT are the same as those for the simply typed lambda calculus.

The system UTT^S is not simply a reduction relation with added type information. Each of the rules involving application requires the normal forms of the type labels to be identical, which means that we are implicitly using a rule of type equality. However, this use of the normal forms of types means that we do not need the rules (=-T) and (=-R).

This use of normal forms gives us useful information which is not available in UTT because of the judgemental equality. For example, if we have that $\Gamma \vdash^S [x{:}A_1]M_0 \ : \ (x{:}B_1)B_2$ then we shall know directly that $\Gamma, x{:}A_1 \vdash^S M_0 \ : \ B_2$ and $\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1$. This is crucially different from UTT: there we only know that if $\Gamma \vdash [x{:}A_1]M_0 \ : \ (x{:}B_1)B_2$ then there is an A_2 such that $\Gamma, x{:}A_1 \vdash M_0 \ : \ A_2$ and $\Gamma \vdash (x{:}A_1)A_2 = (x{:}B_1)B_2$.

Similar to the typed operational semantics for the simply typed lambda calculus, we have weakened some rules of inference in order to formulate an appropriate principle of induction. Using metatheoretic reasoning, we shall again be able to show the admissibility of the more natural η-reduction rule:

$$(S\text{-}\eta') \quad \frac{\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1 \qquad \Gamma, x{:}A_1 \vdash^S M \rightarrow^{\mathrm{nf}} N(x) \ : \ B_2}{\Gamma \vdash^S [x{:}A_1]M_0 \rightarrow^{\mathrm{nf}} N \ : \ (x{:}B_1)B_2} \quad x \notin \mathrm{FV}(N)$$

The rule (S-Var) has also been weakened. The premiss contains an implicit use of weakening, and enough information is already contained in the following rule:

$$(S\text{-}Var') \quad \frac{\vdash^S \Gamma_0, x{:}A, \Gamma_1 \qquad \Gamma_0 \vdash^S A \rightarrow^{\mathrm{nf}} B}{\Gamma_0, x{:}A, \Gamma_1 \vdash^S x \rightarrow^{\mathrm{nf}} x \ : \ B}$$

We can show the admissibility of this rule as well, after establishing weakening for UTT^S.

The kinds in some of the premisses for the weak head reduction relation may be misleading. For example, one of the premisses for the rule (W-$\mathrm{E}_\forall$) is

$$\Gamma \vdash^S \mathrm{E}_\forall(A_1, P_1, R, f) \rightarrow^{\mathrm{nf}} \mathrm{E}_\forall(A_3, P_3, R', f') \ : \ (M{:}\mathrm{Prf}(\forall(A_3, P_3)))S$$

The information in the kind here is merely indicative: the full kind information is uniquely determined by the context and the term $\mathrm{E}_\forall(A_1, P_1, R, f)$. We can use our understanding of the structure of derivations to reconstruct the kinds for the terms A_1, P_1, R and f.

In the rule (S-κ), we have written $A^X[\bar{\Theta}]$ to mean the appropriate type as given in the rules for inductive types in Section 3.3.2.

4.9 Metatheory for UTT^S

In this section, when we use the phrase "induction on derivations" we shall mean derivations in UTT^S unless we state otherwise.

4.9.1 Basic Metatheory

Lemma 4.9.1 (Generation) *Any judgement which is derivable in UTT^S has a uniquely determined last rule of inference.*

Proof By induction on derivations. □

As in Section 2.3.2, we have given a simplified statement of the lemma to avoid rewriting each rule of inference. Some particular consequences of this lemma are:

- If $\vdash^S \Gamma, x{:}A$ then there is a subderivation of $\Gamma \vdash^S A \to^{\mathrm{nf}} B$ for some B.

- If $\Gamma \vdash^S (x{:}A_1)A_2 \to^{\mathrm{nf}} (x{:}B_1)B_2$ then there are subderivations of $\Gamma \vdash^S A_1 \to^{\mathrm{nf}} B_1$ and $\Gamma, x{:}A_1 \vdash^S A_2 \to^{\mathrm{nf}} B_2$.

- If $\Gamma \vdash^S M_1(M_2) \to^{\mathrm{nf}} P : C$ and $M_1(M_2)$ is weak head normal then there are P_1, P_2, B_1 and B_2 such that $P \equiv P_1(P_2)$, $\Gamma \vdash^S M_1 \to^{\mathrm{nf}} P_1 : (x{:}B_1)B_2$, $\Gamma \vdash^S M_2 \to^{\mathrm{nf}} P_2 : B_1$ and $\Gamma \vdash^S [M_2/x]B_2 \to^{\mathrm{nf}} C$.

- If $\Gamma \vdash^S M \to^{\mathrm{nf}} P : B$ and M is not weak head normal then there is an N such that $\Gamma \vdash^S M \to^{\mathrm{wh}} N : B$ and $\Gamma \vdash^S N \to^{\mathrm{nf}} P : B$.

- If $\Gamma \vdash^S M_1(M_2) \rightarrow^{\mathrm{wh}} N : B$ and M_1 is not weak head normal then there are N_1, P_2, B_1 and B_2 such that $N \equiv N_1(M_2)$, where $\Gamma \vdash^S M_1 \rightarrow^{\mathrm{wh}} N_1 : (x{:}B_1)B_2$, $\Gamma \vdash^S M_2 \rightarrow^{\mathrm{nf}} P_2 : B_1$ and $\Gamma \vdash^S [M_2/x]B_2 \rightarrow^{\mathrm{nf}} B$.

Lemma 4.9.2 (Uniqueness of Normal Forms)

- *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : B$ *and* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} Q : C$ *then* $P \equiv Q$ *and* $B \equiv C$.

- *If* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ *and* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} C$ *then* $B \equiv C$.

Proof By induction on derivations, using Generation in each case for the second derivation. □

Lemma 4.9.3 (Context Validity) *Any derivation of* $\Gamma_0, \Gamma_1 \vdash^S J$ *has a subderivation of* $\vdash^S \Gamma_0$, *if* $\Gamma_0, \Gamma_1 \vdash^S J$ *is not* $\vdash^S \Gamma_0$.

Proof By induction on derivations. □

Because we are using complete induction on derivations, we can use Generation or Context Validity in inductive proofs. We shall know by these lemmas that the inductive hypothesis applies for all subderivations.

Corollary 4.9.4 (Subderivation) *If* $\Gamma_0, x{:}A, \Gamma_1 \vdash^S J$ *then there is a subderivation of* $\Gamma_0 \vdash^S A \rightarrow^{\mathrm{nf}} B$ *for some* B.

Proof By Context Validity there is a subderivation of $\vdash^S \Gamma_0, x{:}A$. By Generation there is a subderivation of $\Gamma_0 \vdash^S A \rightarrow^{\mathrm{nf}} B$ for some B.

Lemma 4.9.5 (Completeness for $\boldsymbol{UTT^-}$)

- *If* $\vdash^S \Gamma$ *then* $\vdash^- \Gamma$.

- *If* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ *then* $\Gamma \vdash^- A$ kind *and* $\Gamma \vdash^- A = B$.

- *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *then* $\Gamma \vdash^- M : A$ *and* $\Gamma \vdash^- M = P : A$.

- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ *then* $\Gamma \vdash^- M : A$ *and* $\Gamma \vdash^- M = N : A$.

Proof By induction on derivations. We give several cases:

- (*S-Var*). By Context Validity we know that there is a subderivation of $\vdash^S \Gamma_0, x{:}A, \Gamma_1$, so by inductive hypothesis we know both $\vdash^- \Gamma_0, x{:}A, \Gamma_1$ and $\Gamma_0, x{:}A, \Gamma_1 \vdash^- A = B$. Hence $\Gamma_0, x{:}A, \Gamma_1 \vdash^- x : B$ by (*Var*) and (=-*T*), and $\Gamma_0, x{:}A, \Gamma_1 \vdash^- x = x : B$ by (*Refl*).

- (*S*-η). By inductive hypothesis we know that

  - $\Gamma \vdash^- A_1$ kind and $\Gamma \vdash^- A_1 = B_1$,
  - $\Gamma, x{:}A_1 \vdash^- M_0 : B_2$ and $\Gamma, x{:}A_1 \vdash^- M_0 = N(x) : B_2$, and
  - $\Gamma \vdash^- N : (x{:}B_1)B_2$ and $\Gamma \vdash^- N = P : (x{:}B_1)B_2$.

  Clearly $\Gamma \vdash^- (x{:}A_1)B_2 = (x{:}B_1)B_2$, so $\Gamma \vdash^- [x{:}A_1]M_0 : (x{:}B_1)B_2$. Also $\Gamma \vdash^- [x{:}A_1]M_0 = [x{:}B_1]N(x) : (x{:}B_1)B_2$ by (λ-*Eq*) and $\Gamma \vdash^- N = [x{:}B_1]N(x) : (x{:}B_1)B_2$ by (η), so $\Gamma \vdash^- [x{:}A_1]M_0 = P : (x{:}B_1)B_2$ by (*Sym*) and (*Trans*).

- (*W*-β). By inductive hypothesis

  - $\Gamma \vdash^- [x{:}A_1]M_0 : (x{:}C_1)C_2$ and $\Gamma \vdash^- [x{:}A_1]M_0 = P : (x{:}C_1)C_2$,
  - $\Gamma \vdash^- M_2 : C_1$,
  - $\Gamma \vdash^- [M_2/x]C_2 = C$ and
  - $\Gamma, x{:}A_1 \vdash^- M_0 : C_2$ and $\Gamma \vdash^- A_1 = C_1$ by Generation.

  Hence $\Gamma \vdash^- M_2 : A_1$ implies that $\Gamma \vdash^- ([x{:}A_1]M_0)(M_2) = [M_2/x]M_0 : [M_2/x]C_2$ by (β). Also by (λ) $\Gamma \vdash^- [x{:}A_1]M_0 : (x{:}A_1)C_2$, so by (*App*) $\Gamma \vdash^- ([x{:}A_1]M_0)(M_2) : [M_2/x]C_2$. We then apply the equality $\Gamma \vdash^- [M_2/x]C_2 = C$ twice to get the result.

- (*W*-μ). We know by Generation that there is a subderivation of $\vdash^S \Gamma$, that there is a Θ'_i such that $\Gamma, X{:}\mathrm{Type} \vdash^S \Theta_i \downarrow \Theta'_i\ [\Theta''_i]$, and that for all $(\Gamma', A) \in$

$\textsc{Types}_\Gamma(\Theta'_i)$ there is an a such that $A \equiv \mathrm{T}(a)$ for $1 \leq i \leq n$. Therefore, $\Gamma \vdash^- \mathrm{T}(\mu^X[\bar{\Theta}]) = \mu^X[\bar{\Theta}] : \mathrm{Type}$.

□

Lemma 4.9.6 (Contexts) *If $\Gamma \vdash^S J$ then Γ is a context. Furthermore:*

- *If $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ then $\mathrm{FV}(A) \subseteq dom(\Gamma)$ and $\mathrm{FV}(B) \subseteq dom(\Gamma)$.*
- *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : B$ then $\mathrm{FV}(M) \subseteq dom(\Gamma)$, $\mathrm{FV}(P) \subseteq dom(\Gamma)$ and $\mathrm{FV}(B) \subseteq dom(\Gamma)$.*
- *If $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : B$ then $\mathrm{FV}(M) \subseteq dom(\Gamma)$ and $\mathrm{FV}(B) \subseteq dom(\Gamma)$.*

Proof By induction on derivations. □

We use the same technique as in Section 2.3.2 to show that we can thin the context in judgements in UTT^S.

Definition 4.9.7 (Renaming) *A* renaming *is a substitution δ from Δ to Γ such that for each $(x{:}A) \in \Gamma$ we have $\delta(x) \equiv y$ and $(y{:}\widehat{\delta}(A)) \in \Delta$.*

Lemma 4.9.8
- *If δ is a renaming from Δ to Γ and ϕ is a renaming from Φ to Δ then $\phi \circ \delta$ is a renaming from Φ to Γ.*
- *If M is weak head normal and δ is a renaming from Δ to Γ then $\widehat{\delta}(M)$ is weak head normal.*

Lemma 4.9.9 (Renaming) *If δ is a renaming from Δ to Γ then*

- *If $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ then $\Delta \vdash^S \widehat{\delta}(A) \rightarrow^{\mathrm{nf}} \widehat{\delta}(B)$.*
- *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ then $\Delta \vdash^S \widehat{\delta}(M) \rightarrow^{\mathrm{nf}} \widehat{\delta}(P) : \widehat{\delta}(A)$.*

Proof By induction on derivations. We give several illustrative cases:

- (*S-Var*). Then by inductive hypothesis $\Delta \vdash^S \widehat{\delta}(A) \to^{\mathrm{nf}} \widehat{\delta}(B)$, and by definition $\delta(x)$ is a variable, so $\Delta \vdash^S \widehat{\delta}(x) \to^{\mathrm{nf}} \widehat{\delta}(x) : \widehat{\delta}(B)$.

- (*S*-Π). We know that $\Delta \vdash^S \widehat{\delta}(A_1) \to^{\mathrm{nf}} \widehat{\delta}(B_1)$ by inductive hypothesis, so $\vdash^S \Delta, y{:}\widehat{\delta}(A_1)$ for y fresh in Δ by (*S-Weak*). Hence $\delta[x := y]$ is a renaming from $\Delta, y{:}\widehat{\delta}(A_1)$ to $\Gamma, x{:}A_1$, so by inductive hypothesis again $\Delta, y{:}\widehat{\delta}(A_1) \vdash^S \widehat{\delta[x := y]}(A_2) \to^{\mathrm{nf}} \widehat{\delta[x := y]}(B_2)$. Since $\widehat{\delta}((x{:}A_1)A_2) \equiv (y{:}\widehat{\delta}(A_1))\widehat{\delta[x := y]}(A_2)$ and $\widehat{\delta}((x{:}B_1)B_2) \equiv (y{:}\widehat{\delta}(B_1))\widehat{\delta[x := y]}(B_2)$, we have that $\Delta \vdash^S \widehat{\delta}((x{:}A_1)A_2) \to^{\mathrm{nf}} \widehat{\delta}((x{:}B_1)B_2)$.

- (S-κ). These rules follow by inductive hypothesis and Lemma 4.7.14.

$\square$

Lemma 4.9.10 *If $\vdash^S\Gamma$, $\vdash^S\Delta$ and Δ has all components of Γ then* $\mathrm{weak}_\Gamma^\Delta$ *is a substitution from Δ to Γ.*

Proof By induction on the structure of Γ such that $\vdash^S\Gamma$, using Generation, Renaming and the inductive hypothesis for the case $\Gamma_0, x{:}A$. $\square$

As we mentioned, Renaming has the following corollary:

Corollary 4.9.11 (Thinning) *If $\Gamma \vdash^S J$ and Δ is a valid context with all components of Γ then $\Delta \vdash^S J$.*

Proof By Renaming and Thinning. $\square$

Lemma 4.9.12 (Context Replacement) *If $\Gamma_0 \vdash^S A \downarrow B\ [C]$ and $\Gamma_0, x{:}A, \Gamma_1 \vdash^S J$ then $\Gamma_0, x{:}B, \Gamma_1 \vdash^S J$.*

Proof By induction on derivations. The interesting cases are (*S-Weak*) and (*S-Var*).

- (*S-Weak*). Suppose Γ_1 is empty. Then $\Gamma_0 \vdash^S B \to^{\mathrm{nf}} C$ implies that $\vdash^S\Gamma_0, x{:}B$.

Otherwise, by inductive hypothesis $\Gamma_0, x{:}B, \Gamma_1 \vdash^S D \rightarrow^{\text{nf}} E$ so $\vdash^S \Gamma_0, x{:}B, \Gamma_1, y{:}D$.

- (*S-Var*). Suppose $\Gamma_0, x{:}A, \Gamma_1 \vdash^S J$ is $\Gamma_0, x{:}A, \Gamma_1 \vdash^S x \rightarrow^{\text{nf}} x : D$, where $\Gamma_0, x{:}A, \Gamma_1 \vdash^S A \rightarrow^{\text{nf}} D$. By Context Validity we know that $\vdash^S \Gamma_0, x{:}A, \Gamma_1$, so by Thinning $\Gamma_0, x{:}A, \Gamma_1 \vdash^S A \rightarrow^{\text{nf}} C$, and by Uniqueness of Normal Forms $C \equiv D$. By inductive hypothesis and Context Validity we know that $\vdash^S \Gamma_0, x{:}B, \Gamma_1$. By assumption $\Gamma_0 \vdash^S B \rightarrow^{\text{nf}} C$, so again by Thinning $\Gamma_0, x{:}B, \Gamma_1 \vdash^S B \rightarrow^{\text{nf}} C$. Hence $\Gamma_0, x{:}B, \Gamma_1 \vdash^S x \rightarrow^{\text{nf}} x : C$.

  Otherwise $\Gamma_0, x{:}B, \Gamma_1 \vdash^S D \rightarrow^{\text{nf}} E$ by inductive hypothesis, so $\Gamma_0, x{:}B, \Gamma_1 \vdash^S y \rightarrow^{\text{nf}} y : D$.

□

Strengthening is often the most difficult of the metatheoretic results for a type theory with an equality rule for types. For UTT^S this result is straightforward, because type equality is taken care of in the individual rules. Indeed, the kind in each judgement about terms must always be normal, and although we do not explicitly use this fact it ensures that the strengthened variable never occurs in the kind. This is different from the situation in *UTT*, where the kind could include a redex which has the strengthened variable but where the reduct does not.

Lemma 4.9.13 (Strengthening) *Suppose z is a variable such that $z \notin \mathrm{FV}(\Gamma_1)$. Then:*

- *If* $\vdash^S \Gamma_0, z{:}C, \Gamma_1$ *then* $\vdash^S \Gamma_0, \Gamma_1$.
- *If* $\Gamma_0, z{:}C, \Gamma_1 \vdash^S A \rightarrow^{\text{nf}} B$ *and* $z \notin \mathrm{FV}(A)$ *then* $\Gamma_0, \Gamma_1 \vdash^S A \rightarrow^{\text{nf}} B$.
- *If* $\Gamma_0, z{:}C, \Gamma_1 \vdash^S M \rightarrow^{\text{nf}} N : A$ *and* $z \notin \mathrm{FV}(M)$ *then* $\Gamma_0, \Gamma_1 \vdash^S M \rightarrow^{\text{nf}} N : A$.
- *If* $\Gamma_0, z{:}C, \Gamma_1 \vdash^S M \rightarrow^{\text{wh}} N : A$ *and* $z \notin \mathrm{FV}(M)$ *then* $\Gamma_0, \Gamma_1 \vdash^S M \rightarrow^{\text{wh}} N : A$.

Proof By induction on derivations. We check several cases, with the others being similar or following simply by inductive hypothesis.

- (*S-Weak*). Suppose $z = x$. Then $z \notin dom(\Gamma)$ by assumption, and $\vdash^S \Gamma$ by Context Validity.

  Otherwise, by inductive hypothesis $\Gamma_0, \Gamma_1 \vdash^S A \to^{\mathrm{nf}} B$, and by assumption $x \notin dom(\Gamma_0, z{:}C, \Gamma_1) \supseteq dom(\Gamma_0, \Gamma_1)$. Hence $\vdash^S \Gamma_0, \Gamma_1, x{:}A$.

- (*S-Var*). We know that $\Gamma \equiv \Gamma_0, z{:}C, \Gamma_1$, that $x{:}A \in \Gamma$, that $z \notin \mathrm{FV}(\Gamma_1)$ and $z \neq x$. By Contexts we know that $z \notin \mathrm{FV}(\Gamma_0)$ as well, so $z \notin \mathrm{FV}(A)$. Hence by inductive hypothesis $\Gamma_0, \Gamma_1 \vdash^S A \to^{\mathrm{nf}} B$, so $\Gamma_0, \Gamma_1 \vdash^S x \to^{\mathrm{nf}} x \; : \; B$.

- (*S-App*). By inductive hypothesis we know that $\Gamma_0, \Gamma_1 \vdash^S M_1 \to^{\mathrm{nf}} N_1 \; : \; (x{:}A_1)A_2$ and $\Gamma_0, \Gamma_1 \vdash^S M_2 \to^{\mathrm{nf}} N_2 \; : \; A_1$. By Contexts $z \notin \mathrm{FV}([M_2/x]A_2)$, so by inductive hypothesis $\Gamma_0, \Gamma_1 \vdash^S [M_2/x]A_2 \to^{\mathrm{nf}} B$. Hence $\Gamma_0, \Gamma_1 \vdash^S M_1(M_2) \to^{\mathrm{nf}} N_1(N_2) \; : \; B$.

□

4.9.2 Results About Reduction

The calculus UTT^S is a synthesis of standard reduction and typing rules from UTT. We are able to use this to considerable advantage in showing results about the relationship between untyped reduction and typing in UTT^S.

Lemma 4.9.14 (Adequacy for Untyped Reduction)

- *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P \; : \; A$ *then* P *is normal and there is an* N *such that* $M \rhd^*_{\beta o} N \rhd^*_{\eta} P$.

- *If* $\Gamma \vdash^S A \to^{\mathrm{nf}} C$ *then* C *is normal and* $A \rhd^*_{\beta o} B \rhd^*_{\eta} C$.

- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N \; : \; A$ *then* $M \rhd_{\beta o} N$.

Proof By induction on derivations. We consider (S-η).

By Contexts $x \notin \mathrm{FV}(P)$. Then by inductive hypothesis $A_1 \rhd^*_{\beta o} B_1 \rhd^*_\eta C_1$ and $M_0 \rhd^*_{\beta o} N_0 \rhd^*_\eta P(x)$, and C_1 and $P(x)$ are normal, so $[x{:}A_1]M_0 \rhd^*_{\beta o} [x{:}B_1]N_0 \rhd^*_\eta [x{:}C_1]P(x)$, and furthermore $[x{:}C_1]P(x) \rhd_\eta P$. Then, since $P(x)$ is normal we know that P is normal, so $P \equiv Q$ by Lemma 4.7.10 and the inductive hypothesis. $\square$

Corollary 4.9.15

- *If* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ *and* A *is normal then* $A \equiv B$.
- *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *and* M *is normal then* $M \equiv P$.

Proof By Adequacy for Reduction $M \rhd^* P$. By Lemma 4.7.10 M has no reductions, so $M \equiv P$. $\square$

Lemma 4.9.16 (Subject Reduction for (η)) *If* $\Gamma \vdash^S [x{:}A_1]M(x) \rightarrow^{\mathrm{nf}} P : (x{:}C_1)C_2$ *and* $x \notin \mathrm{FV}(M)$ *then* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : (x{:}C_1)C_2$.

Proof By Generation we know that $\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} C_1$, $\Gamma, x{:}A_1 \vdash^S M(x) \rightarrow^{\mathrm{nf}} P_0 : C_2$, and either $P_0 \equiv P(x)$, $x \notin \mathrm{FV}(P)$ and $\Gamma \vdash^S P \rightarrow^{\mathrm{nf}} P : (x{:}C_1)C_2$ (by Adequacy for Reduction and Corollary 4.9.15) or $P \equiv [x{:}C_1]P_0$ and $P_0 \equiv P'(x)$ implies $x \in \mathrm{FV}(P')$. By induction on the proof that $\Gamma, x{:}A_1 \vdash^S M(x) \rightarrow^{\mathrm{nf}} P_0 : C_2$ we show that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : (x{:}C_1)C_2$:

- (S-*App*). Then $\Gamma, x{:}A_1 \vdash^S M \rightarrow^{\mathrm{nf}} P' : (x{:}C_1)C_2$, $\Gamma, x{:}A_1 \vdash^S x \rightarrow^{\mathrm{nf}} x : C_1$ and $P_0 \equiv P'(x)$. Since $x \notin \mathrm{FV}(M)$ we know by Strengthening that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P' : (x{:}C_1)C_2$, so $x \notin \mathrm{FV}(P')$ implies that $P \equiv P'$.

- (W-β). Then $\Gamma, x{:}A_1 \vdash^S [y{:}B_1]N_0 \rightarrow^{\mathrm{nf}} P' : (x{:}C_1)C_2$, $\Gamma, x{:}A_1 \vdash^S x \rightarrow^{\mathrm{nf}} x : C_1$ and $\Gamma, x{:}A_1 \vdash^S [x/y]N_0 \rightarrow^{\mathrm{nf}} P_0 : C_2$. By Generation we know that $\Gamma, x{:}A_1 \vdash^S B_1 \rightarrow^{\mathrm{nf}} C_1$ and by Strengthening $x \notin \mathrm{FV}(M)$ implies that $\Gamma \vdash^S B_1 \rightarrow^{\mathrm{nf}} C_1$, so by Context Replacement $\Gamma, x{:}B_1 \vdash^S [x/y]N_0 \rightarrow^{\mathrm{nf}} P_0 : C_2$. Hence $\Gamma \vdash^S [x{:}B_1][x/y]N_0 \rightarrow^{\mathrm{nf}} P : (x{:}C_1)C_2$, by either ($S$-$\lambda$) or ($S$-$\eta$).

- (*W-App*). By inductive hypothesis, Strengthening and (*S-WH*).

$\square$

Definition 4.9.17 *The predicates* $\mathcal{S}_\Gamma^C(A)$ *and* $\mathcal{S}_\Gamma^{P,A}(M)$ *are defined as the least relations such that:*

- $\mathcal{S}_\Gamma^C(A)$ *if* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} C$ *and for all* B *if* $A \rhd B$ *then* $\mathcal{S}_\Gamma^C(B)$.
- $\mathcal{S}_\Gamma^{P,A}(M)$ *if*
  - $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$,
  - *for all* N *if* $M \rhd N$ *then* $\mathcal{S}_\Gamma^{P,A}(N)$, *and*
  - *for all* N *which are subterms of* M *there are* B *and* Q *such that* $\Gamma, \Delta \vdash^S N \rightarrow^{\mathrm{nf}} B : Q$ *where* Δ *is the sequence of binders which* N *occurs under in* M.

$\mathcal{C}_\Gamma^C(A)$ *and* $\mathcal{C}_\Gamma^{P,A}(M)$ *will be the same as* $\mathcal{S}_\Gamma^C(A)$ *and* $\mathcal{S}_\Gamma^{P,A}(M)$ *except that instead of* $M \rhd N$ *we use* $M \rhd^+ N$.

This definition is closely related to the definition of strong normalization in Altenkirch's thesis [4], although we have added type information. The predicate is well-defined because it is strictly positive. We shall also have the following induction principle on this predicate: if $\mathcal{S}_\Gamma^C(A)$ and $\mathcal{P}(\mathcal{B})$ for all B such that $A \rhd^* B$ then $\mathcal{P}(\mathcal{A})$, then $\mathcal{P}$ holds for all terms such that $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} C$, and similarly (but including subterms) for $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : C$.

Lemma 4.9.18 *The following properties hold for* $\mathcal{S}_\Gamma^{P,A}(M)$*:*

- *If* $\mathcal{S}_\Gamma^{P,A}(M)$ *then* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$.
- *If* $\mathcal{S}_\Gamma^{P,A}(M)$ *then* M *is strongly normalizing.*
- *If* $\mathcal{S}_\Gamma^{P,A}(M)$ *and* $M \rhd^* N$ *then* $\mathcal{S}_\Gamma^{P,A}(N)$.

- $\mathcal{S}_\Gamma^{P,A}(M)$ *if and only if* $\mathcal{C}_\Gamma^{P,A}(M)$.

- *If* $\mathcal{S}_\Gamma^{P,A}(M)$ *and* N *is a subterm of* M *then there are* Q *and* B *such that* $\mathcal{S}_\Gamma^{Q,B}(N)$.

- *If* $\Gamma_0 \vdash^S C \downarrow D\ [E]$ *then*
  - *if* $\mathcal{S}^B_{\Gamma_0,x:C,\Gamma_1}(A)$ *then* $\mathcal{S}^B_{\Gamma_0,x:D,\Gamma_1}(A)$, *and*
  - *if* $\mathcal{S}^{P,A}_{\Gamma_0,x:C,\Gamma_1}(M)$ *then* $\mathcal{S}^{P,A}_{\Gamma_0,x:D,\Gamma_1}(M)$.

- *If* z *is a variable such that* $z \notin \mathrm{FV}(\Gamma_1)$ *then*
  - *if* $\mathcal{S}^B_{\Gamma_0,x:C,\Gamma_1}(A)$ *and* $z \notin \mathrm{FV}(A)$ *then* $\mathcal{S}^B_{\Gamma_0,\Gamma_1}(A)$, *and*
  - *if* $\mathcal{S}^{P,A}_{\Gamma_0,x:C,\Gamma_1}(M)$ *and* $z \notin \mathrm{FV}(M)$ *then* $\mathcal{S}^{P,A}_{\Gamma_0,\Gamma_1}(M)$.

Proof By induction on proofs of $\mathcal{S}_\Gamma^{P,A}(M)$, using Context Replacement and Strengthening for the last two. $\square$

Lemma 4.9.19 (Reduction)

- *If* $\Gamma \vdash^S A \to^{\mathrm{nf}} B$ *then* $\mathcal{S}_\Gamma^B(A)$.

- *If* $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ *then* $\mathcal{S}_\Gamma^{P,A}(M)$.

- *If* $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$ *and* $\mathcal{S}_\Gamma^{P,A}(N)$ *then* $\mathcal{S}_\Gamma^{P,A}(M)$.

Proof By induction on derivations. We show two cases:

- $(S\text{-}\lambda)$. We need to show that
  - $\Gamma \vdash^S [x{:}A_1]M_0 \to^{\mathrm{nf}} [x{:}C_1]P_0 : (x{:}C_1)C_2$,
  - $[x{:}A_1]M_0 \rhd N$ implies $\mathcal{S}_\Gamma^{P,(x:C_1)C_2}(N)$ and
  - $\mathcal{S}_\Gamma^{C_1}(A_1)$ and $\mathcal{S}^{P_0,C_2}_{\Gamma,x:A_1}(M_0)$.

The first follows directly and the last follows by inductive hypothesis.

For the second, by induction on $\mathcal{S}^{C_1}_{\Gamma}(A_1)$ we know that $A_1 \rhd B_1$ and $\mathcal{S}^{P_0,C_2}_{\Gamma,x:B_1}(M_0)$ imply that $\mathcal{S}^{P,(x:C_1)C_2}_{\Gamma}([x{:}B_1]M_0)$ for all B_1. Also, by induction on $\mathcal{S}^{P_0,C_2}_{\Gamma,x:A_1}(M_0)$ we know that $M_0 \rhd N_0$ implies that $\mathcal{S}^{P_0,C_2}_{\Gamma,x:A_1}(N_0)$ for all N_0.

Hence, suppose $[x{:}A_1]M_0 \rhd N$. Then:

- $A_1 \rhd B_1$ implies that $[x{:}A_1]M_0 \rhd [x{:}B_1]M_0$. Then $\mathcal{S}^{C_1}_{\Gamma}(B_1)$, so $\Gamma \vdash^S B_1 \rightarrow^{\mathrm{nf}} C_1$ by Lemma 4.9.18. Hence $\mathcal{S}^{P_0,C_2}_{\Gamma,x:B_1}(M_0)$ by the Context Replacement component of Lemma 4.9.18, so by inductive hypothesis $\mathcal{S}^{P,(x:C_1)C_2}_{\Gamma}([x{:}B_1]M_0)$.
- $M_0 \rhd N_0$ implies that $[x{:}A_1]M_0 \rhd [x{:}A_1]N_0$. By inductive hypothesis.
- $[x{:}A_1]M(x) \rhd M$ by (η), where $x \notin \mathrm{FV}(M)$. Since M is a subterm of $[x{:}A_1]M(x)$ we know that $\mathcal{S}^{Q,B}_{\Gamma,x:A_1}(M)$ for some Q and B, and by the Strengthening component of Lemma 4.9.18 we know that $\mathcal{S}^{Q,B}_{\Gamma}(M)$. Furthermore by Lemma 4.9.16 we know that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : (x{:}C_1)C_2$, so by Uniqueness of Normal Forms $\mathcal{S}^{P,(x:C_1)C_2}_{\Gamma}(M)$.

- $(W\text{-}\beta)$. The result follows again by induction on the normalization premisses, where if $[x{:}A_1]M(x) \rhd M$ by (η) because $M_0 \equiv M(x)$ with $x \notin \mathrm{FV}(M)$ then $M(M_2) \equiv [M_2/x]M(x)$, and $\mathcal{S}^{P,C}_{\Gamma}([M_2/x]M_0)$ by assumption.

□

The inclusion of the η-equality rule has not significantly altered our treatment of the metatheory for UTT^S. The properties we need can be established in a style similar to the standard development [8,44]. Indeed, typed operational semantics seem to be a useful technical contribution to the metatheory for type theories exactly because of this uniformity of development.

Corollary 4.9.20 (Subject Reduction for UTT^S) *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ and $M \rhd^* N$ then $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P : A$.*

Corollary 4.9.21 (Strong Normalization for UTT^S) *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ then M is strongly normalizing.*

Corollary 4.9.22 (Diamond Property for UTT^S) *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$, $M \rhd^* M_1$ *and* $M \rhd^* M_2$ *then* $M_1 \rhd^* P$ *and* $M_2 \rhd^* P$.

4.9.3 Admissible Rules

We shall now show the admissibility of several rules which will be important in showing the soundness of UTT^S for UTT:

Proposition 4.9.23 (Admissibility of (*S-Var'*)) *If* $\vdash^S \Gamma_0, x{:}A, \Gamma_1$ *and* $\Gamma_0 \vdash^S A \rightarrow^{\mathrm{nf}} B$ *then* $\Gamma_0, x{:}A, \Gamma_1 \vdash^S x \rightarrow^{\mathrm{nf}} x : B$.

Proof By Thinning $\Gamma_0, x{:}A, \Gamma_1 \vdash^S A \rightarrow^{\mathrm{nf}} B$, so by (*S-Var*) $\Gamma_0, x{:}A, \Gamma_1 \vdash^S x \rightarrow^{\mathrm{nf}} x : B$. □

Proposition 4.9.24 (Admissibility of (*S-η'*)) *If* $\Gamma \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1$ *and* $\Gamma, x{:}A_1 \vdash^S M_0 \rightarrow^{\mathrm{nf}} N(x) : B_2$, *with* $x \notin \mathrm{FV}(N)$, *then* $\Gamma \vdash^S [x{:}A_1]M_0 \rightarrow^{\mathrm{nf}} N : (x{:}B_1)B_2$.

Proof By Adequacy for Reduction and Reduction we know that $\Gamma, x{:}A_1 \vdash^S N(x) \rightarrow^{\mathrm{nf}} N(x) : B_2$. Since $N(x)$ is normal by Normal Forms, the only rule which applies is (*S-App*), so by Generation there are C_1 and C_2 such that $\Gamma, x{:}A_1 \vdash^S N \rightarrow^{\mathrm{nf}} N : (x{:}C_1)C_2$, $\Gamma, x{:}A_1 \vdash^S x \rightarrow^{\mathrm{nf}} x : C_1$ and $\Gamma, x{:}A_1 \vdash^S [x/x]C_2 \rightarrow^{\mathrm{nf}} B_2$. By Generation for the variable we know that $\Gamma, x{:}A_1 \vdash^S A_1 \rightarrow^{\mathrm{nf}} C_1$, and by Thinning $\Gamma, x{:}A_1 \vdash^S A_1 \rightarrow^{\mathrm{nf}} B_1$, so by Uniqueness of Normal Forms $C_1 \equiv B_1$. Furthermore, by Normal Forms C_2 is normal, so $[x/x]C_2 \equiv B_2$ by Lemma 4.9.15. Finally, by Strengthening $\Gamma \vdash^S N \rightarrow^{\mathrm{nf}} N : (x{:}B_1)B_2$ and by (*S-η*) $\Gamma \vdash^S [x{:}A_1]M_0 \rightarrow^{\mathrm{nf}} N : (x{:}B_1)B_2$. □

The admissibility of the rules (*S-η'*) and (*S-Var'*) means that we could show the equivalence of a system which has these two rules instead of (*S-η*) and (*S-Var*) in UTT^S. Similarly, the soundness and completeness results for UTT would allow us to prove equivalence with a system with much stronger rules for weak head reduction. In particular, it seems that we could remove the type reductions in the premisses of many of the rules for weak head reduction.

Similar to the simply typed lambda calculus, we can use Adequacy for Reduction, Subject Reduction and Uniqueness of Normal Forms to show that if two terms have a common reduct and they are both well-typed in UTT^S then they have the same normal form. For example, we can show the admissibility of the following rule for application equality:

$$(S\text{-}App\text{-}Eq) \quad \frac{\Gamma \vdash^S M_1 \downarrow N_1\ [P_1] : (x{:}A_1)A_2 \quad \Gamma \vdash^S M_2 \downarrow N_2\ [P_2] : A_1 \quad \Gamma \vdash^S M_1(M_2) \to^{\mathrm{nf}} Q : B \quad \Gamma \vdash^S N_1(N_2) \to^{\mathrm{nf}} R : C}{\Gamma \vdash^S M_1(M_2) \downarrow N_1(N_2)\ [Q] : B}$$

4.10 Discussion

We have discussed many of the benefits and ideas of typed operational semantics in Chapter 2 on the simply typed lambda calculus. However, the technique of using a typed operational semantics to study the metatheory of type theory is particularly useful in the study of systems with dependent types. For the systems UTT and UTT^S, the derivations of UTT^S are easier to manipulate because there is only one derivation of well-typedness for any given context and term: there is no rule of conversion or type equality, which allows the type of a term to be replaced by an equal type. This allows us to have the desirable statement of the Generation lemma that, for example, if $\Gamma \vdash^S [x{:}A_1]M_0 \to^{\mathrm{nf}} P : (x{:}B_1)B_2$ then $\Gamma, x{:}A \vdash^S M_0 \to^{\mathrm{nf}} Q : B_2$ for some Q. In the semantic presentation, the type equality rule means that we cannot recover all of the type information from a judgement.

We emphasize again that the Church–Rosser and subject reduction properties follow straightforwardly for UTT^S. This is in contrast to several proofs for normalizing type systems [70,30], which use complicated syntactic methods to show Church–Rosser. Coquand's proof technique [15], which was part of the inspiration for our presentation, is similar to ours but does not develop a formal system to reason in, instead annotating an untyped reduction relation with type information.

Coquand's proof is for a special variant of Martin-Löf type theory with one universe and a Π-type. This variant does not distinguish syntactically between kinds and elements in the universe, and the Π-type only has application and not the full elimination rule. This means that the problem of matching labels discussed in Section 4.7.2 is not addressed in his paper. UTT is also a more sophisticated type theory because it includes an impredicative universe and a general class of inductive types.

An advantage of Coquand's proof over ours is that the technique of using a bisimulation may be needed for calculi with more sophisticated equalities between terms and types, although we are able to avoid using it for η-equality. For example, one approach to systems with a reduction which is known not to have the Church–Rosser property could be to define a final equality on terms in normal form, following the semantic explanation of the meaning of a type (see Section 1.1). UTT^S could be modified to include such an added equality, as we shall discuss in Section 7.2.

It may also be simpler to use a reduction to weak head normal form rather than reduction to full normal form. Several of the proofs by induction on derivations in UTT^S make extensive use of the Generation lemma, and the reductions under the Π-binder in the rules (W-E$_\forall$) and (W-E$^X[\bar{\Theta}]$) make these proofs even more indirect. This will also affect our soundness proof, because we shall always need to normalize types when it might again be enough to consider the weak head normal form.

We could instead have presented a system which included context reduction to normal form in the judgements: for example, the judgement for context validity would be that $\Gamma \rightarrow^{\mathrm{nf}} \Delta$. One case of the Generation lemma for this system would be that if $\Gamma, x{:}A \rightarrow^{\mathrm{nf}} \Delta, x{:}B$ then there is a subderivation of $\Gamma; A \rightarrow^{\mathrm{nf}} \Delta; B$, which is better than the lemma we are able to prove because it mentions the particular B in the subderivation. It also seems that a proof of Reduction with reduction in the context is more elegant, because we can give a direct proof instead of appealing to Context Replacement. However, giving such a presentation would mean that we would have to formalize reduction in the context, and the lemmas we prove about reduction would be correspondingly more difficult. We have therefore chosen to use the simpler system which only allows reduction in the subject.

Chapter 5

A Set-Theoretic Model Construction

In this chapter we give a semantics for UTT in classical set theory. This semantics is based on Dybjer's semantics for Martin-Löf type theory [26]. Basically, types are interpreted by their intuitive set-theoretic counterparts, with the exception of propositions which are only either true or false. Hence functions will be set-theoretic functions, the natural numbers will be the natural numbers, and in general inductive types will be a least fixed-point of the inductive equation for that type.

The set-theoretic model we give in this chapter will be important in the next chapter, where we show the soundness of UTT^S for UTT. We will use the model to give an induction principle on types which it seems we cannot justify syntactically.

In calculi with dependent types, such as UTT, terms and kinds are introduced simultaneously, which means that for all but the simplest interpretations the notions associated with the model theory are more complicated. In the case of the notion of environment, we need to know what an environment is before we can define the interpretations for terms and kinds. Once we have the interpretation of kinds, to prove soundness we can restrict our attention to derivable judgements and the environments which satisfy the context of those judgements. This is in contrast to the simply typed lambda calculus, as described by Mitchell [62], where terms and types are separate syntactically so the possible values for types can be introduced before the interpretations for terms, and environments are simply maps from variables to the union of all possible values for types. The interpretation of contexts is given after terms and kinds have been interpreted, and essentially the interpretation of a context Γ is an environment for the domain of Γ such that the value of the environment at each variable in the domain of Γ is in the interpretation of its kind. This is the notion of satisfaction of contexts introduced for example by Mitchell.

Set theory extended by an axiom stating the existence of an inaccessible cardinal gives an interpretation for the set theory without this axiom, by assuming that there is a set larger than any set which can be formed using the usual set forming operations such as union and powerset. Once we add this larger set, all of the original sets can be formed in the same way, and forming sets using just the original axioms always results in a set smaller than the inaccessible cardinal, but we can also form new large sets by applying the set forming operations to the inaccessible cardinal. Similarly, in type theory extended by a universe, the universe can be seen as containing a "smaller" version of the type theory, with all of the type-theoretic operations applying to objects within the universe to form new objects in the universe, and to the universe to form new "larger" types. Our interpretation follows this intuition by using inaccessible cardinals to interpret the universe and the kind Type.

Our approach differs from Dybjer's in several respects. First, since our type theory is expressed using the Logical Framework and includes an impredicative universe of propositions, we need to provide an interpretation for these constructs. Secondly, we do not consider the generalization of inductive types to inductive families of types. Thirdly, Dybjer interprets the inductive types using Aczel's rule sets [1], whereas we give an interpretation based on a least fixed-point construction and transfinite induction. These two interpretations are merely different presentations of essentially the same construct.

Lastly, Dybjer gives three possible interpretations for universes in his paper, one of which we have used here. The interpretation we use is to interpret the type-theoretic universe as a set-theoretic universe. Dybjer's second interpretation interprets the universe as the set inductively generated by the type constructors, similar to the inductive types. These two interpretations are unable to include an interpretation of the elimination rule for the universe. His last interpretation assigns to the universe the small set of names formed by the constructors, and only the lifting operator is required to be large. This corresponds more closely to our type-theoretic understanding of a universe, where the universe is inductively generated by its introduction rules.

Following our discussion in Section 4.4, we do not interpret the type theory UTT but instead the theory UTT^-, which does not include rules for substitutions.

5.1 The Interpretation

We assume the existence of three strongly inaccessible cardinals κ_0, κ_1 and κ_2, with $\kappa_0 < \kappa_1 < \kappa_2$.

We shall assume the usual operations on cardinals [82,42].

We recall from Section 4.6 that an environment for a set S is simply a function from a finite set of variables to S.

We shall call an environment for κ_2 a *set environment.*

Let $\{*\}$ be a distinguished one-element set.

Let A be a set and $B(x)$ be a family of sets indexed by $x \in A$. We write $\Pi x \in A.B(x)$ for the set of maps from elements $x \in A$ to $B(x)$, and $\Sigma x \in A.B(x)$ for the set of pairs of elements (x, y) with $x \in A$ and $y \in B(x)$. We also write $\lambda x \in A.f(x)$ for the map from elements $x \in A$ to $B(x)$, where $f(x) \in B(x)$ for each $x \in A$.

We recall that the *cumulative hierarchy* for an ordinal α, notation V_α, is defined by induction on α:

$$\begin{aligned} V_0 &=_{\mathrm{df}} \emptyset \\ V_{\mathrm{succ}(\alpha)} &=_{\mathrm{df}} \mathcal{P}(V_\alpha) \\ V_\alpha &=_{\mathrm{df}} \bigcup_{\beta<\alpha} V_\beta \quad \text{(for limit } \alpha) \end{aligned}$$

This definition can be found in standard books on set theory [82].

Our proof uses the technique of defining a partial interpretation for terms and showing that this interpretation is total for terms which are well-typed. This technique is introduced by Streicher [75]. The idea is that we can require certain conditions to hold in order for the interpretation to be defined, which we know must be true from the structure of derivations. For example, we require in the definition of the interpretation of application $M_1(M_2)$ (the most problematic case) that the interpretation of M_1 must be a set-theoretic function and that the interpretation of M_2 must be in the domain of M_1. When we examine the rule of inference for application in the final soundness proof, these conditions must always hold by inductive hypothesis. Hence the interpretation of application is well defined for well-typed terms.

Definition 5.1.1 (Interpretation) *Let ρ be a set environment. We define the interpretation of terms M and kinds A under ρ, $[\![M]\!](\rho)$ and $[\![A]\!](\rho)$, by induction on the structure of M and A, using the induction principle of Section 4.3:*

- $[\![x]\!](\rho) =_{\mathrm{df}} \rho(x)$ *if $x \in dom(\rho)$.*
- $[\![(x{:}A_1)A_2]\!](\rho) =_{\mathrm{df}} \Pi v \in [\![A_1]\!](\rho).[\![A_2]\!](\rho[x := v])$
  - *if $[\![A_1]\!](\rho)$ is defined and*
  - *$[\![A_2]\!](\rho[x := v])$ is defined for any $v \in [\![A_1]\!](\rho)$ and some $x \notin dom(\rho)$.*
- $[\![[x{:}A_1]M_0]\!](\rho) =_{\mathrm{df}} \lambda v \in [\![A_1]\!](\rho).[\![M_0]\!](\rho[x := v])$
  - *if $[\![A_1]\!](\rho)$ is defined and*
  - *$[\![M_0]\!](\rho[x := v])$ is defined for any $v \in [\![A_1]\!](\rho)$ and some $x \notin dom(\rho)$.*

- $[\![M_1(M_2)]\!](\rho) =_{\mathrm{df}} [\![M_1]\!](\rho)([\![M_2]\!](\rho))$
  - *if* $[\![M_1]\!](\rho)$ *and* $[\![M_2]\!](\rho)$ *are defined,*
  - $[\![M_1]\!](\rho)$ *is a function and*
  - $[\![M_2]\!](\rho) \in dom([\![M_1]\!](\rho))$.

- $[\![\mathrm{Type}]\!](\rho) =_{\mathrm{df}} V_{\kappa_1}$.

- $[\![\mathrm{El}(M_0)]\!](\rho) =_{\mathrm{df}} [\![M_0]\!](\rho)$ *if* $[\![M_0]\!](\rho)$ *is defined.*

- $[\![\mathrm{Prop}]\!] =_{\mathrm{df}} \{\emptyset, \{*\}\}$.

- $[\![\mathrm{Prf}]\!] =_{\mathrm{df}} \lambda v \in [\![\mathrm{Prop}]\!].v$.

- $[\![\forall]\!] =_{\mathrm{df}} \lambda A \in [\![\mathrm{Type}]\!].\lambda P \in A \to [\![\mathrm{Prop}]\!].\begin{cases} \{*\} & \textit{if}\ \forall a \in A.P(a) = \{*\} \\ \emptyset & \textit{otherwise} \end{cases}.$

- $[\![\Lambda]\!] =_{\mathrm{df}} \lambda A \in [\![\mathrm{Type}]\!].\lambda P \in A \to [\![\mathrm{Prop}]\!].\lambda g \in (\Pi a \in A.[\![\mathrm{Prf}]\!](P(a))).*$.

- $$\begin{aligned} [\![\mathrm{E}_\forall]\!] \quad =_{\mathrm{df}} \quad & \lambda A \in [\![\mathrm{Type}]\!].\lambda P \in A \to [\![\mathrm{Prop}]\!]. \\ & \lambda R \in [\![\mathrm{Prf}]\!]([\![\forall]\!](A, P)) \to [\![\mathrm{Prop}]\!]. \\ & \lambda f \in (\Pi g \in (\Pi a \in A.[\![\mathrm{Prf}]\!](P(a))).[\![\mathrm{Prf}]\!](R([\![\Lambda]\!](A, P, g)))). \\ & \lambda z \in [\![\mathrm{Prf}]\!]([\![\forall]\!](A, P)). \\ & \quad f(\lambda a \in A.*) \end{aligned}$$

- $[\![\mathrm{U}]\!] =_{\mathrm{df}} V_{\kappa_0}$.

- $[\![\mathrm{T}]\!] =_{\mathrm{df}} \lambda v \in [\![\mathrm{U}]\!].v$.

- $[\![\mathrm{prop}]\!] =_{\mathrm{df}} \{\emptyset, \{*\}\}$.

- $[\![\mathrm{prf}]\!] =_{\mathrm{df}} \lambda v \in [\![\mathrm{Prop}]\!].v$.

Remarks

- We recall that terms which are syntactically equivalent are equal. Therefore, in the above definition for terms with binders, we can always choose a subterm which has a bound variable distinct from the domain of the set environment ρ.

- In the rest of the text we omit writing $[\![\mathrm{Prf}]\!]$ and $[\![\mathrm{T}]\!]$ since both are the identity.

- For the definitions of $[\![\Lambda]\!]$ and $[\![\mathrm{E}_\forall]\!]$, the set $\Pi a \in A.P(a)$ is the empty set if $P(a)$ is the empty set for some $a \in A$.

We define several functions which allow us to define the set-theoretic semantics of inductive data types by least fixed-point constructions.

Definition 5.1.2 *Let κ be a strongly inaccessible cardinal, let ρ be a set environment and suppose* $\mathrm{PS{\scriptsize CH}}_X(\bar{\Theta})$*. We define a partial function on sets relative to κ, $\bar{\Theta}^{\#}_{\rho} \in V_\kappa \to V_\kappa$, by induction on* $\mathrm{PS{\scriptsize CH}}_X(\bar{\Theta})$*:*

- $\Theta_1, \ldots, \Theta_n{}^{\#}_{\rho}(S) =_{\mathrm{df}} \Theta_1{}^{\#}_{\rho}(S) + \cdots + \Theta_n{}^{\#}_{\rho}(S)$ *if* $\Theta_1{}^{\#}_{\rho}, \ldots, \Theta_n{}^{\#}_{\rho}$ *are defined.*

- $X^{\#}_{\rho}(S) =_{\mathrm{df}} \{*\}$.

- $(\Phi)\Theta^{\#}_{\rho}(S) =_{\mathrm{df}} \Phi^{\#}_{\rho}(S) \times \Theta^{\#}_{\rho}(S)$ *if* $\Phi^{\#}_{\rho}$ *and* $\Theta^{\#}_{\rho}$ *are defined.*

- $(x{:}A)\Theta^{\#}_{\rho}(S) =_{\mathrm{df}} \Sigma a \in [\![A]\!](\rho).\Theta^{\#}_{\rho[x:=a]}(S)$

  - *if* $[\![A]\!](\rho)$ *defined,*

  - $[\![A]\!](\rho) \in V_\kappa$ *and*

  - $\Theta^{\#}_{\rho[x:=a]}$ *is defined for any* $a \in [\![A]\!](\rho)$.

- $\Phi^{\#}_{\rho}(S) =_{\mathrm{df}} [\![\Phi]\!](\rho[X := S])$ *if* $[\![\Phi]\!](\rho[X := S])$ *is defined.*

Lemma 5.1.3 *Suppose* $\mathrm{PS}\textsc{ch}_X(\Theta)$*,* ρ *is a set environment, and* S *is a set. Then we have* $[\![\Theta]\!](\rho[X := S]) \cong \Theta^{\#}_{\rho}(S) \to S$.

Proof We give the obvious maps in each direction:

$$\begin{array}{rcl}
\Theta^{\rightarrow} & =_{\mathrm{df}} & \lambda f \in [\![\Theta]\!](\rho[X := S]).\lambda{<}\bar{M}, *{>}.f\bar{M} \\
 & \in & [\![\Theta]\!](\rho[X := S]) \to \Theta^{\#}_{\rho}(S) \to S \\
\Theta^{\leftarrow} & =_{\mathrm{df}} & \lambda f \in \Theta^{\#}_{\rho}(S) \to S.\lambda\bar{M}.f{<}\bar{M}, *{>} \\
 & \in & (\Theta^{\#}_{\rho}(S) \to S) \to [\![\Theta]\!](\rho[X := S])
\end{array}$$

which trivially give an isomorphism. □

Lemma 5.1.4 (Monotonicity of $\bar{\Theta}^{\#}_{\rho}$) *Suppose* ρ *is a set environment. Then:*

- *If* $\mathrm{PS}\textsc{ch}_X(\bar{\Theta})$ *and* $\bar{\Theta}^{\#}_{\rho}$ *is defined then* $\bar{\Theta}^{\#}_{\rho}$ *is a monotone function on sets.*
- *If* $\mathrm{PS}\textsc{ch}_X(\Theta)$ *and* $\Theta^{\#}_{\rho}$ *is defined then* $\Theta^{\#}_{\rho}$ *is a monotone function on sets.*
- *If* $\mathrm{PP}\textsc{os}_X(\Phi)$ *and* $\Phi^{\#}_{\rho}$ *is defined then* $\Phi^{\#}_{\rho}$ *is a monotone function on sets.*

Proof By induction on $\mathrm{PS}\textsc{ch}_X(\bar{\Theta})$, using the monotonicity of the set-theoretic operators $+$ and $\times$, and Σ and Π on the right-hand side. □

Lemma 5.1.5 $card(X) = card(A \to X)$ *if* $card(X) \geq card(2^{card(A)})$.

Proof By transfinite induction on A [42]. □

Lemma 5.1.6 (Boundedness of $\bar{\Theta}^{\#}_{\rho}$) *Let* ρ *be a set environment, suppose* $\mathrm{PS}\textsc{ch}_X(\bar{\Theta})$ *and assume that* $\bar{\Theta}^{\#}_{\rho}$ *is defined relative to* κ*. Then there is an ordinal* $\alpha_0 < \kappa$ *such that* $\bar{\Theta}^{\#}_{\rho}(S) \in V_{\alpha_0} \to V_{\alpha_0}$.

Proof First, we construct the bound α_0 using only set-theoretic operators on sets smaller than κ, so we are guaranteed that $\alpha_0 < \kappa$ because κ is strongly inaccessible.

We construct α_0 by induction on $\mathrm{PS}\textsc{ch}_X(\bar{\Theta})$:

- $\Theta_1, \ldots, \Theta_n$. Let α_0 be $\alpha_{\Theta_1} + \cdots + \alpha_{\Theta_n}$.

- $\Theta \equiv X$. Then we take α_0 to be 1.

- $\Theta \equiv (x{:}A)\Theta_0$. By inductive hypothesis we know that for each $a \in [\![A]\!](\rho)$ there exists an α_a such that ${\Theta_0}^{\#}_{\rho}(X) \in V_{\alpha_a} \to V_{\alpha_a}$. Let $\alpha_0 =_{\mathrm{df}} \beta \times \bigsqcup_{a \in [\![A]\!](\rho)} \alpha_a$, where β is the least limit ordinal such that $A \in V_\beta$.

- $\Theta \equiv (\Phi)\Theta_0$. Let α_0 be $\alpha_\Phi \times \alpha_{\Theta_0}$.

- $\Phi \equiv (x_1{:}A_1)\ldots(x_n{:}A_n)X$. Then

$$
\begin{aligned}
& [\![(x_1{:}A_1)\ldots(x_n{:}A_n)X]\!](\rho[X := X]) \\
= \;& \Pi v_1 \in [\![A_1]\!](\rho). \ldots \Pi v_n \in [\![A_n]\!](\rho).X \\
\cong \;& (\Sigma v_1 \in [\![A_1]\!](\rho). \ldots [\![A_n]\!](\rho[x_1 := v_1]\ldots[x_{n-1} := v_{n-1}])) \to X
\end{aligned}
$$

  Let A be the domain of the function space. We can take the least ordinal α_0 such that $A \subseteq V_{\alpha_0}$ and $card(A) < card(\alpha_0)$. This is because if $f \in A \to V_{\alpha_0}$ then for each $a \in A$ we know that $(a, f(a)) \in V_{\beta_a}$ for some $\beta_a < \alpha_0$. Let $\beta =_{\mathrm{df}} \sqcup_{a \in A} \beta_a$ which is less than α_0. Hence $f = \{(a, f(a)) \mid a \in A\} \subseteq V_\beta$, so $f \in V_{\alpha_0}$.

$\square$

Definition 5.1.7 *Let κ be a strongly inaccessible cardinal, let $F \in V_\kappa \to V_\kappa$ and let α be an ordinal. We define F^α as:*

$$
\begin{aligned}
F^0 &=_{\mathrm{df}} \emptyset \\
F^{\mathrm{succ}(\alpha)} &=_{\mathrm{df}} F(F^\alpha) \\
F^\alpha &=_{\mathrm{df}} \cup_{\beta<\alpha} F^\beta \qquad \textit{for limit } \alpha
\end{aligned}
$$

Proposition 5.1.8 *If $F \in V_\alpha \to V_\alpha$ and F is a monotone function on V_α then there is an ordinal α_0 such that F^{α_0} is a least fixed-point for F.*

Proof Standard [1]. $\square$

Definition 5.1.9 (Interpretation of Inductive Types) *Let $\bar{\Theta}^{\#}_{\rho}(S)$ be defined relative to some κ, and let α_0 be the ordinal such that $(\bar{\Theta}^{\#}_{\rho})^{\alpha_0}$ is a least fixed-point for $\bar{\Theta}^{\#}_{\rho}$. Then we define the interpretation of an inductive type as:*

$$\begin{array}{rcll} [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho) & =_{\mathrm{df}} & (\bar{\Theta}^{\#}_{\rho})^{\alpha_0}(\emptyset) & \textit{if } \kappa \leq \kappa_1 \\ [\![\mu^X[\bar{\Theta}]]\!](\rho) & =_{\mathrm{df}} & (\bar{\Theta}^{\#}_{\rho})^{\alpha_0}(\emptyset) & \textit{if } \kappa \leq \kappa_0 \end{array}$$

Definition 5.1.10 *The interpretation of the introduction rule for an inductive type is given as*

$$[\![\iota_i^X[\bar{\Theta}]]\!](\rho) =_{\mathrm{df}} \Theta_i^{\leftarrow} \circ \mathrm{intro}_{\bar{\Theta}^{\#}_{\rho}} \circ in_i$$

where $\mathrm{intro}_{\bar{\Theta}^{\#}_{\rho}}$ *is the map from $\bar{\Theta}^{\#}_{\rho}((\bar{\Theta}^{\#}_{\rho})^{\alpha_0})$ to $(\bar{\Theta}^{\#}_{\rho})^{\alpha_0}$.*

Definition 5.1.11 *Suppose*

$$\begin{array}{rcl} C & \in & [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho) \to [\![\mathrm{Type}]\!]. \\ f_1 & \in & [\![\Theta_1^{\circ}(\mathcal{M}^X[\bar{\Theta}], C, \iota_1^X[\bar{\Theta}])]\!](\rho[C := C]) \\ \ldots & & \\ f_n & \in & [\![\Theta_n^{\circ}(\mathcal{M}^X[\bar{\Theta}], C, \iota_n^X[\bar{\Theta}])]\!](\rho[C := C]) \end{array}$$

Let α_0 be the ordinal which defines the least fixed-point for $\bar{\Theta}^{\#}_{\rho}$.

We define $R^{\alpha}(M)$, for $M \in (\bar{\Theta}^{\#}_{\rho})^{\alpha}$, by transfinite induction on α:

- $\alpha = 0$. *Then $R^{\alpha}(M) =_{\mathrm{df}} \epsilon$, the unique function from the empty set $\emptyset$.*

- $\alpha = \mathrm{succ}(\beta)$. *Then*

$$R^{\mathrm{succ}(\beta)}(\mathrm{intro}_i(N_1, \ldots, N_m)) \quad =_{\mathrm{df}} \quad f_i(N_1, \ldots, N_m, \Phi^*_{i_1}, \ldots)$$

if

$$\begin{array}{rcl} \Phi^*_{i_j} & =_{\mathrm{df}} & [\![\Phi^{\natural}_{i_j}[A, C, f, z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][f := R^{\beta}][z := N_{i_j}]) \\ & \in & [\![\Phi^{\circ}_{i_j}[A, C, z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][z := N_{i_j}]) \end{array}$$

for $1 \leq j \leq k$, where we also assume that both of these interpretations are defined.

- *Limit α. Then*

$$R^{\alpha}(M) =_{\mathrm{df}} R^{\gamma}(M)$$

where $\gamma =_{\mathrm{df}} \cap\{\beta \mid M \in (\bar{\Theta}^{\#}_{\rho})^{\beta}\}$, if $R^{\gamma}(M)$ is defined.

Then

$$[\![\mathrm{E}^{X}[\bar{\Theta}]]\!](\rho)(C, f_1, \ldots, f_n) =_{\mathrm{df}} \lambda M \in [\![\mathcal{M}^{X}[\bar{\Theta}]]\!](\rho).R^{\alpha_0}(M)$$

We have used the induction principle defined in Section 4.3 to know that $[\![\Phi^{\natural}_{i_j}[A, C, f, z]]\!](\rho')$ will already have been defined by inductive hypothesis if it has a value. Since the interpretation is a partial function, it may of course fail to be defined because the conditions required have not been met.

Definition 5.1.12 *We define the interpretation of a context Γ by induction on the structure of Γ:*

- $[\![()]\!] =_{\mathrm{df}} \{\epsilon\}$.
- $[\![\Gamma, x{:}A]\!] =_{\mathrm{df}} \{\rho[x := v] \mid \rho \in [\![\Gamma]\!]$ *and* $v \in [\![A]\!](\rho)\}$
  - *if $[\![\Gamma]\!]$ is defined and*
  - *$[\![A]\!](\rho)$ is defined for any $\rho \in [\![\Gamma]\!]$.*

5.2 Properties of the Interpretation

In this section we establish the basic properties of the interpretation, in preparation for the final proof of soundness. These properties will be proved by the same induction principle which we used to define the interpretation, the alternative induction principle on terms defined in Section 4.3.

Definition 5.2.1

- *a and b are* Kleene equal, *notation $a \simeq b$, if a is defined if and only if b is defined, and if they are both defined then $a = b$.*

- *a is* Kleene greater than b, *notation* $a \succ b$, *if whenever* a *is defined then* b *is defined and* $a = b$.

The following lemma says that although we have defined the interpretation of terms with binders by choosing a particular fresh variable, we can treat this as though we used an arbitrary fresh variable. This will be important in the proof of the substitution lemma.

Lemma 5.2.2 *Suppose* γ *is an environment for variables and* ρ *is a set environment. Then* $[\![M]\!](\rho \circ \gamma) \simeq [\![\widehat{\gamma}(M)]\!](\rho)$, *where* $\widehat{\gamma}(M)$ *is for* γ *as a substitution.*

Proof By induction on the structure of M. We only consider the variable and Π cases:

- $M \equiv x$. Then $[\![x]\!](\rho \circ \gamma) \simeq \rho(\gamma(x)) \simeq [\![\gamma(x)]\!](\rho) \simeq [\![\widehat{\gamma}(x)]\!](\rho)$.

- $A \equiv (x{:}A_1)A_2$. Suppose $[\![(x{:}A_1)A_2]\!](\rho \circ \gamma)$ is defined. Then $[\![A_1]\!](\rho \circ \gamma)$ is defined and there is an $x \notin dom(\gamma)$ such that $[\![A_2]\!]((\rho \circ \gamma)[x := v])$ is defined for any $v \in [\![A_1]\!](\rho \circ \gamma)$. By inductive hypothesis we know that $[\![A_1]\!](\rho \circ \gamma) = [\![\widehat{\gamma}(A_1)]\!](\rho)$.

  Choose some $y \notin dom(\rho)$, then by inductive hypothesis

  $$\begin{array}{ll} & [\![A_2]\!]((\rho \circ \gamma)[x := v]) = [\![A_2]\!](\rho[y := v] \circ \gamma[x := y]) \\ \succ & [\![\widehat{\gamma[x := y]}(A_2)]\!](\rho[y := v]) \end{array}$$

  for any $v \in [\![\widehat{\gamma}(A_1)]\!](\rho)$, so

  $$\begin{array}{rcl} [\![(x{:}A_1)A_2]\!](\rho \circ \gamma) & = & [\![(y{:}\widehat{\gamma}(A_1))\widehat{\gamma[x := y]}(A_2)]\!](\rho) \\ & = & [\![\widehat{\gamma}((x{:}A_1)A_2)]\!](\rho) \end{array}$$

  The other direction follows similarly.

□

The above lemma is important in our proof of the substitution lemma. We need it because in performing a substitution $[N/x]M$, the terms M and N could both have binders for which the interpretation has chosen the same fresh variable. For example, in the substitution $[[x{:}A_1]M_0/y]([x{:}A_1]y)$, we need to interpret $[x{:}A_1]M_0$ where x is already in the environment, so if the interpretation of $[x{:}A_1]M_0$ has chosen x as the fresh variable then there will be a clash. The above lemma says that we can interpret the substituted term by choosing a different variable.

Lemma 5.2.3 *Let ρ_1 and ρ_2 be two set environments. If for all $x \in \mathrm{FV}(M)$ we have that $\rho_1(x) = \rho_2(x)$ then $[\![M]\!](\rho_1) \simeq [\![M]\!](\rho_2)$.*

Proof We only need to show that $[\![M]\!](\rho_1) \succ [\![M]\!](\rho_2)$ under the conditions mentioned. This follows by induction on the structure of terms and types, where all cases except binders are straightforward.

We illustrate the case of $[x{:}A_1]M_0$. Suppose $[\![[x{:}A_1]M_0]\!](\rho_1)$ is defined. Then $[\![A_1]\!](\rho_1)$ is defined and $[\![M_0]\!](\rho_1[x := v])$ is defined for any $v \in [\![A_1]\!](\rho_1)$ and for some $x \notin dom(\rho_1)$. By inductive hypothesis $[\![A_1]\!](\rho_1) = [\![A_1]\!](\rho_2)$. Choose $y \notin dom(\rho_1) \cup dom(\rho_2)$. Then by Lemma 5.2.2 and inductive hypothesis

$$\begin{array}{rcl} [\![M_0]\!](\rho_1[x := v]) & = & [\![M_0]\!](\rho_1[y := v] \circ \mathrm{id}_{dom(\rho_1)}[x := y]) \\ & \simeq & [\![\widehat{\mathrm{id}_{dom(\rho_1)}[x} := y](M_0)]\!](\rho_1[y := v]) \\ & = & [\![\widehat{\mathrm{id}_{dom(\rho_2)}[x} := y](M_0)]\!](\rho_2[y := v]) \\ & \simeq & [\![M_0]\!](\rho_2[y := v] \circ \mathrm{id}_{dom(\rho_2)}[x := y]) = [\![M_0]\!](\rho_2[x := v]) \end{array}$$

for any $v \in [\![A_1]\!](\rho_2)$, so $[\![[x{:}A_1]M_0]\!](\rho_2)$ is defined and $[\![[x{:}A_1]M_0]\!](\rho_1) = [\![[x{:}A_1]M_0]\!](\rho_2)$. $\square$

Lemma 5.2.4 (Substitution) *Let $\rho = \rho_0\rho_1$ be a set environment. Then*

$$[\![M]\!](\rho[z := [\![N]\!](\rho_0)]) \succ [\![[N/z]M]\!](\rho)$$

Proof By induction on the structure of terms and kinds. We show several cases.

- $M \equiv x$. Then $x \not\equiv z$ and $[\![x]\!](\rho[z := [\![N]\!](\rho_0)])$ defined imply that

$$[\![x]\!](\rho[z := [\![N]\!](\rho_0)]) = \rho[z := [\![N]\!](\rho_0)](x) = \rho(x) = [\![x]\!](\rho)$$

- $M \equiv z$. Then $[\![z]\!](\rho[z := [\![N]\!](\rho_0)]) = [\![N]\!](\rho_0) \simeq [\![N]\!](\rho)$ by definition and Lemma 5.2.3.

- $A \equiv (x{:}A_1)A_2$. If $[\![(x{:}A_1)A_2]\!](\rho[z := [\![N]\!](\rho_0)])$ is defined then $[\![A_1]\!](\rho[z := [\![N]\!](\rho_0)])$ is defined and $[\![A_2]\!](\rho[z := [\![N]\!](\rho_0)][x := v])$ is defined for some $x \notin dom(\rho)$ and any $v \in [\![A_1]\!](\rho[z := [\![N]\!](\rho_0)])$. By inductive hypothesis we have that $[\![[N/z]A_1]\!](\rho)$ is defined and equal to $[\![A_1]\!](\rho[z := [\![N]\!](\rho_0)])$ and

$$\begin{aligned}[\![A_2]\!](\rho[z := [\![N]\!](\rho_0)][x := v]) &= [\![A_2]\!](\rho[x := v][z := [\![N]\!](\rho_0)]) \\ &= [\![[N/z]A_2]\!](\rho[x := v])\end{aligned}$$

for $v \in [\![[N/z]A_1]\!](\rho)$.

- $M \equiv \kappa^X[\bar{\Theta}]$, where $\kappa \in \{\mathcal{M}, \iota_i, \mathrm{E}, \mu\}$. $\mathcal{M}^X[\bar{\Theta}]$, $\iota_i^X[\bar{\Theta}]$ and $\mu^X[\bar{\Theta}]$ follow straightforwardly by inductive hypothesis. $\mathrm{E}^X[\bar{\Theta}]$ follows by transfinite induction and the inductive hypothesis.

- M is a constant. Straightforward.

□

5.3 Soundness

Lemma 5.3.1 (Soundness) *For any derivable judgement $\Gamma \vdash^- J$ we know that $[\![\Gamma]\!]$ is defined. Furthermore:*

- *If $\Gamma \vdash^- A$ kind and $\rho \in [\![\Gamma]\!]$ then $[\![A]\!](\rho)$ is defined.*

- *If $\Gamma \vdash^- M : A$ and $\rho \in [\![\Gamma]\!]$ then $[\![A]\!](\rho)$ and $[\![M]\!](\rho)$ are defined and $[\![M]\!](\rho) \in [\![A]\!](\rho)$.*

- *If $\Gamma \vdash^- A = B$ and $\rho \in [\![\Gamma]\!]$ then $[\![A]\!](\rho)$ and $[\![B]\!](\rho)$ are defined and $[\![A]\!](\rho) = [\![B]\!](\rho)$.*

- *If* $\Gamma \vdash^- M = N : A$ *and* $\rho \in [\![\Gamma]\!]$ *then* $[\![M]\!](\rho)$, $[\![N]\!](\rho)$ *and* $[\![A]\!](\rho)$ *are defined,* $[\![M]\!](\rho) = [\![N]\!](\rho)$, $[\![M]\!](\rho) \in [\![A]\!](\rho)$ *and* $[\![N]\!](\rho) \in [\![A]\!](\rho)$.

Proof By induction on derivations in UTT^-:

- (Emp). Immediate.

- $(Weak)$. By inductive hypothesis $[\![\Gamma]\!]$ is defined and if $\rho \in [\![\Gamma]\!]$ then $[\![A]\!](\rho)$ is defined, so $[\![\Gamma, x{:}A]\!]$ is defined.

- (Var). By inductive hypothesis $[\![\Gamma_0, x{:}A, \Gamma_1]\!]$ is defined, and by definition $\rho \in [\![\Gamma_0, x{:}A, \Gamma_1]\!]$ implies that there are N and $\rho_0 \in [\![\Gamma_0]\!]$ such that $N \in [\![A]\!](\rho_0)$. By Contexts (Lemma 4.5.3) we know that each $x_i \in dom(\Gamma_0, x{:}A, \Gamma_1)$ is distinct and $\mathrm{FV}(A) \subseteq dom(\Gamma_0)$, so $\rho(x) = N \in [\![A]\!](\rho_0) = [\![A]\!](\rho)$ by Lemma 5.2.3.

- (Π). By inductive hypothesis $[\![\Gamma, x{:}A_1]\!]$ is defined, so by definition $[\![\Gamma]\!]$ is defined and $[\![A_1]\!](\rho)$ is defined if $\rho \in [\![\Gamma]\!]$. Therefore, if $v \in [\![A_1]\!](\rho)$ then $\rho[x := v] \in [\![\Gamma, x{:}A_1]\!]$, so by inductive hypothesis $[\![A_2]\!](\rho[x := v])$ is defined, which means that $[\![(x{:}A_1)A_2]\!](\rho)$ is defined.

- (λ). By inductive hypothesis and the definition of the interpretation of contexts.

- (App). By inductive hypothesis if $\rho \in [\![\Gamma]\!]$ then $[\![M_1]\!](\rho)$ and $[\![(x{:}A_1)A_2]\!](\rho)$ are defined and $[\![M_1]\!](\rho) \in [\![(x{:}A_1)A_2]\!](\rho)$, and also $[\![M_2]\!](\rho)$ and $[\![A_1]\!](\rho)$ are defined and $[\![M_2]\!](\rho) \in [\![A_1]\!](\rho)$. Therefore

$$\begin{aligned} & [\![M_1(M_2)]\!](\rho) = [\![M_1]\!](\rho)([\![M_2]\!](\rho)) \\ \in \quad & [\![A_2]\!](\rho[x := [\![M_2]\!](\rho)]) = [\![[M_2/x]A_2]\!](\rho) \end{aligned}$$

by definition and Lemma 5.2.4.

- (Type). Immediate.

- (El). By inductive hypothesis if $\rho \in [\![\Gamma]\!]$ then $[\![A]\!](\rho) \in [\![\mathrm{Type}]\!]$, so $[\![\mathrm{El}(A)]\!](\rho)$ is defined.

- $(=T)$. By inductive hypothesis if $\rho \in [\![\Gamma]\!]$ then $[\![M]\!](\rho)$, $[\![A]\!](\rho)$, and $[\![B]\!](\rho)$ are defined, $[\![M]\!](\rho) \in [\![A]\!](\rho)$ and $[\![A]\!](\rho) = [\![B]\!](\rho)$. Hence $[\![M]\!](\rho) \in [\![B]\!](\rho)$.

- $(=R)$. Similar to the previous case.

- (*KRefl*)(*KSym*)(*KTrans*)(*Refl*)(*Sym*)(*Trans*). By inductive hypothesis.

- (Π-*Eq*). By reasoning similar to (Π).

- (λ-*Eq*). By inductive hypothesis $[\![\Gamma, x{:}A_1]\!]$ is defined, so by definition $[\![\Gamma]\!]$ is defined and $[\![A_1]\!](\rho)$ is defined if $\rho \in [\![\Gamma]\!]$. We also know by inductive hypothesis that $[\![B_1]\!](\rho)$ is defined and that $[\![A_1]\!](\rho) = [\![B_1]\!](\rho)$ if $\rho \in [\![\Gamma]\!]$. Therefore if $v \in [\![A_1]\!](\rho)$ then $\rho[x := v] \in [\![\Gamma, x{:}A_1]\!]$ and by inductive hypothesis $[\![M_0]\!](\rho[x := v])$ and $[\![N_0]\!](\rho[x := v])$ are defined and equal. Hence by extensionality

$$\lambda v \in [\![A_1]\!](\rho).[\![M_0]\!](\rho[x := v]) = \lambda v \in [\![B_1]\!](\rho).[\![N_0]\!](\rho[x := v])$$

  and by reasoning similar to (λ) they are both in $[\![(x{:}A_1)A_2]\!](\rho)$.

- (*App-Eq*). By reasoning similar to (*App*).

- (β). By inductive hypothesis $[\![\Gamma, x{:}A_1]\!]$ is defined, so by definition $[\![\Gamma]\!]$ is defined and $[\![A_1]\!](\rho)$ is defined for $\rho \in [\![\Gamma]\!]$. Also by inductive hypothesis, $\rho \in [\![\Gamma]\!]$ implies that $[\![M_2]\!](\rho)$ is defined and in $[\![A_1]\!](\rho)$, so $\rho[x := [\![M_2]\!](\rho)] \in [\![\Gamma, x{:}A_1]\!]$. By a final application of the inductive hypothesis we have that $[\![M_0]\!](\rho[x := [\![M_2]\!](\rho)])$ and $[\![A_2]\!](\rho[x := [\![M_2]\!](\rho)])$ are defined and

$$[\![M_0]\!](\rho[x := [\![M_2]\!](\rho)]) \in [\![A_2]\!](\rho[x := [\![M_2]\!](\rho)])$$

so by definition of the interpretation and Lemma 5.2.4 we have that

$$\begin{aligned}
& [\![([x{:}A_1]M_0)(M_2)]\!](\rho) \\
= \;& (\lambda v \in [\![A_1]\!](\rho).[\![M_0]\!](\rho[x := v]))([\![M_2]\!](\rho)) \\
= \;& [\![M_0]\!](\rho)\rho[x := [\![M_2]\!](\rho)] \\
= \;& [\![[M_2/x]M_0]\!](\rho)
\end{aligned}$$

and $[\![A_2]\!](\rho[x := [\![M_2]\!](\rho)]) = [\![[M_2/x]A_2]\!](\rho)$.

- (El-Eq). By inductive hypothesis.

- (Prop). Immediate.

- (Prf). If $v \in \{\emptyset, \{*\}\}$ then $v \in V_\kappa$.

- ($\forall$). Both possible values are in $\{\emptyset, \{*\}\}$.

- (Λ). Suppose

$$\begin{aligned}
\rho &\in [\![\Gamma]\!] \\
A &\in [\![\mathrm{Type}]\!] \\
P &\in \Pi a \in A.[\![\mathrm{Prop}]\!] \\
g &\in \Pi a \in A.P(a)
\end{aligned}$$

  We show that $* \in [\![\forall]\!](A, P)$ by case analysis on $[\![\forall]\!](A, P)$:

  - $[\![\forall]\!](A, P) = \{*\}$, where for all $a \in A$ we have $P(a) = \{*\}$. Then $* \in \{*\} = [\![\forall]\!](A, P)$.

  - $[\![\forall]\!](A, P) = \emptyset$. Then there is an $a \in A$ such that $g(a) \in P(a) = \emptyset$, which is a contradiction.

- ($E_\forall$). Suppose

$$\begin{array}{rcl} \rho & \in & [\![\Gamma]\!] \\ A & \in & [\![\mathrm{Type}]\!] \\ P & \in & \Pi a \in A.[\![\mathrm{Prop}]\!] \\ R & \in & \Pi p \in [\![\forall]\!](A, P).[\![\mathrm{Prop}]\!] \\ f & \in & \Pi g \in (\Pi a \in A.P(a)).R([\![\Lambda]\!](A, P, g)) \\ z & \in & [\![\forall]\!](A, P) \end{array}$$

  Then by analyzing the possible values of $[\![\forall]\!](A, P)$ we show that $f(\lambda a \in A.*) \in R(z)$:

  - $[\![\forall]\!](A, P) = \{*\}$, where for all $a \in A$ we have $P(a) = \{*\}$. Then $(\lambda a \in A.*) \in \Pi a \in A.P(a)$ and

    $$f(\lambda a \in A.*) \in R([\![\forall]\!](A, P, \lambda a \in A.*)) = R(z)$$

    where the equality holds because $[\![\Lambda]\!](A, P, \lambda a \in A.*) = * = z$.
  - $[\![\forall]\!](A, P) = \emptyset$. Then $z \in [\![\forall]\!](A, P)$ is a contradiction.

- (Prop-Eq). Again suppose

$$\begin{array}{rcl} \rho & \in & [\![\Gamma]\!] \\ A & \in & [\![\mathrm{Type}]\!] \\ P & \in & \Pi a \in A.[\![\mathrm{Prop}]\!] \\ R & \in & \Pi p \in [\![\forall]\!](A, P).[\![\mathrm{Prop}]\!] \\ f & \in & \Pi g \in (\Pi a \in A.P(a)).R([\![\Lambda]\!](A, P, g)) \\ g & \in & \Pi a \in A.P(a) \end{array}$$

  Then

  $$[\![E_\forall]\!](A, P, R, f, [\![\Lambda]\!](A, P, g)) = f(g)$$

  follows from $g = \lambda a \in A.*$, which we show by analyzing the possible values of $[\![\forall]\!](A, P)$:

- $[\![\forall]\!](A, P) = \{*\}$, where for all $a \in A$ we have $P(a) = \{*\}$. Then for each $a \in A$ we have $g(a) = * \in \{*\}$, so by extensionality $g = \lambda a \in A.*$.
- $[\![\forall]\!](A, P) = \emptyset$. Then there is an $a \in A$ such that $g(a) \in P(a)$ and $P(a) = \emptyset$, which is a contradiction.

- $(\mathcal{M}^X[\bar{\Theta}])$. By inductive hypothesis we know that $[\![\Gamma']\!]$ is defined for all $(\Gamma', A) \in \textsc{Types}_\Gamma(\bar{\Theta})$, and we have that $[\![A]\!](\rho)$ is defined and in $[\![\mathrm{Type}]\!]$ for all $\rho \in [\![\Gamma']\!]$. Hence $[\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho) \in [\![\mathrm{Type}]\!]$.

- $(\iota_i^X[\bar{\Theta}])$. We know that $\mathrm{intro}_i \in {\Theta_i}_\rho^\#([\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)) \rightarrow [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)$, so

$$\bar{\Theta}^\rightarrow(\mathrm{intro}_i) \in [\![\Theta_i]\!](\rho[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)]) = [\![\Theta_i[\mathcal{M}^X[\bar{\Theta}]]]\!](\rho)$$

by Lemma 5.1.3 and Lemma 5.2.4.

- $(\mathrm{E}^X[\bar{\Theta}])$. Suppose

$$\begin{array}{rcl} \rho & \in & [\![\Gamma]\!] \\ C & \in & [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho) \rightarrow [\![\mathrm{Type}]\!] \\ f_1 & \in & [\![\Theta_1^\circ(\mathcal{M}^X[\bar{\Theta}], C, \iota_1^X[\bar{\Theta}])]\!](\rho[C := C]) \\ & \cdots & \\ f_n & \in & [\![\Theta_n^\circ(\mathcal{M}^X[\bar{\Theta}], C, \iota_n^X[\bar{\Theta}])]\!](\rho[C := C]) \end{array}$$

We show by transfinite induction on α that if $M \in (\bar{\Theta}_\rho^\#)^\alpha$ then $R^\alpha(M)$ is defined and in $C(M)$:

- $\alpha = 0$. Trivial.
- $\alpha = \mathrm{succ}(\beta)$. If

$$M \in (\bar{\Theta}_\rho^\#)^{\mathrm{succ}(\beta)} = \bar{\Theta}_\rho^\#((\bar{\Theta}_\rho^\#)^\beta)$$

then $M = \mathrm{intro}_i(N_1, \ldots, N_n)$, where

  * $N_j \in [\![M_{i,j}]\!](\rho[X := (\bar{\Theta}_\rho^\#)^\beta][x_1, \ldots, x_{j-1} := N_1, \ldots, N_{j-1}])$,
  * $\Theta_i \equiv (x_1{:}M_1)\ldots(x_m{:}M_m)X$, and

* $M_{i_1}, \ldots, M_{i_p}$ are the strictly positive operators for Θ_i.

Then for each $1 \leq j \leq p$, let $M_{i_j} \equiv (y_1{:}A_1)\ldots(y_q{:}A_q)X$. We know that if $P_k \in [\![A_{i,j,k}]\!](\rho[x_1,\ldots,y_1,\ldots := N_1,\ldots,P_1,\ldots])$ for $1 \leq k \leq q$ then $N_j(P_1,\ldots,P_q) \in (\bar{\Theta}^{\#}_{\rho})^{\beta}$, so by inductive hypothesis

$$R^{\beta}(N_j(P_1,\ldots,P_q)) \in C(N_j(P_1,\ldots,P_q))$$

Therefore

$$\begin{array}{ll} & [\![\Phi^{\natural}_{i_j}[A,C,f,z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][f := R^{\beta}][z := N_{i_j}]) \\ \in & [\![\Phi^{\circ}_{i_j}[A,C,z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][z := N_{i_j}]) \end{array}$$

so $[\![\mathrm{E}^X[\bar{\Theta}]]\!](\rho)$ is defined.

Finally, we know that

$$\begin{array}{ll} & f_i(N_1,\ldots,N_n, \\ & \quad [\![\Phi^{\natural}_{i_1}[A,C,f,z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][f := R^{\beta}][z := N_{i_j}]),\ldots) \\ \in & [\![C(\iota^X_i[\bar{\Theta}](x_1,\ldots,x_n))]\!](\rho[x_1,\ldots,x_n := N_1,\ldots,N_n]) \\ = & C(\mathrm{intro}_i(N_1,\ldots,N_n)) \end{array}$$

– Limit α. By inductive hypothesis.

- ($\bar{\Theta}$-*Eq*). We need to show that

$$\begin{array}{ll} & [\![\Phi^{\natural}_{i_j}[A,C,f,z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][f := R^{\beta}][z := N_{i_j}]) \\ = & [\![\Phi^{\natural}_{i_j}[A,C,f,z]]\!](\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!](\rho)][C := C][f := R^{\alpha_0}][z := N_{i_j}]) \end{array}$$

for any β such that $N_{i_j} \in [\![\Phi_{i_j}]\!](\rho[X := (\bar{\Theta}^{\#}_{\rho})^{\beta}])$. This follows by transfinite induction on α_0.

- (κ-*Eq*). By inductive hypothesis we know that

$$[\![\Theta_i]\!](\rho[X := S]) = [\![\Theta'_i]\!](\rho[X := S])$$

for any $S \in [\![\mathrm{Type}]\!]$ and $1 \leq i \leq n$, so ${\Theta_i}^{\#}_{\rho} = {\Theta'_i}^{\#}_{\rho}$. Hence $\bar{\Theta}^{\#}_{\rho} = \bar{\Theta}'^{\#}_{\rho}$, so the interpretations of the operators on these schemas are equal.

- (U)(T)(prop)(prf). Straightforward.

- $(\mu^X[\bar{\Theta}])$. Similar to $(\mathcal{M}^X[\bar{\Theta}])$, using $[\![\mathrm{U}]\!]$ in the place of $[\![\mathrm{Type}]\!]$. We know for every $(\Gamma', A) \in \textsc{Types}_\Gamma(\bar{\Theta}')$ that $A \equiv \mathrm{T}(a)$ for some a, so $[\![a]\!](\rho') \in V_{\kappa_0}$ for any $\rho' \in [\![\Gamma']\!]$.

$\square$

Chapter 6

Soundness

In this chapter we show that UTT^S is sound for UTT: that is,

- if $\Gamma \vdash A$ kind then there is a B such that $\Gamma \vdash^S A \to^{\mathrm{nf}} B$,
- if $\Gamma \vdash M : A$ then there are P and B such that $\Gamma \vdash^S A \to^{\mathrm{nf}} B$ and $\Gamma \vdash^S M \to^{\mathrm{nf}} P : B$,
- if $\Gamma \vdash A = B$ then there is a C such that $\Gamma \vdash^S A \downarrow B\ [C] :$ and
- if $\Gamma \vdash M = N : A$ then there are P and B such that $\Gamma \vdash^S A \to^{\mathrm{nf}} B$ and $\Gamma \vdash^S M \downarrow N\ [P] : B$.

As we have mentioned frequently before, the soundness result is crucial to our treatment of the metatheory of UTT. It is through soundness and completeness for UTT^S that we transfer the metatheoretic development of UTT^S to UTT. Of particular interest is that we use this new system to show strengthening and subject reduction for UTT.

The technique we use to show soundness is closely related to existing proofs of normalization. However, by instead showing soundness for the system UTT^S, we are able to make more explicit the exact conditions needed in the proof: the rules of inference of UTT^S show that the well-typed terms are exactly those constructed with the canonical constructors of the kinds and a series of weak head expansions. Furthermore, as we mentioned in Section 4.10, the reasoning used to show strong normalization is applicable to other important results about reduction as well, and the rules of inference for UTT^S provide a general structure for showing such results.

6.1 Normalization Proofs

We have already discussed normalization proofs in Section 2.4.1. In this section we shall concentrate on proofs of normalization for type theories with dependent types.

We mentioned in Section 2.4.1 that the standard proof of normalization for the simply typed lambda calculus relies on an infinite number of variables of each type. For simple types, where there is no dependency of types upon terms, it is easy to see how to enumerate all of the types. The extension to systems with dependent types has been developed by Pottinger [65] and Luo [44]. These proofs introduce an infinite context with infinitely many variables of each type and shows that any term typable in the original system is typable in this infinite context. However, we shall again use the technique of a Kripke-style model construction which we used in Chapter 2.

A standard technique for showing normalization for simpler type systems interprets types as sets of terms which need not be well-typed. Gallier [28] discusses the proof of normalization for System F using untyped interpretations in detail, following Tait [77] and Mitchell [61]. Altenkirch [4] and Geuvers [30] use a similar technique for the more difficult system of the Calculus of Constructions, which has dependent types.

However, in our proof we are interested in soundness for the formal system UTT^S instead of simple normalization, so by definition we need to construct a model with well-typed terms. In our view, the improvements this approach leads to in the study of the metatheory, as developed in Chapter 4, justify the added complexity of a full soundness proof even for simpler systems like the Calculus of Constructions, because our typed operational semantics is so well suited to studying the relationship between reduction and well-typed terms. We have already discussed this in detail in Section 4.9.

There are several proofs of normalization that do not use infinite contexts and interpret types as sets of well-typed terms. Martin-Löf [53] gives an early such proof, which as we mentioned in Section 2.4.1 uses a restricted reduction relation to avoid the need for the general notion of substitution. This proof also uses a typed reduction relation, similar to our typed operational semantics. Coquand [15] uses a Kripke-style model construction to show normalization of a subset of Martin-Löf type theory using sets of well-typed terms.

It seems that we could also construct the model in this chapter for UTT^S, which would prove the admissibility of application and substitution for UTT^S without reference to UTT. Studying the close relationship between terms and derivations as expressed by Uniqueness of Normal Forms (Lemma 4.9.2) might lead to a better proof method. This topic seems to be an area for further research, and we discuss possible directions briefly in Section 7.2.

Our proof follows standard techniques for proving normalization [52,44,15]. However, we note the following differences:

- We prove the result for UTT, a type theory presented in the Logical Framework which has a predicative universe, a wide class of inductive types, and an impredicative universe of propositions. The impredicativity in particular forces us to use a complicated proof technique.

- Our proof requires strong normalization as a component of the definition of the interpretation of kinds, following Martin-Löf [52] and Koletsos [41]. This means that we need neither Luo's quasi-normalization result [44] nor strengthening in our proof of normalization.

- We give an explicit set-theoretic construction of the complexity measure which we use in the proof.

- Again, we are proving soundness for the system UTT^S instead of normalization. This, in conjunction with the completeness theorem, means that all of the metatheory we have developed for UTT^S will transfer to UTT. In particular, we shall have strengthening, subject reduction and Church–Rosser, in addition to strong normalization, as consequences of the soundness theorem.

6.2 An Overview of the Proof

In this section we shall give a general description of the proof of soundness of UTT^S for UTT. We intend to give a description of the important points in the proof without going into technical details.

Normalization and soundness proofs in general are done by model constructions. Our first concern in a model construction is the semantic domain of discourse. In Section 6.3 we define the semantic objects for our soundness proof.

Because we are proving soundness for UTT^S, the primary component of our semantic objects will be terms with derivations of well-typedness in UTT^S. We shall show that, relative to any substitution, all terms which are well-typed in UTT are in the interpretation of their kinds, where the kinds are interpreted as sets of terms which are well-typed in UTT^S.

It is well known that combining the strong logical principle of impredicativity with other principles can easily lead to inconsistency. Girard showed that Martin-Löf's original type theory, with an axiom of Type: Type, was inconsistent, and he later showed that a system with two levels of impredicativity is inconsistent as well [14]. Hence, an important first step in showing the consistency of a type theory is to give a complexity measure on the types of the theory, since the systems mentioned above fail to have such a measure.

As we mentioned in Chapter 5, we use the set-theoretic model to derive a complexity measure. One intuition for the complexity measure is that we need to be able to give a "natural" set-theoretic interpretation for the non-propositional types. The idea is that one component of the interpretation is essentially the set-theoretic interpretation, but where types are sets of terms. This allows us to model substitution for quantification on types. For example, if we want to interpret the impredicative proposition $\forall(\mathrm{Prop}, [X{:}\mathrm{Prop}]X)$, which quantifies over propositions, we need to be able to consider arbitrary possible interpretations of propositions. The interpretation of the quantification will be the intersection of the interpretation of all these possible interpretations. We shall need this not only for propositions but for any type in the type theory, although not the kinds of the Logical Framework. However, since the interpretation of the impredicative universe will be one of the first defined, we need to know all of the semantic objects before the interpretation is defined.

This problem motivates the definition of value sets. The value sets are intended to capture the possible set-theoretic behaviors of elements of each type: for propositions A the values are sets of terms of type $\mathrm{El}(\mathrm{Prf}(A))$ with special conditions on these sets, for functions the values are functions from values for the domain to values for the range, and so on.

Luo [44] uses what he calls a quasi-normalization result to give such a complexity measure for the type theory *ECC*. He gives this definition of the measure by considering types in a quasi-normal form, where the type constructors for Π, Σ and universes can all be distinguished. He therefore shows that all types can be reduced to another type of this form before the full normalization result.

We have avoided needing such a result in two ways. First, because we do not use Tait's specific non-structural technique for proving normalization, as discussed in Section 2.4.2, we can always know that the type of a term is normalizing by inductive hypothesis in the course of the proof. We therefore have a technique that would not require quasi-normalization for *ECC* but could still use Luo's simple complexity measure by considering the normal forms. However, our complexity measure is defined by considering the complexity of the set-theoretic interpretation, which does not rely on normal forms in any way. We have needed to use this complexity measure not because of difficulty in obtaining normal forms but because there seems to be no simple measure for types in *UTT*, even if we restrict ourselves to the types in normal form. We have discussed this problem in Appendix A.

In order to give a Kripke-style interpretation of the type theory, we need semantic values to be closed under transitions, which for us will be context extensions δ from Δ' to Δ. Because of Renaming (Lemma 4.9.9), we know that terms typed in UTT^S are closed under context extensions. To allow the semantic objects, which include values, to be similarly closed, we index the values by context extensions. Therefore our semantic objects at a context Δ will be pairs of a term well-typed in UTT^S at context Δ and a family of values, indexed by context extensions δ from Δ' to Δ. The formal definition of semantic objects and value sets is given in Definition 6.3.1. We also define the obvious notions of monotonicity $\mathrm{mon}_\delta(.)$ (Definition 6.3.3) and application $\mathrm{APP}_\Delta(.,.)$ (Definition 6.3.4) on these objects.

Coquand and Gallier's proof[17], using Kripke-style models, seems to fail because their notion of substitution is not closed under context extensions. There is no way to pass from a substitution from Δ to Γ to a substitution from Δ' to Γ if δ is a renaming from Δ' to Δ. As we have seen already in the proof of soundness for the simply typed lambda calculus, this extensibility is an essential property of substitution. We also need the property which we call coherence (Lemma 6.5.2), which states the distributivity of monotonicity and the interpretation. We discuss this more in Section 6.4, and the essential problem with the paper is addressed by Lemma 6.4.5. Ritter [68] has also addressed this problem, using a Kripke-like interpretation to show strong normalization for a version of the Calculus of Constructions presented with categorical combinators.

The equality we intend for semantic objects is that two semantic objects are equal if the term components have the same normal form and the indexed value components are equal at all extensions. In the course of our soundness proof, we shall need to show that the application equality rule (*App-Eq*) is sound: in particular, we shall need to know that if (M, v_M) is a semantic object for $(x{:}A_1)A_2$ at Δ and (N_1, v) and (N_2, v) are equal semantic objects for A_1 at Δ then $\mathrm{APP}_\Delta((M, v_M), (N_1, v)) = \mathrm{APP}_\Delta((M, v_M), (N_2, v))$. For this to be the case, the function in the value v_M will need to return values which depend only on the normal form of the term component of the argument. Many proofs of normalization give this requirement as part of the definition of values, but we follow

Coquand [13] in giving this instead as a predicate on semantic objects, uniformity (Definition 6.3.6).

Section 6.4 is a short section which introduces the notion of environments for the semantic objects and gives the expected definitions of equality and uniformity for these environments, lifted from equality and uniformity for semantic objects.

We have now defined the semantic objects we shall use throughout the proof. Section 6.5 follows Streicher's technique [75] by giving a partial interpretation of terms to semantic objects and kinds to sets of semantic objects, relative to an environment of semantic objects, as we have already discussed in Section 5.1. However, in showing that the interpretation is a semantic object if it is defined, we need to show simultaneously the distributivity of monotonicity and the interpretation. We also define the interpretation for contexts as environments whose semantic objects are elements of the interpretations of the constituent kinds (Definition 6.5.3).

Indeed, the similarity of this entire development with Chapter 5 suggests that a better approach might be to give a soundness proof for a more abstract semantics, perhaps to a category-theoretic interpretation of type theory, followed by specific treatments for set theory and UTT^S. Altenkirch [4] uses this technique to show strong normalization of the Calculus of Constructions.

After defining the interpretation, Definition 6.5.1, we show that this interpretation function has desirable properties. Specifically, we prove by induction on the structure of terms lemmas for the interpretation about:

- changing the bound variables (Lemma 6.6.2),
- equality of the interpretation under environments which correspond on the free variables (Lemma 6.6.3),
- uniformity of the interpretation (Lemma 6.6.1), for the soundness of the application equality rule (*App-Eq*), and to allow us to pass from a valuation at Δ to a valuation at Δ' given a renaming from Δ' to Δ, and

- a substitution property for the interpretation (Lemma 6.6.4), used in the soundness of the rules (*S-App*) and (β).

Finally, in Section 6.7 we give the final proof of soundness by induction on derivations in *UTT*. This result is for any valuation in the interpretation of the context, so we prove that if $\vdash\Gamma$ then there is such an valuation from Γ to Γ, which gives the exact soundness result we need.

6.3 Semantic Objects

In this section we give the important definitions for semantic objects. These include the value sets, which are one of the components of the semantic objects, the operations of monotonicity and application on semantic objects, and the property of uniformity for semantic objects.

We shall make use of the complexity measures $c(A)$ and $k(M)$ which we define in Appendix A. We summarize the important properties of these complexity measures:

- If $\textsc{Sch}_{\Gamma;X}(\bar{\Theta})$ and for any $(\Gamma', A) \in \textsc{Types}_\Gamma(\bar{\Theta})$, where $\Gamma' \equiv \Gamma, x_1{:}A_1, \ldots, x_n{:}A_n$, if for all A_i we have that $\Gamma, x_1{:}A_1, \ldots, x_{i-1}{:}A_{i-1} \vdash N_i : A_i$, then
$$c(\Gamma \vdash [\bar{N}/\Gamma']A \text{ kind}) < c(\Gamma \vdash \mathrm{El}(\mathcal{M}^X[\bar{\Theta}]) \text{ kind})$$
- If $\Gamma \vdash (x{:}A_1)A_2$ kind then $c(\Gamma \vdash A_1 \text{ kind}) < c(\Gamma \vdash (x{:}A_1)A_2 \text{ kind})$ and
$$c(\Gamma \vdash [N/x]A_2 \text{ kind}) < c(\Gamma \vdash (x{:}A_1)A_2 \text{ kind})$$
for any N such that $\Gamma \vdash N : A_1$.
-
  - $c(\Gamma \vdash \mathrm{El}(\mathrm{Prop}) \text{ kind}) < c(\Gamma \vdash \mathrm{El}(\mathrm{U}) \text{ kind})$.
  - $c(\Gamma \vdash \mathrm{El}(\mathrm{Prf}(P)) \text{ kind}) < c(\Gamma \vdash \mathrm{El}(\mathrm{U}) \text{ kind})$ for any P such that $\Gamma \vdash^* P : \mathrm{Prop}$.

- $c(\Gamma \vdash \mathrm{El}(\mathcal{M}^X[\bar{\Theta}])\ \mathrm{kind}) < c(\Gamma \vdash \mathrm{El}(\mathrm{U})\ \mathrm{kind})$ if $\Gamma \vdash \mu^X[\bar{\Theta}] : \mathrm{El}(\mathrm{U})$.

- $k(\Gamma \vdash N_j(P_1, \ldots, P_p) : \mathcal{M}^X[\bar{\Theta}]) < k(\Gamma \vdash \iota_i^X[\bar{\Theta}](N_1, \ldots, N_n) : \mathcal{M}^X[\bar{\Theta}])$ for $1 \leq j \leq n$.

Definition 6.3.1 (Value Sets) *We define the* semantic objects *for* Δ *and* A $\mathrm{SO}_\Delta(A)$, *the* value sets $V(\Delta \vdash^S M : A)$, *and the* saturated sets $\mathrm{SAT}_\Delta(A)$, *where we assume that* $\Delta \vdash^S A \rightarrow^{\mathrm{nf}} A$ *or* $\Delta \vdash^S M \rightarrow^{\mathrm{nf}} M : A$, *simultaneously. The following definitions are for arbitrary normal* A*:*

- *A semantic object for* Δ *and* A *is a pair* (M, v), *where* $\Delta \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *and* v *is a family of objects* $v(\delta) \in V(\Delta' \vdash^S P : A)$, *indexed by renamings* δ *from* Δ' *to* Δ.

- $\mathrm{SO}_\Delta(A)$ *is the set of semantic objects for* Δ *and* A.

- S *is a saturated set for* Δ *and* A, *notation* $\mathrm{SAT}_\Delta(A)$, *if* S *is a set of semantic objects for* Δ *and* A *such that the following conditions hold:*

  (S1) *If* M *is a base term such that* $\Delta \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *then* $(M, \lambda\delta.*)$ *is in* S, *and*

  (S2) *if* (N, v) *is in* S *and* $\Delta \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ *then* (M, v) *is in* S.

- *A family* $v(\delta)$ *of sets indexed by renamings* δ *from* Δ' *to* Δ *is a* saturated family of sets *for* Δ *and* A *if* $v(\delta)$ *is a saturated set for* Δ' *and* A *for each renaming* δ *from* Δ' *to* Δ.

The definition of $V(\Delta \vdash^S M : A)$, *where* $\Delta \vdash^S M \rightarrow^{\mathrm{nf}} M : A$, *is given by induction on the complexity of* A*:*

- $V(\Delta \vdash^S M : A) =_{\mathrm{df}} \{*\}$ *if* M *is a base term, for any* A.

- $V(\Delta \vdash^S \Lambda(A, P, g) : \mathrm{El}(\mathrm{Prf}(\forall(A, P)))) =_{\mathrm{df}} \{*\}$.

- $V(\Delta \vdash^S \forall(A, P) : \mathrm{El}(\mathrm{Prop})) =_{\mathrm{df}} \mathrm{SAT}_\Delta(\mathrm{El}(\mathrm{Prf}(\forall(A, P))))$.

- $V(\Delta \vdash^S \iota_i^X[\bar{\Theta}](N_1, \ldots, N_n) : \mathrm{El}(\mathcal{M}^X[\bar{\Theta}])) =_{\mathrm{df}} f_\Delta(\Theta_i, N_1, \ldots, N_n)$, *where* $f_\Delta(\Theta, N_1, \ldots, N_n)$ *is defined by induction on the complexity of* $\mathcal{M}^X[\bar{\Theta}]$ *and* $\mathrm{S{\small CH}}^S_{\Delta;X}(\Theta_i)$*:*

  - $f_\Delta(X, ()) =_{\mathrm{df}} \{*\}$.
  - $f_\Delta((x{:}A)\Theta_0, N_1, \ldots, N_n) \ =_{\mathrm{df}} \ V(\Delta \vdash^S N_1 : A) \times f_\Delta(\Theta_0', N_2, \ldots, N_n)$, *where* $\Delta \vdash^S [N_1/x]\Theta_0 \to^{\mathrm{nf}} \Theta_0'$.
  -
  $$\begin{aligned} f_\Delta((\Phi)\Theta_0, N_1, \ldots, N_n) \ &=_{\mathrm{df}} \ V(\Delta \vdash^S N_1 : \Phi(\mathcal{M}^X[\bar{\Theta}])) \times \\ &\qquad f_\Delta(\Theta_0, N_2, \ldots, N_n) \end{aligned}$$

- $V(\Delta \vdash^S \mathrm{prop} : \mathrm{El}(\mathrm{U})) =_{\mathrm{df}} \mathrm{SAT}_\Delta(\mathrm{El}(\mathrm{Prop}))$.

- $V(\Delta \vdash^S \mathrm{prf}(\forall(A, P)) : \mathrm{El}(\mathrm{U})) =_{\mathrm{df}} \mathrm{SAT}_\Delta(\mathrm{El}(\mathrm{Prf}(\forall(A, P))))$.

- $V(\Delta \vdash^S \mu^X[\bar{\Theta}] : \mathrm{El}(\mathrm{U})) =_{\mathrm{df}} \mathrm{SAT}_\Delta(\mathrm{El}(\mathcal{M}^X[\bar{\Theta}]))$.

- $V(\Delta \vdash^S M : \mathrm{Type}) =_{\mathrm{df}} \mathrm{SAT}_\Delta(\mathrm{El}(M))$ *if* M *is not a base term.*

- $V(\Delta \vdash^S M : (x{:}A_1)A_2)$, *if* M *is not a base term, is the set of* f *such that:*

  - $dom(f) \in \mathrm{SAT}_\Delta(A_1)$ *and*
  - $f(N, v_N) \in V(\Delta \vdash^S P : B_2)$, *where* $(N, v_N) \in dom(f)$, $\Delta \vdash^S M(N) \to^{\mathrm{nf}} P : B_2$ *and* $\Delta \vdash^S [N/x]A_2 \to^{\mathrm{nf}} B_2$.

There are several important points about this definition:

- The induction is well defined by Completeness (Lemma 4.9.5) and the complexity measure in Appendix A.

- By Strong Normalization, semantic objects are always strongly normalizing. Also, if $\Delta \vdash^S M \to^{\mathrm{nf}} P : A$ then $\Delta \vdash^S P \to^{\mathrm{nf}} P : A$ by Adequacy for Reduction and Subject Reduction.

- The cases of the definition are exhaustive by Generation (Lemma 4.9.1) and unique by Uniqueness of Normal Forms (Lemma 4.9.2).

- We shall use the usual symbol for equality on semantic objects to denote the least equivalence relation which includes α-equivalence on the first component.

- In many proofs of normalization for impredicative calculi in the literature [44,17] there is a lemma stating that for any context Δ, term M and kind A there is a canonical value $v_{\Delta,M,A} \in V(\Delta \vdash^S M : A)$. However, this lemma is only used for base terms, to allow the context to be extended under binders. In our formulation it is obvious that if $\Delta \vdash^S M \to^{\mathrm{nf}} P : A$ and P is a base term then $* \in \{*\} = V(\Delta \vdash^S P : A)$.

- The domain of a function in the value set for dependent product kinds $(x{:}A_1)A_2$ need not be the whole of $\mathrm{SO}_\Delta(A_1)$. This is because when we define the interpretation of $[x{:}A_1]M_0$, for example, the domain of the function defined will need to be the interpretation of A_1, since the inductive hypothesis will only apply for elements of this set. This is the reason that base terms always have the same value set regardless of their kind, because we do not know the interpretation of A_1 beforehand but we need to know that base terms are always in saturated sets.

- Because of Uniqueness of Normal Forms, we shall not distinguish between terms M well-typed in context Δ in UTT^S and proofs of $\Delta \vdash^S M \to^{\mathrm{nf}} P : A$.

Lemma 6.3.2 *For any kind A such that $\Delta \vdash^S A \to^{\mathrm{nf}} B$, we have $\mathrm{SO}_\Delta(B) \in \mathrm{SAT}_\Delta(B)$.*

Proof Straightforward. $\square$

Definition 6.3.3 (Monotonicity) *If (M, v_M) is a semantic object for Δ and A and δ is a renaming from Δ' to Δ then*

$$\mathrm{mon}_\delta(M, v_M) =_{\mathrm{df}} (\widehat{\delta}(M), \lambda\delta'.v_M(\delta' \circ \delta))$$

By construction we know that $\mathrm{mon}_\delta(M, v_M)$ is a semantic object for Δ' and A.

Definition 6.3.4 (Application) *Let (M_1, v_1) be a semantic object for Δ and $(x{:}A_1)A_2$ such that $\Delta \vdash^S M_1 \rightarrow^{\mathrm{nf}} P_1 \;:\; (x{:}A_1)A_2$, and (M_2, v_2) be a semantic object for Δ and A_1 such that $\Delta \vdash^S M_2 \rightarrow^{\mathrm{nf}} P_2 \;:\; A_1$. We show that there is a B_2 such that $\Delta \vdash^S [M_2/x]A_2 \rightarrow^{\mathrm{nf}} B_2$ and define*

$$\mathrm{APP}_\Delta((M_1, v_1), (M_2, v_2)) \in \mathrm{SO}_\Delta(B_2)$$

the partial function for semantic application:

- *If P_1 is a base term and $\Delta \vdash^S [M_2/x]A_2 \rightarrow^{\mathrm{nf}} B_2$ then*

  $$\mathrm{APP}_\Delta((M_1, v_1), (M_2, v_2)) =_{\mathrm{df}} (M_1(M_2), *)$$

  where $\Delta \vdash^S M_1(M_2) \rightarrow^{\mathrm{nf}} P_1(P_2) \;:\; B_2$ by (S-App).

- *If P_1 is not a base term and $\mathrm{mon}_\delta(M_2, v_2) \in dom(v_1(\delta))$ for all renamings δ from Δ' to Δ then*

  $$\mathrm{APP}_\Delta((M_1, v_1), (M_2, v_2)) =_{\mathrm{df}} (M_1(M_2), \lambda\delta.v_1(\delta)(\mathrm{mon}_\delta(M_2, v_2)))$$

  where $\Delta \vdash^S [M_2/x]A_2 \rightarrow^{\mathrm{nf}} B_2$ and $\Delta \vdash^S M_1(M_2) \rightarrow^{\mathrm{nf}} P \;:\; B_2$ by definition of $v_2(\mathrm{id}_\Delta) \in V(\Delta \vdash^S M_1 \;:\; (x{:}A_1)A_2)$.

Lemma 6.3.5

$$\begin{aligned} & \mathrm{APP}_{\Delta''}(\mathrm{mon}_{\delta'\circ\delta}(M, v_M), \mathrm{mon}_{\delta'}(N, v_N)) \\ = \;& \mathrm{mon}_{\delta'}(\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta(M, v_M), (N, v_N))) \end{aligned}$$

Proof Straightforward by the definitions of application and monotonicity, using Thinning (Lemma 4.9.11). □

Notation We write $(M, v_M) \downarrow (N, v_N)$ if (M, v_M) and (N, v_N) are semantic objects for Δ and A, there is a P such that $\Delta \vdash^S M \downarrow N \; [P] \;:\; A$, and $v_M = v_N$.

Definition 6.3.6 (Uniform Semantic Objects) *Let S be a saturated family of sets for Δ and A. Then S is* uniform *if $(M, v_M) \in S(\delta)$ implies that (M, v_M) is uniform and $\mathrm{mon}_{\delta'}(M, v_M) \in S(\delta' \circ \delta)$ for all renamings δ' from Δ'' to Δ'.*

A semantic object (M, v_M) for Δ and A is uniform *if:*

- *M is a base term,*
- *$M \equiv \forall(A, P)$ and v_M is uniform,*
- *$M \equiv \Lambda(A, P, g)$,*
- *$M \equiv \iota_i^X[\bar{\Theta}](N_1, \ldots, N_n)$ and (N_i, v_i) is uniform for $1 \leq i \leq n$,*
- *$M \equiv \mathrm{prop}$ and v_M is uniform,*
- *$M \equiv \mathrm{prf}(\forall(A, P))$ and v_M is uniform,*
- *$M \equiv \mu^X[\bar{\Theta}]$ and v_M is uniform,*
- *$A \equiv \mathrm{Type}$, M is not a base term and v_M is uniform, or*
- *$\Delta \vdash^S M \rightarrow^{\mathrm{nf}} P : (x{:}A_1)A_2$ and*

  (U1) *$\lambda\delta.dom(v_M(\delta))$ is uniform,*

  (U2) *if $(N, v_N) \in dom(v(\delta))$ then $\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta(M, v_M), (N, v_N))$ is uniform, and*

  (U3) *if $(N_1, v_1) \downarrow (N_2, v_2)$ then*

$$\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta(M, v_M), (N_1, v_1)) \downarrow \mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta(M, v_M), (N_2, v_2))$$

We have adopted Coquand's technique [13] of separating this condition from the definition of the value sets. This allows us to simplify the definition of value sets and to prove that the interpretation is uniform (Lemma 6.6.1) separate from proving that the interpretation is well-defined (Lemma 6.5.2).

The definition of uniformity is not as basic as it could be. We could instead have separated it into a property of monotonicity and a property about semantic objects for $(x{:}A_1)A_2$ being effectively functions on equivalence classes of terms quotiented by having a common normal form (property *(U3)*).

Definition 6.3.7 (Uniform Semantic Objects) *The set of* uniform semantic objects *for* Δ *and* A*, notation* $\mathrm{USO}_\Delta(A)$*, is the set of semantic objects for* Δ *and* A *which are uniform.*

Lemma 6.3.8 *We list several simple properties of uniform semantic objects:*

- $\mathrm{USO}_\Delta(A)$ *is uniform and in* $\mathrm{SAT}_\Delta(A)$*.*
- *If* (M, v_M) *is a uniform semantic object then* $\mathrm{mon}_\delta(M, v_M)$ *is also a uniform semantic object, for any renaming* δ *from* Δ' *to* Δ*.*

6.4 Valuations

We recall from Section 4.9 that a pre-substitution is an environment for terms.

Definition 6.4.1 (Substitution) *A* substitution ϕ *from* Δ *to* Γ*, where* $\Gamma \equiv x_1{:}A_1, \ldots, x_m{:}A_m$ *and* $\vdash^S \Delta$*, is a pre-substitution over* Γ *such that*

$$\Delta \vdash^S \widehat{\phi}(A_i) \rightarrow^{\mathrm{nf}} B_i \quad \textit{and} \quad \Delta \vdash^S \phi(x_i) \rightarrow^{\mathrm{nf}} P \,:\, B_i \qquad \textit{for } 1 \leq i \leq m.$$

A pre-valuation is an environment for semantic objects at some context Δ.

Definition 6.4.2 (Valuation) *A* valuation *from* Δ *to* Γ *is a pre-valuation such that the term component is a substitution from* Δ *to* Γ*.*

Definition 6.4.3 (Confluent Valuations) *Let* ρ_1 *and* ρ_2 *be valuations from* Δ *to* Γ*. Then if there is a* P *such that* $\Delta \vdash^S \phi_1(x) \downarrow \phi_2(x)\ [P] \,:\, \widehat{\phi_1}(A)$ *and* $\mathrm{val}_1(x) = \mathrm{val}_2(x)$ *for all* $(x{:}A) \in \Gamma$*, we say that* ρ_1 *and* ρ_2 *are* confluent valuations *for* Γ *at* Δ*.*

Definition 6.4.4 (Uniform Valuation) *A valuation ρ from Δ to Γ is* uniform *if $\rho(x)$ is uniform for all $(x{:}A) \in \Gamma$.*

Lemma 6.4.5 *If ρ is a valuation from Δ to Γ and δ is a renaming from Δ' to Δ then $\mathrm{mon}_\delta(\rho)$ is a valuation from Δ' to Γ.*

Proof By Thinning and the definitions of monotonicity and valuation. □

6.5 The Interpretation

We give a partial interpretation function from terms to semantic objects and from kinds to sets of semantic objects. For a further discussion of the technique for partial interpretations of terms to define models for type theory we refer the reader to Section 5.1.

As we mentioned in Section 6.2, the value of the interpretation function for a term relative to a valuation from Δ to Γ is a semantic object for Δ, which is a pair of a term well-typed in UTT^S in context Δ and a family of values indexed by renamings from Δ' to Δ. However, we are somewhat informal in our treatment of the interpretation with respect to terms which are well-typed in UTT^S. In the definition of the interpretation we only give the term rather than a derivation of its well-typedness, but we justify this by Lemma 6.5.2, which shows that the term is indeed well-typed in UTT^S and that the interpretation of a term is a semantic object.

Definition 6.5.1 (Interpretation) *We define the interpretation of terms $[\![M]\!]\rho\Delta$ and of kinds $[\![A]\!]\rho\Delta$ by induction on the structure of M and A.*

$[\![M]\!]\rho\Delta$ is always a pair $(\widehat{\phi}(M), V[\![M]\!]\rho\Delta)$, with $V[\![M]\!]\rho\Delta$ a family of values indexed by δ from Δ' to Δ, defined as below:

- $V[\![x]\!]\rho\Delta =_{\mathrm{df}} \mathrm{val}(x)$ *if $x \in dom(\rho)$.*
- $[\![(x{:}A_1)A_2]\!]\rho\Delta$ *is defined if for some $x \notin dom(\rho)$*

- $\Delta \vdash^S \widehat{\phi}(A_1) \to^{\mathrm{nf}} B_1$,
- $\Delta, y{:}\widehat{\phi}(A_1) \vdash^S \phi\widehat{[x := y]}(A_2) \to^{\mathrm{nf}} B_2$ *for any* y *fresh in* Δ
- $[\![A_1]\!]\rho\Delta$ *is defined, uniform and a saturated family of sets for* Δ *and* B_1*, and*
- *for any renaming* δ *from* Δ' *to* Δ *and any* $(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta)$
  - $\Delta' \vdash^S [M_2/x]B_2 \to^{\mathrm{nf}} B_2'$ *and*
  - $[\![A_2]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'$ *is defined and a saturated family of sets for* Δ' *and* B_2'*, where* $\Delta' \vdash^S \phi\widehat{[x := M_2]}(A_2) \to^{\mathrm{nf}} B_2'$.

$[\![(x{:}A_1)A_2]\!]\rho\Delta(\delta)$ *is the set of* (M_1, v_1) *such that*

- (M_1, v_1) *is a uniform semantic object for* Δ' *and* $(x{:}B_1)B_2$ *and*
- *for all renamings* δ' *from* Δ' *to* Δ*:*
  - *if* M_1 *is not a base term then* $dom(v_1(\delta')) = [\![A_1]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''(\mathrm{id}_{\Delta''})$ *and*
  - *for all* $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''(\mathrm{id}_{\Delta''})$,

$$\begin{array}{ll} & \mathrm{APP}_{\Delta''}(\mathrm{mon}_{\delta'}(M_1, v_1), (M_2, v_2)) \\ \in & [\![A_2]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)[x := (M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''}) \end{array}$$

- $[\![[x{:}A_1]M_0]\!]\rho\Delta$ *is defined if for some* $x \notin dom(\rho)$
  - $\Delta \vdash^S \widehat{\phi}(A_1) \to^{\mathrm{nf}} B_1$,
  - $\Delta, y{:}\widehat{\phi}(A_1) \vdash^S \phi\widehat{[x := y]}(M_0) \to^{\mathrm{nf}} [y/x]P_0 : B_2$ *for any* y *fresh in* Δ,
  - $[\![A_1]\!]\rho\Delta$ *is defined, uniform and a saturated family of sets for* Δ *and* B_1 *and*
  - $[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'$ *is defined and in* $\mathrm{SO}_{\Delta'}(B_2')$*, for any renaming* δ *from* Δ' *to* Δ *and any* $(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta)$*, where* $\Delta' \vdash^S [M_2/x]B_2 \to^{\mathrm{nf}} B_2'$.

$V[\![[x{:}A_1]M_0]\!]\rho\Delta(\delta) =_{\mathrm{df}} *$ *if* $P_0 \equiv P(x)$*, where* $x \notin \mathrm{FV}(P)$ *and* P *is a base term.*

Otherwise,

$$\begin{aligned} & V[\![[x{:}A_1]M_0]\!]\rho\Delta(\delta) =_{\mathrm{df}} \lambda(M_2, v_2) \\ \in \;\; & [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}).V[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

- $[\![M_1(M_2)]\!]\rho\Delta$ *is defined if:*
  - $[\![M_1]\!]\rho\Delta$ *is defined and in* $\mathrm{USO}_\Delta((x{:}B_1)B_2)$, *where* $\Delta \vdash^S \widehat{\phi}(M_1) \to^{\mathrm{nf}} P_1 : (x{:}B_1)B_2$,
  - $[\![M_2]\!]\rho\Delta$ *is defined and in* $\mathrm{USO}_\Delta(B_1)$,
  - *if* P_1 *is a base term then* $\Delta \vdash^S [\widehat{\phi}(M_2)/x]A_2 \to^{\mathrm{nf}} B_2'$ *and*
  - *if* P_1 *is not a base term then* $[\![M_2]\!]\rho\Delta \in dom(V[\![M_1]\!]\rho\Delta(\mathrm{id}_\Delta))$.

  $[\![M_1(M_2)]\!]\rho\Delta =_{\mathrm{df}} \mathrm{APP}_\Delta([\![M_1]\!]\rho\Delta, [\![M_2]\!]\rho\Delta)$

- $[\![\mathrm{Type}]\!]\rho\Delta(\delta) =_{\mathrm{df}} \mathrm{USO}_{\Delta'}(\mathrm{Type})$.

- $[\![\mathrm{El}(M_0)]\!]\rho\Delta$ *is defined if* $[\![M_0]\!]\mathrm{mon}_\delta(\rho)\Delta'$ *is defined and in* $\mathrm{USO}_{\Delta'}(\mathrm{Type})$ *for any renaming* δ *from* Δ' *to* Δ, *where* $\Delta \vdash^S \widehat{\phi}(M_0) \to^{\mathrm{nf}} P_0 : \mathrm{Type}$.

$$[\![\mathrm{El}(M_0)]\!]\rho\Delta(\delta) \;\; =_{\mathrm{df}} \;\; \begin{cases} V[\![M_0]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) & P_0 \textit{ not a base term} \\ \mathrm{USO}_{\Delta'}(\mathrm{El}(P_0)) & P_0 \textit{ a base term} \end{cases}$$

- $V[\![\mathrm{Prop}]\!]\rho\Delta(\delta) =_{\mathrm{df}} \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop}))$.

- $V[\![\mathrm{Prf}]\!]\rho\Delta(\delta) =_{\mathrm{df}} \lambda(P, v_P) \in [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}).v_P$.

- $V[\![\forall]\!]\rho\Delta(\delta)$ *is the function which given*
  - $(A, v_A) \in [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$ *and*
  - $(P, v_P) \in [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)]\Delta'(\mathrm{id}_{\Delta'})$

  gives the saturated set S *such that if*

  $(g, v_g) \in [\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'(\mathrm{id}_{\Delta'})$

  where $\Delta' \vdash^S A \downarrow A'\ [A''] : \mathrm{Type}$ *and* $\Delta' \vdash^S P \downarrow P'\ [P''] : (\mathrm{El}(A''))\mathrm{El}(\mathrm{Prop})$, *then* $(\Lambda(A', P', g), \lambda\delta.*) \in S$.

- $V[\![\Lambda]\!]\rho\Delta(\delta)$ *is the function which given arguments in the appropriate domains returns* $*$.

- $V[\![\mathrm{E}_\forall]\!]\rho\Delta(\delta)$ *is the function which given arguments in the appropriate domains returns* $*$.

- $[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta$ *is defined if* $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_i) \to^{\mathrm{nf}} \Theta'_i$ *for some* $X \notin dom(\rho)$ *and any* Y *fresh in* Δ, *and* $[\![\Theta_i]\!]\rho'\Delta'$ *is defined for any valuation* ρ' *from* Δ' *to* $\Gamma, X{:}\mathrm{Type}$.

  $V[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta(\delta)$ *is defined by transfinite induction on the ordinal* α_0 *to be the least saturated set* S *such that if*

  - $(M_j, v_j) \in [\![M_j]\!]\mathrm{mon}_\delta(\rho)[X := S][x_1 := (M_1, v_1)]\ldots\Delta'(\mathrm{id}_{\Delta'})$ *and*
  - $\Delta', Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \downarrow \Theta'_j\ [\Theta''_j]$ *for any* Y *fresh in* Δ'

  for $1 \leq j \leq n$ *then*

  $$(\iota_i^X[\bar{\Theta}'](N_1, \ldots, N_n), \lambda\delta'.{<}v_1(\delta'), \ldots, v_n(\delta'){>}) \in S$$

- $[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta$ *is defined if* $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_i) \to^{\mathrm{nf}} \Theta'_i$ *for some* $X \notin dom(\rho)$ *and for any* Y *fresh in* Δ, *and* $[\![\Theta_i]\!]\rho'\Delta'$ *is defined for any valuation* ρ' *from* Δ' *to* $\Gamma, X{:}\mathrm{Type}$.

  $V[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta(\delta) =_{\mathrm{df}}$

  $\lambda(M_1, v_1) \in [\![M_1]\!]\mathrm{mon}_\delta(\rho[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])\Delta'(\mathrm{id}_{\Delta'}).$

  $\ldots$

  $\lambda(M_n, v_n) \in [\![M_n]\!]\mathrm{mon}_\delta(\rho[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])[x_1 := (M_1, v_1)]\ldots\Delta'(\mathrm{id}_{\Delta'}).$

  ${<}v_1(\mathrm{id}_{\Delta'}), \ldots, v_n(\mathrm{id}_{\Delta'}){>}$

- $[\![\mathrm{E}^X[\bar{\Theta}]]\!]\rho\Delta$ *is defined if* $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_i) \to^{\mathrm{nf}} \Theta'_i$, *for some* $X \notin dom(\rho)$ *and for any* Y *fresh in* Δ, *and* $[\![\Theta_i]\!]\rho'\Delta'$ *is defined for any valuation* ρ' *from* Δ' *to* $\Gamma, X{:}\mathrm{Type}$.

Assume

$$\begin{array}{rcl}(C, v_C) & \in & [\![(X)\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\mathrm{mon}_\delta(\rho)\Delta']\Delta'(\mathrm{id}_{\Delta'}). \\ (f_1, v_{f_1}) & \in & [\![\Theta_1^\circ[A, C, z]]\!]\rho_1'\Delta'(\mathrm{id}_{\Delta'}) \\ & \ldots & \\ (f_n, v_{f_n}) & \in & [\![\Theta_n^\circ[A, C, z]]\!]\rho_n'\Delta'(\mathrm{id}_{\Delta'})\end{array}$$

where

$\rho_i' =_{\mathrm{df}} \mathrm{mon}_\delta(\rho)[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\mathrm{mon}_\delta(\rho)\Delta'][C := (C, v_C)][z := [\![\iota_i^X[\bar{\Theta}]]\!]\mathrm{mon}_\delta(\rho)\Delta']$

for $1 \leq i \leq n$.

Then $V[\![\mathrm{E}^X[\bar{\Theta}]]\!]\rho\Delta(\delta)((C, v_C), (f_1, v_{f_1}), \ldots, (f_n, v_{f_n})) =_{\mathrm{df}} R^\alpha$ *is the partial function defined by transfinite induction on the complexity of* $\mathcal{M}^X[\bar{\Theta}]$*:*

- $\alpha = 0$. *Then*

$$R^0 =_{\mathrm{df}} \lambda(M, v_M) \in [\![\mathrm{El}(\mathcal{M}^X[\bar{\Theta}])]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}).$$
$$\begin{cases} * & \text{if } M \text{ is a base term} \\ R^0(N, v_M) & \text{if } \Delta' \vdash^S M \to^{\mathrm{wh}} N : \mathrm{El}(\mathcal{M}^X[\bar{\Theta}]) \end{cases}$$

- $\alpha = \mathrm{succ}(\beta)$. *Let* $(M, v_M) \in [\![\mathrm{El}(\mathcal{M}^X[\bar{\Theta}])]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$. *Then*
  - $R^{\mathrm{succ}(\beta)}(M, v_M) =_{\mathrm{df}} *$ *if* M *is a base term,*
  - $R^{\mathrm{succ}(\beta)}(M, v_M) =_{\mathrm{df}} R^{\mathrm{succ}(\beta)}(N, v_M)$ *if* $\Delta' \vdash^S M \to^{\mathrm{wh}} N : \mathrm{El}(\mathcal{M}^X[\bar{\Theta}])$ *and* $R^{\mathrm{succ}(\beta)}(N, v_M)$ *is defined, and*
  - *let* $\Phi_{i_1}, \ldots, \Phi_{i_j}$ *be the strictly positive operators in* Θ *and define* $\Phi_{i_k}^*$ *to be the interpretation of* $\Phi_{i_k}^\natural[A, C, f, z]$ *under the valuation* $\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta][C := (C, v_C)][f := R^\beta][z := (N_{i_k}, v_{i_k})]$, *for* $1 \leq k \leq j$. *Then*

$$R^{\mathrm{succ}(\beta)}(M, v_M) =_{\mathrm{df}} v_{f_i}(N_1, \ldots, N_n, \Phi_{i_1}^*, \ldots, \Phi_{i_j}^*)$$

  if $M \equiv \iota_i^X[\bar{\Theta}'](N_1, \ldots, N_n)$ *and*

$$\begin{array}{l}\Phi_{i_k}^* \\ \in [\![\Phi^\circ[A, C, z]]\!]\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta][C := (C, v_C)][z := (N_{i_k}, v_{i_k})]\Delta\end{array}$$

  for $1 \leq k \leq j$.

- *Limit* α. *Then* $R^{\alpha}(M, v_M) =_{\mathrm{df}} R^{\beta}(M, v_M)$, *where* $\beta =_{\mathrm{df}} \bigcap\{\gamma \in \alpha \mid M \in [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta^{\gamma}(\delta)\}$.

- $V[\![\mathrm{U}]\!]\rho\Delta(\delta)$ *is the saturated set* S *such that if*

  - (M, v_M) *is a uniform semantic object for* Δ' *and* $\mathrm{El}(\mathrm{U})$ *and*
  - $M \equiv \mathrm{prop}$, $M \equiv \mathrm{prf}(P)$ *or* $M \equiv \mu^X[\bar{\Theta}]$

  then $(M, v_M) \in S$.

- $V[\![\mathrm{prop}]\!]\rho\Delta(\delta) =_{\mathrm{df}} \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop}))$.

- $V[\![\mathrm{prf}]\!]\rho\Delta(\delta) =_{\mathrm{df}} \lambda(P, v_P) \in [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_{\delta}(\rho)\Delta'(\mathrm{id}_{\Delta'}).v_P(\mathrm{id}_{\Delta'})$.

- $[\![\mu^X[\bar{\Theta}]]\!]\rho\Delta$ *is defined if there are* $\bar{\Theta}'$ *such that for each* $(\Gamma', A) \in \textsc{Types}_{\Gamma}(\bar{\Theta}')$ *there is an* a *such that* $A \equiv T(a)$, $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_i) \downarrow \Theta_i'\ [\Theta_i'']$ *for* $1 \leq i \leq n$, *and* $[\![\Theta_i]\!]\rho'\Delta'$ *is defined for any valuation* ρ' *from* Δ' *to* $\Gamma, X{:}\mathrm{Type}$.

  In this case $V[\![\mu^X[\bar{\Theta}]]\!]\rho\Delta$ *is the same family of sets defined for* $V[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta$.

- $V[\![\mathrm{T}]\!]\rho\Delta(\delta) =_{\mathrm{df}} \lambda(M, v_M) \in [\![\mathrm{El}(\mathrm{U})]\!]\mathrm{mon}_{\delta}(\rho)\Delta'(\mathrm{id}_{\Delta'}).v_M(\mathrm{id}_{\Delta'})$.

We rely more heavily in this chapter on the alternative induction principle for terms introduced in Section 4.3 than we did in Chapter 5. This is because in the set-theoretic model we always know that $\Pi v \in A.B(x)$ is defined if A and $B(x)$ are defined, whereas in the interpretation here the interpretation of Π is sometimes undefined.

We have chosen our definition of the interpretation of U for convenience. The definition is simple because the computation rule for the universe is simple, only changing the outermost constructor prop, prf or $\mu^X[\bar{\Theta}]$ to its type-level equivalent Prop, Prf or $\mathcal{M}^X[\bar{\Theta}]$. We could instead define the value of the interpretation to be $\mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{U}))$, which would require us to reason in UTT^S to show that T is well-defined. If we had an elimination rule for the universe then we would need to define the interpretation by induction on the complexity of types in the universe, similar to the interpretation of inductive types.

Lemma 6.5.2 (Well-Definedness of the Interpretation) *Suppose ρ is a valuation from Δ to Γ. Then:*

Well-Definedness

- *If $[\![M]\!]\rho\Delta$ is defined then there is a B such that $[\![M]\!]\rho\Delta \in \mathrm{SO}_\Delta(B)$.*
- *If $[\![A]\!]\rho\Delta$ is defined then $\Delta \vdash^S A \rightarrow^{\mathrm{nf}} B$ and $[\![A]\!]\rho\Delta \in \mathrm{SAT}_\Delta(B)$.*

Coherence *If δ is a renaming from Δ' to Δ and then*

- *if $[\![A]\!]\rho\Delta$ is defined then $\mathrm{mon}_\delta([\![A]\!]\rho\Delta) = [\![A]\!]\mathrm{mon}_\delta(\rho)\Delta'$.*
- *if $[\![M]\!]\rho\Delta$ is defined then $\mathrm{mon}_\delta([\![M]\!]\rho\Delta) = [\![M]\!]\mathrm{mon}_\delta(\rho)\Delta'$.*

Proof By simultaneous induction on the structure of M and A.

For Coherence, the terms are identical because

$$\widehat{\delta \circ \phi}(M) \equiv \widehat{\delta}(\widehat{\phi}(M))$$

The second component of the interpretation is always a family of values indexed by a renaming δ from Δ' to Δ, and the interpretation of each subterm is used relative to δ. Hence we only need

$$\mathrm{mon}_{\delta' \circ \delta}(\rho) = \mathrm{mon}_{\delta'}(\mathrm{mon}_\delta(\rho))$$

which is trivial.

Well-Definedness is as follows:

- $M \equiv x$. By definition $\Delta \vdash^S \widehat{\phi}(A) \rightarrow^{\mathrm{nf}} B$ and $\rho(x)$ is a semantic object for Δ and B.

- $A \equiv (x{:}A_1)A_2$. Suppose the conditions we require for $[\![(x{:}A_1)A_2]\!]\rho\Delta$ to be defined are met. This means that $\Delta \vdash^S \widehat{\phi}(A_1) \rightarrow^{\mathrm{nf}} B_1$ and $\Delta, y{:}\widehat{\phi}(A_1) \vdash^S \widehat{\phi[x := y]}(A_2) \rightarrow^{\mathrm{nf}} B_2$, so $\Delta \vdash^S \widehat{\phi}((x{:}A_1)A_2) \rightarrow^{\mathrm{nf}} (y{:}B_1)B_2$ by (S-Π).

  To show $[\![(x{:}A_1)A_2]\!]\rho\Delta$ defined, we need to show that the application used in the definition is well-defined. Suppose δ is a renaming from Δ' to Δ and

(M_1, v_1) is a uniform semantic object for Δ' and $(y{:}B_1)B_2$, where $\Delta' \vdash^S M_1 \rightarrow^{\mathrm{nf}} P_1 : (y{:}B_1)B_2$. If P_1 is a base term then by assumption $\Delta'' \vdash^S [M_2/y]B_2 \rightarrow^{\mathrm{nf}} B_2'$ for any $(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta' \circ \delta)$. If P_1 is not a base term then $dom(v_1(\delta'')) = [\![A_1]\!]\rho\Delta(\delta'' \circ (\delta' \circ \delta))$ for any renaming δ'' from Δ''' to Δ'', and if $(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta' \circ \delta)$ then by the definition of uniformity $\mathrm{mon}_{\delta''}(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta'' \circ (\delta' \circ \delta))$.

To show $[\![(x{:}A_1)A_2]\!]\rho\Delta(\delta) \in \mathrm{SAT}_\Delta((x{:}B_1)B_2)$ we need to show properties *(S1)* and *(S2)* for $[\![(x{:}A_1)A_2]\!]\rho\Delta(\delta)$:

(S1) Suppose M_1 is a base term such that $\Delta' \vdash^S M_1 \rightarrow^{\mathrm{nf}} P_1 : \widehat{\delta}((x{:}B_1)B_2)$ and δ' is a renaming from Δ'' to Δ'. Then if $(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta' \circ \delta)$ then $\Delta'' \vdash^S M_2 \rightarrow^{\mathrm{nf}} P_2 : B_1$, so $M_1(M_2)$ is a base term such that $\Delta'' \vdash^S M_1(M_2) \rightarrow^{\mathrm{nf}} P_1(P_2) : \phi[\widehat{x := M_2}](A_2)$ by Renaming and (*S-App*). Hence by inductive hypothesis and *(S1)* we know that

$$\begin{aligned} & \mathrm{APP}_{\Delta''}(\mathrm{mon}_{\delta'}(M_1, *), (M_2, v_2)) \\ \in \;& [\![A_2]\!]\mathrm{mon}_{\delta' \circ \delta}(\rho)[x := (M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''}) \end{aligned}$$

(S2) Suppose

- $\Delta' \vdash^S M_1 \rightarrow^{\mathrm{wh}} N_1 : (x{:}B_1)B_2$,
- $(N_1, v_1) \in [\![(x{:}A_1)A_2]\!]\rho\Delta(\delta)$ and
- δ' is a renaming from Δ'' to Δ'.

Then if $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_{\delta' \circ \delta}(\rho)\Delta''(\mathrm{id}_{\Delta''})$ then $\Delta'' \vdash^S M_2 \rightarrow^{\mathrm{nf}} P_2 : B_1$ and $\Delta'' \vdash^S [M_2/x]B_2 \rightarrow^{\mathrm{nf}} B_2'$, and

$$\begin{aligned} & \mathrm{APP}_{\Delta''}(\mathrm{mon}_{\delta'}(N_1, v_1), (M_2, v_2)) \\ \in \;& [\![A_2]\!]\mathrm{mon}_{\delta' \circ \delta}(\rho)[x := (M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''}) \end{aligned}$$

by definition of $[\![(x{:}A_1)A_2]\!]\rho\Delta$. Hence $\Delta'' \vdash^S M_1(M_2) \rightarrow^{\mathrm{wh}} N_1(M_2) : B_2'$ by Renaming and (*S-App-WH*), so by inductive hypothesis and *(S2)* for $[\![A_2]\!]\mathrm{mon}_{\delta' \circ \delta}(\rho)[x := (M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''})$ we know that

$$\begin{aligned} & \mathrm{APP}_{\Delta''}(\mathrm{mon}_{\delta'}(M_1, v_1), (M_2, v_2)) \\ \in \;& [\![A_2]\!]\mathrm{mon}_{\delta' \circ \delta}(\rho)[x := (M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''}) \end{aligned}$$

- $M \equiv [x{:}A_1]M_0$. If $[\![[x{:}A_1]M_0]\!]\rho\Delta$ is defined then $\Delta \vdash^S \widehat{\phi}(A_1) \to^{\mathrm{nf}} B_1$ and $\Delta, y{:}\widehat{\phi}(A_1) \vdash^S \phi\widehat{[x := y]}(M_0) \to^{\mathrm{nf}} P_0 \,:\, B_2$. If $P_0 \equiv P(y)$ with $y \notin \mathrm{FV}(P)$ then $\Delta \vdash^S \widehat{\phi}([x{:}A_1]M_0) \to^{\mathrm{nf}} P \,:\, (x{:}B_1)B_2$ by (S-η'), and otherwise $\Delta \vdash^S \widehat{\phi}([x{:}A_1]M_0) \to^{\mathrm{nf}} [y{:}B_1]P_0 \,:\, (y{:}B_1)B_2$ by (S-λ). Let P be the normal form in either case. P need not be an abstraction or a base term because of the possibility of an η-reduction.

  Suppose P is a base term. Then $* \in \{*\} = V(\Delta' \vdash^S P \,:\, (x{:}B_1)B_2)$ for any renaming δ from Δ' to Δ.

  Otherwise, suppose δ is a renaming from Δ' to Δ and $(M_2, v_2) \in [\![A_1]\!]\rho\Delta(\delta)$. We need to show that $V[\![[x{:}A_1]M_0]\!]\rho\Delta(\delta) \in V(\Delta' \vdash^S P \,:\, (y{:}B_1)B_2)$:

  - $dom(V[\![[x{:}A_1]M_0]\!]\rho\Delta(\delta)) = [\![A_1]\!]\rho\Delta(\delta) \in \mathrm{SAT}_{\Delta'}(B_1)$ by inductive hypothesis (Coherence) and assumption.
  - We know that $\Delta' \vdash^S [M_2/x]B_2 \to^{\mathrm{nf}} B_2'$ and

    $$[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta' \in \mathrm{SO}_{\Delta'}(B_2')$$

    so $\Delta' \vdash^S \phi\widehat{[x := M_2]}(M_0) \to^{\mathrm{nf}} P' \,:\, B_2'$ and

    $$V[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\mathrm{id}_{\Delta'}) \in V(\Delta' \vdash^S P' \,:\, B_2')$$

    We also have $\Delta' \vdash^S \widehat{\phi}([x{:}A_1]M_0)(M_2) \to^{\mathrm{nf}} P' \,:\, B_2'$ by Renaming, (S-β) and (S-WH), and $\Delta' \vdash^S [M_2/x]B_2 \to^{\mathrm{nf}} B_2'$ by assumption.

- $M \equiv M_1(M_2)$. Suppose $[\![M_1(M_2)]\!]\rho\Delta$ is defined. Then we have exactly the conditions required for $\mathrm{APP}_\Delta([\![M_1]\!]\rho\Delta, [\![M_2]\!]\rho\Delta)$ to be defined, in particular because $[\![M_1]\!]\rho\Delta$ is uniform.

- $A \equiv \mathrm{Type}$. By (S-Type) and Lemma 6.3.2.

- $A \equiv \mathrm{El}(M_0)$. By definition $[\![M_0]\!]\rho\Delta \in \mathrm{USO}_\Delta(\mathrm{Type})$ and $\Delta \vdash^S M_0 \to^{\mathrm{nf}} P_0 :$ Type, so $\Delta \vdash^S \mathrm{El}(M_0) \to^{\mathrm{nf}} \mathrm{El}(P_0)$ by (S-El). If P_0 is not a base term then $V[\![M_0]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \in V(\Delta' \vdash^S P_0 \,:\, \mathrm{Type}) = \mathrm{SAT}_{\Delta'}(\mathrm{El}(P_0))$ by inductive hypothesis (Coherence), and if P_0 is a base term then $\mathrm{USO}_{\Delta'}(\mathrm{El}(P_0)) \in \mathrm{SAT}_{\Delta'}(\mathrm{El}(P_0))$ by Lemma 6.3.8.

- $M \equiv \mathrm{Prop}$. We know that $\Delta \vdash^S \mathrm{Prop} \to^{\mathrm{nf}} \mathrm{Prop} : \mathrm{El}(\mathrm{U})$ by (S-C).

  Suppose δ is a renaming from Δ' to Δ. Then by Lemma 6.3.8

$$\begin{array}{rl} & V[\![\mathrm{Prop}]\!]\rho\Delta(\delta) = \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop})) \\ \in & \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prop})) = V(\Delta' \vdash^S \mathrm{Prop} : \mathrm{Type}) \end{array}$$

- $M \equiv \mathrm{Prf}$. We know that $\Delta \vdash^S \mathrm{Prf} \to^{\mathrm{nf}} \mathrm{Prf} : (\mathrm{El}(\mathrm{Prop}))\mathrm{Type}$ by (S-C).

  Suppose δ is a renaming from Δ' to Δ. Then:

  - $dom(V[\![\mathrm{Prf}]\!]\rho\Delta(\delta)) = [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \in \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prop}))$ by inductive hypothesis, Well-Definedness and Coherence.
  - Suppose $(P, v_P) \in [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$. Then $\Delta' \vdash^S P \to^{\mathrm{nf}} P' : \mathrm{El}(\mathrm{Prop})$ implies that $\Delta' \vdash^S \mathrm{Prf}(P) \to^{\mathrm{nf}} \mathrm{Prf}(P') : \mathrm{Type}$. If P' is a base term then

$$v_P(\mathrm{id}_{\Delta'}) \in V(\Delta' \vdash^S P' : \mathrm{El}(\mathrm{Prop})) = \{*\} = V(\Delta' \vdash^S \mathrm{Prf}(P') : \mathrm{Type})$$

    and if P' is not a base term then

$$\begin{array}{rl} & v_P(\mathrm{id}_{\Delta'}) \in V(\Delta' \vdash^S P' : \mathrm{El}(\mathrm{Prop})) \\ = & \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prf}(P'))) = V(\Delta' \vdash^S \mathrm{Prf}(P') : \mathrm{Type}) \end{array}$$

- $M \equiv \forall$. We know that $\Delta \vdash^S \forall \to^{\mathrm{nf}} \forall : ((\mathrm{El}(A))\mathrm{El}(\mathrm{Prop}))\mathrm{El}(\mathrm{Prop})$ by (S-C), so suppose δ is a renaming from Δ' to Δ. We need to show that

$$V[\![\forall]\!]\rho\Delta(\delta) \in V(\Delta' \vdash^S \forall : (A{:}\mathrm{Type})((\mathrm{El}(A))\mathrm{El}(\mathrm{Prop}))\mathrm{El}(\mathrm{Prop}))$$

  The domains are saturated sets by inductive hypothesis, and also $\Delta' \vdash^S (\mathrm{El}(A))\mathrm{El}(\mathrm{Prop}) \to^{\mathrm{nf}} B$. Hence, suppose

$$\begin{array}{rcl} (A, v_A) & \in & [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ (P, v_P) & \in & [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)]\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

  We know that $\Delta' \vdash^S A \to^{\mathrm{nf}} A' : \mathrm{Type}$. Then $B \equiv (\mathrm{El}(A'))\mathrm{El}(\mathrm{Prop})$ by Generation and Uniqueness of Normal Forms, so $\Delta' \vdash^S P \to^{\mathrm{nf}} P'$:

$(\mathrm{El}(A'))\mathrm{El}(\mathrm{Prop})$. Hence $\Delta' \vdash^S \forall(A,P) \rightarrow^{\mathrm{nf}} \forall(A',P') : \mathrm{Type}$ follows by (*S-App*).

Finally, we need

$$\begin{aligned} V[\![\forall]\!]\rho\Delta(\delta)((A,v_A),(P,v_P)) &\in V(\Delta' \vdash^S \forall(A',P') : \mathrm{El}(\mathrm{Prop})) \\ &= \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prf}(\forall(A',P')))) \end{aligned}$$

but this follows by construction.

- $M \equiv \Lambda$. We know that Λ is normalizing, so we need to show that if δ is a renaming from Δ' to Δ then $V[\![\Lambda]\!]\rho\Delta(\delta)$ is in its value set.

  We again know by inductive hypothesis that each of the domains is a saturated set. Furthermore, if

$$\begin{aligned} (A,v_A) &\in [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ (P,v_P) &\in [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)]\Delta'(\mathrm{id}_{\Delta'}) \\ (g,v_g) &\in [\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)][P:=(P,v_P)]\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

  then $\Delta' \vdash^S A \rightarrow^{\mathrm{nf}} A' : \mathrm{Type}$, $\Delta' \vdash^S P \rightarrow^{\mathrm{nf}} P' : (\mathrm{El}(A'))\mathrm{El}(\mathrm{Prop})$ and $\Delta' \vdash^S g \rightarrow^{\mathrm{nf}} g' : (x{:}\mathrm{El}(A'))\mathrm{El}(\mathrm{Prf}(P'(x)))$ by Generation and Uniqueness of Normal Forms. Hence $\Delta' \vdash^S \Lambda(A,P,g) \rightarrow^{\mathrm{nf}} \Lambda(A',P',g') : \mathrm{El}(\mathrm{Prf}(\forall(A',P')))$.

  Finally,

$$\begin{aligned} & V[\![\Lambda]\!]\rho\Delta(\delta)((A,v_A),(P,v_P),(g,v_g)) = * \\ \in\ & \{*\} = V(\Delta' \vdash^S \Lambda(A',P',g') : \mathrm{El}(\mathrm{Prf}(\forall(A',P')))) \end{aligned}$$

- $M \equiv \mathrm{E}_\forall$. We know that the domains are saturated sets by inductive hypothesis.

  Suppose δ is a renaming from Δ' to Δ and

$$\begin{aligned} (A,v_A) &\in [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ (P,v_P) &\in [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)]\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

and that (R, v_R), (f, v_f) and (M, v_M) are in the interpretations of the appropriate kinds.

By definition of uniform semantic object and the interpretation of Π, we know that $\Delta' \vdash^S R(M) \to^{\rm nf} B : \mathrm{El}(\mathrm{Prop})$ and that $(M, v_M) \downarrow (N, v_M)$ implies $\Delta' \vdash^S R(M) \downarrow R(N)\ [B] : \mathrm{El}(\mathrm{Prop})$. Also, $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f) \to^{\rm nf} \mathrm{E}_\forall(A'', P'', R', f') : (M{:}\mathrm{El}(\mathrm{Prf}(\forall(A'', P''))))\mathrm{El}(B_M)$.

We show that $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, M) \to^{\rm nf} Q : B$ by induction on the construction of $[\![\mathrm{El}(\mathrm{Prf}(\forall(A, P)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'(\mathrm{id}_{\Delta'})$, from which it follows that

$$(\mathrm{E}_\forall(A, P, R, f, M), \lambda\delta'.*) \in \mathrm{SO}_{\Delta'}(B)$$

- M is a base term such that $\Delta' \vdash^S M \to^{\rm nf} Q : \mathrm{El}(\mathrm{Prf}(\forall(A', P')))$. Then $\mathrm{E}_\forall(A, P, R, f, M)$ is a base term and $\mathrm{E}_\forall(A, P, R, f)$ is an object-level pre-redex, so $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, M) \to^{\rm nf} \mathrm{E}_\forall(A'', P'', R', f, Q) : \mathrm{El}(\mathrm{Prf}(B))$ by (*S-base*).
- $\Delta' \vdash^S M \to^{\rm wh} N : \mathrm{El}(\mathrm{Prf}(\forall(A', P')))$ and

  $$(N, v_N) \in [\![\mathrm{El}(\mathrm{Prf}(\forall(A, P)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'(\mathrm{id}_{\Delta'})$$

  By inductive hypothesis $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, N) \to^{\rm nf} Q : \mathrm{El}(\mathrm{Prf}(B))$, and $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, M) \to^{\rm wh} \mathrm{E}_\forall(A, P, R, f, N) : \mathrm{El}(\mathrm{Prf}(B))$ by (*S-base*). Hence $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, M) \to^{\rm nf} Q : \mathrm{El}(\mathrm{Prf}(B))$ by (*S-WH*).
- $M \equiv \Lambda(A', P', g)$, where $\Delta' \vdash^S A \downarrow A'\ [A''] : \mathrm{Type}$, $\Delta' \vdash^S P \downarrow P'\ [P''] : (\mathrm{El}(A''))\mathrm{El}(\mathrm{Prop})$ and

  $$(g, v_g) \in [\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'$$

  Since this last set is saturated by inductive hypothesis, we know that $\Delta' \vdash^S g \to^{\rm nf} g' : (x{:}\mathrm{El}(A''))\mathrm{El}(\mathrm{Prf}(P'_x))$, where $\Delta' \vdash^S P'(x) \to^{\rm nf} P'_x : \mathrm{El}(\mathrm{Prop})$, so $\Delta' \vdash^S \Lambda(A, P, g) \downarrow \Lambda(A', P', g)\ [\Lambda(A'', P'', g')] : \mathrm{El}(\mathrm{Prf}(\forall(A'', P'')))$. Then

  $$\begin{array}{rl} & \mathrm{APP}_{\Delta'}((f, v_f), (g, v_g)) \\ \in & [\![\mathrm{El}(\mathrm{Prf}(R(\Lambda(A, P, g))))]\!]\rho'\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

where ρ' is $\mathrm{mon}_\delta(\rho)$ extended by A, P, R, f and g, and the set is saturated by inductive hypothesis and the definition of the interpretation. Hence $\Delta' \vdash^S f(g) \to^{\mathrm{nf}} Q : \mathrm{El}(\mathrm{Prf}(B))$, and

$$\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, \Lambda(A', P', g)) \to^{\mathrm{wh}} f(g) : \mathrm{El}(\mathrm{Prf}(B))$$

implies that $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, \Lambda(A', P', g)) \to^{\mathrm{nf}} Q : \mathrm{El}(\mathrm{Prf}(B))$ by (*S-WH*).

- $M \equiv \mathcal{M}^X[\bar{\Theta}]$. If $[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta$ is defined then $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \to^{\mathrm{nf}} \Theta'_j$ for $1 \le j \le n$, so $\Delta \vdash^S \widehat{\phi}(\mathcal{M}^X[\bar{\Theta}]) \to^{\mathrm{nf}} \mathcal{M}^Y[\bar{\Theta}'] : \mathrm{Type}$ by (*S-κ*).

  Furthermore, the set defined is a saturated set at each renaming δ from Δ' to Δ by construction.

- $M \equiv \iota_i^X[\bar{\Theta}]$. If $[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta$ is defined then $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \to^{\mathrm{nf}} \Theta'_j$ for $1 \le j \le n$, so $\Delta \vdash^S \widehat{\phi}(\iota_i^X[\bar{\Theta}]) \to^{\mathrm{nf}} \iota_i^Y[\bar{\Theta}'] : \Theta'_i(\mathcal{M}^Y[\bar{\Theta}'])$ by (*S-κ*).

  Suppose δ is a renaming from Δ' to Δ and

  $$(M_j, v_j) \in [\![M_j]\!]\mathrm{mon}_\delta(\rho[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])[x_1 := (M_1, v_1)] \ldots \Delta'(\mathrm{id}_{\Delta'})$$

  for $1 \le j \le m$, which means that $\Delta' \vdash^S N_j \to^{\mathrm{nf}} N'_j : B$ with $\Delta' \vdash^S \phi[X := \widehat{\phi}(\mathcal{M}^X[\bar{\Theta}])][\widehat{x_1 := N_1}] \ldots [x_{j-1} := N_{j-1}](M_j) \to^{\mathrm{nf}} B$.

  Each of the domains is a saturated set by inductive hypothesis, so we only need to show that $\Delta' \vdash^S \iota_i^X[\bar{\Theta}](N_1, \ldots, N_n) \to^{\mathrm{nf}} P : \mathrm{El}(\mathcal{M}^X[\bar{\Theta}'])$ and

  $$V[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta(\delta)((M_1, v_1), \ldots, (M_n, v_n)) \in V(\Delta' \vdash^S P : \mathrm{El}(\mathcal{M}^X[\bar{\Theta}']))$$

  The first of these follows by (*S-App*), since $\iota_i^X[\bar{\Theta}](N_1, \ldots, N_j)$ is always weak head normal. The second follows by construction.

- $M \equiv \mathrm{E}^X[\bar{\Theta}]$. If $[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta$ is defined then $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \to^{\mathrm{nf}} \Theta'_j$ for $1 \le j \le n$, so $\Delta \vdash^S \widehat{\phi}(\mathrm{E}^X[\bar{\Theta}]) \to^{\mathrm{nf}} \mathrm{E}^X[\bar{\Theta}'] : A^X[\bar{\Theta}']$ by (*S-κ*).

  Suppose δ is a renaming from Δ' to Δ and (C, v_C), (f_i, v_{f_i}) and (M, v_M) are in the interpretations of the appropriate kinds, where (C, v_C) is uniform.

We already know that $\mathrm{E}^X[\bar{\Theta}](C, f_1, \ldots, f_n)$ is well-typed in UTT^S of the appropriate type, that $\Delta' \vdash^S C(M) \rightarrow^{\mathrm{nf}} B : \mathrm{El}(\mathrm{Prop})$ and that if $(M, v_M) \downarrow (N, v_M)$ then $\Delta' \vdash^S C(M) \downarrow C(N)\ [B] : \mathrm{El}(\mathrm{Prop})$.

We show by transfinite induction on α that $R^\alpha(M, v_M) \in \mathrm{SO}_{\Delta'}(B)$ for all $(M, v_M) \in [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta^\alpha(\delta)$:

- $\alpha = 0$. Then $(M, v_M) \in [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta^0(\delta)$ implies that M is a base term or a weak head expansion. These cases are similar to the proof for $\mathrm{E}_\forall$.
- $\alpha = \mathrm{succ}(\beta)$. By induction on the construction of $[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta^{\mathrm{succ}(\beta)}(\delta)$. The cases for base terms and weak head expansions are again straightforward, so we only demonstrate the proof for the constructor case.

  We know that

  $$(M_j, v_j) \in [\![M_j]\!]\mathrm{mon}_\delta(\rho)[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta^\beta][x_1 := (M_1, v_1)] \ldots \Delta'(\mathrm{id}_{\Delta'})$$

  and $\Delta', Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \downarrow \Theta'_j\ [\Theta''_j]$ for any Y fresh in Δ'.

  Let $\Phi_{i_k}{}^*$ be the interpretation of $\Phi^\natural_{i_k}[A, C, f, z]$ under the valuation $\mathrm{mon}_\delta(\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])[C := (C, v_C)][f := R^\beta][z := (N_{i_k}, v_{i_k})]$. By assumption we know that

  $$\begin{aligned} & \Phi^*_{i_k} \\ \in\ & [\![\Phi^\circ_{i_k}[A, C, z]]\!]\mathrm{mon}_\delta(\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])[C := (C, v_C)][z := (N_{i_k}, v_{i_k})]\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

  for $1 \leq k \leq j$.

  Hence by definition

  $$\begin{aligned} & \mathrm{APP}_\Delta((f_i, v_{f_i}), (N_1, v_1), \ldots, (N_n, v_n), \Phi^*_{i_1}, \ldots, \Phi^*_{i_j}) \\ \in\ & [\![C(\iota^X_i[\bar{\Theta}](x_1, \ldots, x_n))]\!]\rho[x_1 := (N_1, v_1)] \ldots [x_n := (N_n, v_n)]\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

  so $\Delta' \vdash^S f_i(N_1, \ldots, N_n, \widehat{\phi}(\Phi^\natural[\mathcal{M}^X[\bar{\Theta}], C, \mathrm{E}^X[\bar{\Theta}](C, f_1, \ldots, f_n), N_{i_1}]), \ldots) \rightarrow^{\mathrm{nf}} P : B$.
- Limit α. This follows by inductive hypothesis.

- $M \equiv \mathrm{U}$. We know that $\Delta \vdash^S \mathrm{U} \rightarrow^{\mathrm{nf}} \mathrm{U} : \mathrm{Type}$ by (*S-C*).

Suppose δ is a renaming from Δ' to Δ. Then $V[\![\mathrm{U}]\!]\rho\Delta(\delta) \in \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{U})) = V(\Delta' \vdash^S \mathrm{U} : \mathrm{Type})$ by construction.

- $M \equiv \mathrm{prop}$. Similar to Prop.

- $M \equiv \mathrm{prf}$. Similar to Prf.

- $M \equiv \mu^X[\bar{\Theta}]$. We know that $\Delta \vdash^S \widehat{\phi}(\mu^X[\bar{\Theta}]) \rightarrow^{\mathrm{nf}} \mu^X[\bar{\Theta}''] : \mathrm{U}$ by (S-μ), where the conditions for the interpretation being well-defined are sufficient for the application of the rule.

  The second component is similar to $\mathcal{M}^X[\bar{\Theta}]$.

- $M \equiv \mathrm{T}$. We know that $\Delta \vdash^S \mathrm{T} \rightarrow^{\mathrm{nf}} \mathrm{T} : (\mathrm{El}(\mathrm{U}))\mathrm{Type}$ by (S-C).

  We need to show that $V[\![\mathrm{T}]\!]\rho\Delta(\delta) \in V(\Delta' \vdash^S \mathrm{T} : (\mathrm{El}(\mathrm{U}))\mathrm{Type})$ if δ is a renaming from Δ' to Δ, where the domain is a saturated set by inductive hypothesis. This means we need to show that if $(N, v_N) \in \mathit{dom}(V[\![\mathrm{T}]\!]\rho\Delta(\delta))$ then $V[\![\mathrm{T}]\!]\rho\Delta(\delta)(N, v_N) \in V(\Delta' \vdash^S Q : \mathrm{Type})$, where $\Delta' \vdash^S \mathrm{T}(N) \rightarrow^{\mathrm{nf}} Q : \mathrm{Type}$. We show this by induction on the construction of $[\![\mathrm{U}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$:

  - N is a base term such that $\Delta' \vdash^S N \rightarrow^{\mathrm{nf}} P : \mathrm{El}(\mathrm{U})$. Then $\mathrm{T}(N)$ is a base term as well, so $\Delta' \vdash^S \mathrm{T}(N) \rightarrow^{\mathrm{nf}} \mathrm{T}(P) : \mathrm{Type}$ and

    $$V(\Delta' \vdash^S P : \mathrm{El}(\mathrm{U})) = \{*\} = V(\Delta' \vdash^S \mathrm{T}(P) : \mathrm{Type})$$

  - $\Delta' \vdash^S M \rightarrow^{\mathrm{wh}} N : \mathrm{El}(\mathrm{U})$ and $(N, v_N) \in [\![\mathrm{U}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$. By inductive hypothesis there is a Q such that $\Delta' \vdash^S \mathrm{T}(N) \rightarrow^{\mathrm{nf}} Q : \mathrm{Type}$ and $V[\![\mathrm{T}]\!]\rho\Delta(\delta)(N, v_N) \in V(\Delta' \vdash^S Q : \mathrm{Type})$. Then $\Delta' \vdash^S \mathrm{T}(M) \rightarrow^{\mathrm{nf}} Q : \mathrm{Type}$ by (S-*base*) and (S-*WH*), and

    $$V[\![\mathrm{T}]\!]\rho\Delta(\delta)(M, v_N) = v_N = V[\![\mathrm{T}]\!]\rho\Delta(\delta)(N, v_N)$$

  - $N \equiv \mathrm{prop}$. Then $\Delta' \vdash^S \mathrm{T}(\mathrm{prop}) \rightarrow^{\mathrm{nf}} \mathrm{Prop} : \mathrm{Type}$ and

    $$V(\Delta' \vdash^S \mathrm{prop} : \mathrm{El}(\mathrm{U})) = \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prop})) = V(\Delta' \vdash^S \mathrm{Prop} : \mathrm{Type})$$

– $N \equiv \mathrm{prf}(P)$. We know by Generation that $\Delta' \vdash^S \mathrm{prf}(P) \to^{\mathrm{nf}} \mathrm{prf}(P') :$ $\mathrm{El}(\mathrm{U})$ implies that $\Delta' \vdash^S P \to^{\mathrm{nf}} P' : \mathrm{El}(\mathrm{Prop})$, so $\Delta' \vdash^S \mathrm{Prf}(P) \to^{\mathrm{nf}}$ $\mathrm{Prf}(P') : \mathrm{Type}$ and $\Delta' \vdash^S \mathrm{T}(\mathrm{prf}(P)) \to^{\mathrm{wh}} \mathrm{Prf}(P) : \mathrm{Type}$ imply $\Delta' \vdash^S \mathrm{T}(\mathrm{prf}(P)) \to^{\mathrm{nf}} \mathrm{Prf}(P') : \mathrm{Type}$ by (*S-WH*).

If P' is a base term then

$$V(\Delta' \vdash^S \mathrm{prf}(P') : \mathrm{El}(\mathrm{U})) = \{*\} = V(\Delta' \vdash^S \mathrm{Prf}(P') : \mathrm{Type})$$

and if P' is not a base term then

$$\begin{aligned} V(\Delta' \vdash^S \mathrm{prf}(P') : \mathrm{El}(\mathrm{U})) &= \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prf}(P'))) \\ &= V(\Delta' \vdash^S \mathrm{Prf}(P') : \mathrm{Type}) \end{aligned}$$

– $N \equiv \mu^X[\bar{\Theta}]$. We know by Generation that $\Delta' \vdash^S \mu^X[\bar{\Theta}] \to^{\mathrm{nf}} \mu^X[\bar{\Theta}'] :$ $\mathrm{El}(\mathrm{U})$ implies that $\Delta', X{:}\mathrm{Type} \vdash^S \Theta_i \to^{\mathrm{nf}} \Theta_i'$ for $1 \leq i \leq n$, so $\Delta' \vdash^S$ $\mathcal{M}^X[\bar{\Theta}] \to^{\mathrm{nf}} \mathcal{M}^X[\bar{\Theta}'] : \mathrm{Type}$ and $\Delta' \vdash^S \mathrm{T}(\mu^X[\bar{\Theta}]) \to^{\mathrm{wh}} \mathcal{M}^X[\bar{\Theta}] : \mathrm{U}$ imply $\Delta' \vdash^S \mathrm{T}(\mu^X[\bar{\Theta}]) \to^{\mathrm{nf}} \mathcal{M}^X[\bar{\Theta}'] : \mathrm{Type}$ by (*S-WH*).

Furthermore

$$\begin{aligned} V(\Delta' \vdash^S \mu^X[\bar{\Theta}'] : \mathrm{El}(\mathrm{U})) &= \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathcal{M}^X[\bar{\Theta}'])) \\ &= V(\Delta' \vdash^S \mathcal{M}^X[\bar{\Theta}'] : \mathrm{Type}) \end{aligned}$$

Definition 6.5.3 (Interpretation of Contexts) *The interpretation of a context Γ, $[\![\Gamma]\!]\Delta$, is a set of valuations from Δ to Γ defined by induction on the structure of Γ:*

- $[\![()]\!]\Delta =_{\mathrm{df}} \{\epsilon\}$.
- $[\![\Gamma, x{:}A]\!]\Delta =_{\mathrm{df}} \{\rho[x := (M, v_M)] \mid \rho \in [\![\Gamma]\!]\Delta$ *and* $(M, v_M) \in [\![A]\!]\rho\Delta(\mathrm{id}_\Delta)\}$
  - *if* $[\![\Gamma]\!]\Delta$ *is defined and*
  - $[\![A]\!]\rho\Delta$ *is defined for any* $\rho \in [\![\Gamma]\!]\Delta$.

6.6 Properties of the Interpretation

Lemma 6.6.1 (Uniformity) *If ρ is a uniform valuation from Δ to Γ and $[\![M]\!]\rho\Delta$ is defined then $[\![M]\!]\rho\Delta$ is a uniform semantic object, and if ρ_1 and ρ_2 are confluent uniform valuations then $[\![M]\!]\rho_1\Delta \downarrow [\![M]\!]\rho_2\Delta$.*

Proof First, if ρ_1 and ρ_2 are confluent uniform valuations then by definition $\Delta \vdash^S \phi_1(x) \downarrow \phi_2(x)\ [P_x] :\ B_x$ for every $x \in dom(\Gamma)$, so by Adequacy for Reduction $\phi_1(x) \rhd^* P_x$ and $\phi_2(x) \rhd^* P_x$. Let $\phi'(x) =_{\mathrm{df}} P_x$ for every $x \in dom(\Gamma)$. Then $\widehat{\phi_1}(M) \rhd^* \widehat{\phi}'(M)$ and $\widehat{\phi_2}(M) \rhd^* \widehat{\phi}'(M)$, so by Subject Reduction and Uniqueness of Normal Forms $\Delta \vdash^S \widehat{\phi_1}(M) \downarrow \widehat{\phi_2}(M)\ [P] :\ B$. Hence for the second condition we only need to show that $V[\![M]\!]\rho_1\Delta = V[\![M]\!]\rho_2\Delta$ if ρ_1 and ρ_2 are confluent uniform valuations.

We prove the lemma by induction on the structure of M. Many of the cases are straightforward.

- $M \equiv x$. By definition.

- $A \equiv (x{:}A_1)A_2$. By definition if $(M, v_M) \in [\![(x{:}A_1)A_2]\!]\rho\Delta$ then (M, v_M) is uniform.

  Suppose δ is a renaming from Δ' to Δ, δ' is a renaming from Δ'' to Δ' and $(M, v_M) \in [\![(x{:}A_1)A_2]\!]\rho\Delta(\delta)$. We need to show that $\mathrm{mon}_{\delta'}(M, v_M) \in [\![(x{:}A_1)A_2]\!]\rho\Delta(\delta' \circ \delta)$. This follows trivially because if δ'' is a renaming from Δ''' to Δ'' then $\delta' \circ \delta$ is a renaming from Δ''' to Δ'. Hence $[\![(x{:}A_1)A_2]\!]\rho\Delta$ is uniform.

  If ρ_1 and ρ_2 are confluent uniform valuations then by inductive hypothesis we know that the interpretations of A_1 and A_2 are equal for any confluent uniform valuations ρ_1' and ρ_2'. Since these sets determine the construction of the interpretation of $(x{:}A_1)A_2$, we know that the interpretations of $[\![(x{:}A_1)A_2]\!]\rho_1\Delta$ and $[\![(x{:}A_1)A_2]\!]\rho_2\Delta$ must be equal.

- $M \equiv [x{:}A_1]M_0$. We first need to show that $[\![[x{:}A_1]M_0]\!]\rho\Delta$ is uniform.

  (U1) By Coherence and inductive hypothesis $[\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) = [\![A_1]\!]\rho\Delta(\delta)$ is uniform.

  (U2) Suppose $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$. By inductive hypothesis

$$\begin{aligned} & [\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta' \\ = \;& ([N/x]M_0, \lambda\delta'.V[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x_2 := (N_2, v_2)](\delta')) \end{aligned}$$

  is uniform, and therefore so is $\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta([\![[x{:}A_1]M_0]\!]\rho\Delta), (M_2, v_2))$.

  (U3) Suppose $(M_1, v_1) \downarrow (M_2, v_2)$, where both are in $[\![A_1]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''$. Then by inductive hypothesis

$$\begin{aligned} & V[\![M_0]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)[x := (M_1, v_1)]\Delta'' \\ = \;& V[\![M_0]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)[x := (M_2, v_2)]\Delta'' \end{aligned}$$

  so

$$\begin{aligned} & \lambda\delta'.V[\![[x{:}A_1]M_0]\!]\mathrm{mon}_\delta(\rho)\Delta'(\delta')(M_1, v_1) \\ = \;& \lambda\delta'.V[\![[x{:}A_1]M_0]\!]\mathrm{mon}_\delta(\rho)\Delta'(\delta')(M_2, v_2) \end{aligned}$$

  If ρ_1 and ρ_2 are confluent uniform valuations then $\mathrm{mon}_\delta(\rho_1)[x := (M_2, v_2)]$ and $\mathrm{mon}_\delta(\rho_2)[x := (M_2, v_2)]$ are confluent uniform valuations for any $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho_1)\Delta' = [\![A_1]\!]\mathrm{mon}_\delta(\rho_2)\Delta'$. Hence

$$\begin{aligned} & \lambda\delta'.V[\![M_0]\!]\mathrm{mon}_\delta(\rho_1)[x := (M_2, v_2)]\Delta'(\delta') \\ = \;& \lambda\delta'.V[\![M_0]\!]\mathrm{mon}_\delta(\rho_2)[x := (M_2, v_2)]\Delta'(\delta') \end{aligned}$$

- $M \equiv M_1(M_2)$. By inductive hypothesis and the definition of uniformity.

- $A \equiv \mathrm{Type}$. By definition of $\mathrm{USO}_{\Delta'}(\mathrm{Type})$.

- $A \equiv \mathrm{El}(M)$. By inductive hypothesis.

- $M \equiv \forall$. The domains are again clearly uniform by inductive hypothesis. Furthermore, if

$$\begin{array}{rcl} (A, v_A) & \in & [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta' \\ (P, v_P) & \in & [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)]\Delta' \end{array}$$

then the set defined for $V[\![\forall]\!]\rho\Delta(\delta)((A, v_A), (P, v_P))$ includes all $(\Lambda(A', P', g), \lambda\delta'.*)$ such that $\Delta' \vdash^S A \downarrow A'\, [A''] : \mathrm{Type}$ and $\Delta' \vdash^S P \downarrow P'\, [P''] : (\mathrm{El}(A''))\mathrm{El}(\mathrm{Prop})$. Hence it is clear by Uniqueness of Normal Forms that if $(A_1, v_{A_1}) \downarrow (A_2, v_{A_2})$ and $(P_1, v_{P_1}) \downarrow (P_2, v_{P_2})$ then

$$V[\![\forall]\!]\rho\Delta(\delta)((A_1, v_{A_1}), (P_1, v_{P_1})) = V[\![\forall]\!]\rho\Delta(\delta)((A_2, v_{A_2}), (P_2, v_{P_2}))$$

We therefore only need to show that $V[\![\forall]\!]\rho\Delta(\delta)((A, v_A), (P, v_P))$ is uniform by induction on the construction of $V[\![\forall]\!]\rho\Delta(\delta)((A, v_A), (P, v_P))$.

  - M is a base term such that $\Delta' \vdash^S M \rightarrow^{\mathrm{nf}} N : \mathrm{El}(\mathrm{Prf}(\forall(A'', P'')))$. Then any renaming of M is a base term as well.
  - $\Delta' \vdash^S M \rightarrow^{\mathrm{wh}} N : \mathrm{El}(\mathrm{Prf}(\forall(A, P)))$ and $(N, v_N) \in V[\![\forall]\!]\rho\Delta(\delta)((A, v_A), (P, v_P))$. By inductive hypothesis.
  - $g \in [\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'$ implies that $(\Lambda(A', P', g), *) \in V[\![\forall]\!]\rho\Delta(\delta)((A, v_A), (P, v_P))$, where $\Delta' \vdash^S A \downarrow A'\ [A''] : \mathrm{Type}$ and $\Delta' \vdash^S P \downarrow P'\ [P''] : (\mathrm{El}(A''))\mathrm{El}(\mathrm{Prop})$. Then $[\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'$ is uniform by inductive hypothesis.

The following two lemmas about the behavior of the interpretation with respect to free variables and substitution are similar to the lemmas in the construction of the set-theoretic model for *UTT*, Lemma 5.2.2 and Lemma 5.2.3.

Lemma 6.6.2 *If ρ is a valuation from Δ to Γ and γ is an environment for variables then $[\![M]\!](\rho \circ \gamma)\Delta \simeq [\![\hat{\gamma}(M)]\!]\rho\Delta$.*

Proof By induction on the structure of terms, following the proof of Lemma 5.2.2.
□

Lemma 6.6.3 *Let ρ_1 and ρ_2 be two valuations from Δ to Γ. If for all $x \in \mathrm{FV}(M)$ we have that $\rho_1(x) = \rho_2(x)$ then $V[\![M]\!]\rho_1\Delta \simeq V[\![M]\!]\rho_2\Delta$.*

Proof By induction on the structure of terms and types, following Lemma 5.2.3. In particular we again use Lemma 6.6.2 for terms with binders. □

Lemma 6.6.4 (Substitution) *If*

- $\rho = \rho_0\rho_1$ *is a valuation from Δ to $\Gamma_0, [N/z]\Gamma_1$*
- $\rho[z := [\![N]\!]\rho_0\Delta]$ *is a valuation from Δ to $\Gamma_0, z{:}C, \Gamma_1$, and*
- $[\![N]\!]\rho_0\Delta$ *is a semantic object for Δ and C, where $\Delta \vdash^S \widehat{\phi}(C) \to^{\mathrm{nf}} D$,*

then $[\![M]\!]\rho[z := [\![N]\!]\rho_0\Delta]\Delta \succ [\![[N/z]M]\!]\rho\Delta$.

Proof By induction on the structure of M. We give several cases:

- $M \equiv x$, where $x \neq z$. Then $[\![x]\!]\rho[z := [\![N]\!]\rho_0\Delta]\Delta = \rho[z := [\![N]\!]\rho_0\Delta](x) = \rho(x) = [\![x]\!]\rho\Delta = [\![[N/z]x]\!]\rho\Delta$.
- $M \equiv z$. Then $[\![z]\!]\rho[z := [\![N]\!]\rho_0\Delta]\Delta = [\![N]\!]\rho_0\Delta = [\![N]\!]\rho\Delta = [\![[N/z]z]\!]\rho\Delta$ by Lemma 6.6.3.
- $A \equiv (x{:}A_1)A_2$. Suppose $[\![(x{:}A_1)A_2]\!]\rho[z := [\![N]\!]\rho_0\Delta]\Delta$ is defined for some x such that $x \notin dom(\Gamma_0, z{:}C, \Gamma_1)$. Then by definition
  - $\Delta \vdash^S \phi[\widehat{z := \widehat{\phi}}(N)](A_1) \to^{\mathrm{nf}} B_1$,
  - $\Delta, y{:}\phi[\widehat{z := \widehat{\phi}}(N)](A_1) \vdash^S \phi[z := \widehat{\widehat{\phi}(N)}][x := y](A_2) \to^{\mathrm{nf}} B_2$ for y fresh in Δ,
  - $[\![A_1]\!]\mathrm{mon}_\delta(\rho[z := [\![N]\!]\rho_0\Delta])\Delta'$ is defined and
  - $[\![A_2]\!]\mathrm{mon}_\delta(\rho[z := [\![N]\!]\rho_0\Delta])[x := (M_2, v_2)]\Delta'$ is defined for any $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho[z := [\![N]\!]\rho_0\Delta])\Delta'$.

Clearly $\phi[\widehat{z := \widehat{\phi}(N)}](A_1) \equiv \widehat{\phi}([N/z]A_1)$ and

$$\phi[\widehat{z := \widehat{\phi}(N)}][x := y](A_2) \equiv \phi[\widehat{x := y}]([y/x]A_2)$$

Furthermore,

$$\begin{array}{rl} & [\![A_1]\!]\mathrm{mon}_\delta(\rho[z := [\![N]\!]\rho_0\Delta])\Delta' \\ = & [\![A_1]\!]\mathrm{mon}_\delta(\rho)[z := \mathrm{mon}_\delta([\![N]\!]\rho\Delta)]\Delta' \\ = & [\![A_1]\!]\mathrm{mon}_\delta(\rho)[z := [\![N]\!]\mathrm{mon}_\delta(\rho)\Delta']\Delta' \\ = & [\![[N/z]A_1]\!]\mathrm{mon}_\delta(\rho)\Delta' \end{array}$$

by Coherence and inductive hypothesis, and similarly

$$\begin{array}{rl} & [\![A_2]\!]\mathrm{mon}_\delta(\rho[z := [\![N]\!]\rho_0\Delta])[x := (M_2, v_2)]\Delta' \\ = & [\![[N/z]A_2]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta' \end{array}$$

for any $(M_2, v_2) \in [\![[N/z]A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'$.

Hence $[\![[N/z](x{:}A_1)A_2]\!]\rho\Delta$ is defined and equal to $[\![(x{:}A_1)A_2]\!]\rho[z := [\![N]\!]\rho_0\Delta]\Delta$.

- M is a constant. Straightforward, using the inductive hypothesis where appropriate.

- $M \equiv \kappa^X[\bar{\Theta}]$. These are all straightforward by inductive hypothesis and the fact that $\phi[\widehat{z := \widehat{\phi}(N)}](\Theta_i) \equiv \widehat{\phi}([N/z]\Theta_i)$.

Lemma 6.6.5 *Let* $\rho \in [\![\Gamma]\!]\Delta$. *Then:*

- ρ *is uniform, and*

- *if* δ *is a renaming from* Δ' *to* Δ *then* $\mathrm{mon}_\delta(\rho) \in [\![\Gamma]\!]\Delta'$.

Proof By induction on the structure of Γ:

- $\Gamma \equiv ()$. Immediate.

- $\Gamma \equiv \Gamma_0, x{:}A$. We know that $\rho = \rho_0[x := (M, v_M)]$, where $\rho_0 \in [\![\Gamma_0]\!]\Delta$ and $(M, v_M) \in [\![A]\!]\rho_0\Delta(\mathrm{id}_\Delta)$. By inductive hypothesis ρ_0 is uniform and $\mathrm{mon}_\delta(\rho_0) \in [\![\Gamma_0]\!]\Delta'$. Hence $[\![A]\!]\rho_0\Delta$ is uniform by Uniformity, so by Coherence

$$\mathrm{mon}_\delta(M, v_M) \in [\![A]\!]\rho_0\Delta(\delta) = [\![A]\!]\mathrm{mon}_\delta(\rho_0)\Delta'(\mathrm{id}_{\Delta'})$$

$\square$

Lemma 6.6.6 (Substitution for Contexts) *Suppose* $\Gamma_0 \vdash P : C$, $\rho_0 \in [\![\Gamma_0]\!]\Delta$ *and* $[\![P]\!]\rho_0\Delta$ *is defined and in* $[\![C]\!]\rho_0\Delta$. *Then*

- *if* $[\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ *is defined then* $[\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ *is defined, and*

- *if* $\rho_0\rho_1 \in [\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ *then* $\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1 \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$.

Proof By induction on the structure of Γ_1:

- $\Gamma_1 \equiv ()$. Clearly $[\![\Gamma_0]\!]\Delta$ is defined and if $\rho_0 \in [\![\Gamma_0]\!]\Delta$ then $\rho_0[z := [\![P]\!]\rho_0\Delta] \in [\![\Gamma_0, z{:}C]\!]\Delta$.

- $\Gamma_1 \equiv \Gamma_1', x{:}A$. Suppose $[\![\Gamma_0, z{:}C, \Gamma_1', x{:}A]\!]\Delta$ is defined. Then $[\![\Gamma_0, z{:}C, \Gamma_1']\!]\Delta$ is defined and if $\rho_0[z := (M, v_M)]\rho_1' \in [\![\Gamma_0, z{:}C, \Gamma_1']\!]\Delta$ then $[\![A]\!]\rho_0[z := (M, v_M)]\rho_1'\Delta$ is defined. By inductive hypothesis $[\![\Gamma_0, [P/z]\Gamma_1']\!]\Delta$ is defined.

  Suppose $\rho_0\rho_1' \in [\![\Gamma_0, [P/z]\Gamma_1']\!]\Delta$. Then $\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1' \in [\![\Gamma_0, z{:}C, \Gamma_1']\!]\Delta$ by inductive hypothesis, and

$$[\![A]\!]\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1'\Delta = [\![[P/z]A]\!]\rho_0\rho_1'\Delta$$

  by Substitution. Hence $[\![\Gamma_0, [P/z]\Gamma_1', x{:}[P/z]A]\!]\Delta$ is defined and if

$$\rho_0\rho_1'[x := (M, v_M)] \in [\![\Gamma_0, [P/z]\Gamma_1', x{:}[P/z]A]\!]\Delta$$

  then

$$\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1'[x := (M, v_M)] \in [\![\Gamma_0, z{:}C, \Gamma_1', x{:}A]\!]\Delta$$

6.7 Soundness

We use the following lemma to show that the interpretations of constants are in the interpretations of their kinds.

Lemma 6.7.1 *Suppose* $\rho \in [\![\Gamma]\!]\Delta$ *and*

- $[\![(x_1{:}A_1)\ldots(x_n{:}A_n)A]\!]\rho\Delta$ *is a saturated set for* Δ *and* $(x_1{:}B_1)\ldots(x_n{:}B_n)B$,
- (M, v_M) *is a uniform semantic object for* Δ *and* $(x_1{:}B_1)\ldots(x_n{:}B_n)B$ *and*
- *for all renamings* δ *from* Δ' *to* Δ *and*

  $$\begin{array}{rcl} (N_1, v_1) & \in & [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ & \ldots & \\ (N_n, v_n) & \in & [\![A_n]\!]\mathrm{mon}_\delta(\rho)[x_1 := (N_1, v_1)]\ldots[x_{n-1} := (N_{n-1}, v_{n-1})]\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

  we know that

  $$\begin{array}{ll} & (M(N_1, \ldots, N_n), \lambda\delta'.v_M(\delta' \circ \delta)(\mathrm{mon}_{\delta'}(N_1, v_1), \ldots, \mathrm{mon}_{\delta'}(N_n, v_n))) \\ \in & [\![A]\!]\mathrm{mon}_\delta(\rho)[x_1 := (N_1, v_1)]\ldots[x_n := (N_n, v_n)]\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

Then $(M, v_M) \in [\![(x_1{:}A_1)\ldots(x_n{:}A_n)A]\!]\rho\Delta(\mathrm{id}_\Delta)$.

Proof By induction on n:

- $n = 0$. Immediate.
- $n = m + 1$. We know that (M, v_M) is a uniform semantic object.

  Suppose δ is a renaming from Δ' to Δ and $(M_1, v_1) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$. Then $\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta(M, v_M), (N_1, v_1))$ is a uniform semantic object for Δ' and $(x_2{:}B_2)\ldots(x_{m+1}{:}B_{m+1})B$ by definition of uniform semantic object. Furthermore $\mathrm{mon}_\delta(\rho)[x_1 := (N_1, v_1)] \in [\![\Gamma, x_1{:}A_1]\!]\Delta'$ by Lemma 6.6.5 and definition.

We also know that if δ' is a renaming from Δ'' to Δ' then $\mathrm{mon}_{\delta'}(N_1, v_1) \in [\![A_1]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''(\mathrm{id}_{\Delta''})$ by Uniformity and Coherence, so for all

$$\begin{array}{rcl} (N_2, v_2) & \in & [\![A_2]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''(\mathrm{id}_{\Delta''}) \\ & \dots & \\ (N_{m+1}, v_{m+1}) & \in & [\![A_{m+1}]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)[x_2 := (N_2, v_2)]\dots\Delta''(\mathrm{id}_{\Delta''}) \end{array}$$

we know that

$$\begin{array}{ll} & (M(N_1, \dots, N_{m+1}), \\ & \quad \lambda\delta''.v_M(\delta'' \circ (\delta' \circ \delta))(\mathrm{mon}_{\delta''\circ\delta'}(N_1, v_1), \mathrm{mon}_{\delta''}(N_2, v_2), \dots)) \\ \in & [\![A]\!]\mathrm{mon}_\delta(\mathrm{mon}_\delta(\rho)[x_1 := (N_1, v_1)])\dots[x_n := (N_n, v_n)]\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

Hence

$$\begin{array}{ll} & \mathrm{APP}_{\Delta'}((M, v_M), (N_1, v_1)) \\ \in & [\![(x_2{:}A_2)\dots(x_{m+1}{:}A_{m+1})A]\!]\mathrm{mon}_\delta(\rho)[x_1 := (N_1, v_1)]\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

by inductive hypothesis, so by definition

$$(M, v_M) \in [\![(x_1{:}A_1)\dots(x_n{:}A_n)A]\!]\rho\Delta(\mathrm{id}_\Delta)$$

□

Theorem 6.7.2 (Soundness) *If any judgement $\Gamma \vdash J$ holds then $[\![\Gamma]\!]\Delta$ is defined. Furthermore:*

- *If $\Gamma \vdash A$ kind and $\rho \in [\![\Gamma]\!]\Delta$ then $[\![A]\!]\rho\Delta$ is defined.*
- *If $\Gamma \vdash M : A$ and $\rho \in [\![\Gamma]\!]\Delta$ then $[\![M]\!]\rho\Delta$ and $[\![A]\!]\rho\Delta$ are defined and $[\![M]\!]\rho\Delta \in [\![A]\!]\rho\Delta(\mathrm{id}_\Delta)$.*
- *If $\Gamma \vdash A = B$ and $\rho \in [\![\Gamma]\!]\Delta$ then $[\![A]\!]\rho\Delta$ and $[\![B]\!]\rho\Delta$ are defined and $[\![A]\!]\rho\Delta \downarrow [\![B]\!]\rho\Delta$.*
- *If $\Gamma \vdash M = N : A$ and $\rho \in [\![\Gamma]\!]\Delta$ then $[\![M]\!]\rho\Delta$, $[\![N]\!]\rho\Delta$ and $[\![A]\!]\rho\Delta$ are defined, $[\![M]\!]\rho\Delta \in [\![A]\!]\rho\Delta(\mathrm{id}_\Delta)$ and $[\![M]\!]\rho\Delta \downarrow [\![N]\!]\rho\Delta$.*

Proof We prove this by induction on derivations in UTT.

- (Emp). Immediate.

- $(Weak)$. By inductive hypothesis we know that $[\![\Gamma]\!]\Delta$ is defined, and $[\![A]\!]\rho\Delta$ is defined for all $\rho \in [\![\Gamma]\!]\Delta$. Hence $[\![\Gamma, x{:}A]\!]\Delta$ is defined.

- (Var). By inductive hypothesis $[\![\Gamma]\!]\Delta$ is defined. Hence $\Gamma \equiv \Gamma_0, x{:}A, \Gamma_1$ implies that $\rho = \rho_0[x := (M, v_M)]\rho_1$ for some $(M, v_M) \in [\![A]\!]\rho_0\Delta$, where $[\![A]\!]\rho_0\Delta$ is defined, $\Delta \vdash^S \widehat{\phi_0}(A) \to^{\mathrm{nf}} B$ and $[\![A]\!]\rho_0\Delta \in \mathrm{SAT}_\Delta(B)$ by inductive hypothesis, the definition of the interpretation of contexts and Well-Definedness. By Contexts (Lemma 4.5.3) we know that each $x_i \in dom(\Gamma_0, x{:}A, \Gamma_1)$ is distinct and that $\mathrm{FV}(A) \subseteq dom(\Gamma_0)$, so $\rho(y) = \rho_0(y)$ for all $y \in \mathrm{FV}(A)$. Hence $\rho(x) = (M, v_M) \in [\![A]\!]\rho_0\Delta = [\![A]\!]\rho\Delta$ by Lemma 6.6.3.

- (Π). By inductive hypothesis $[\![\Gamma, x{:}A_1]\!]\Delta''$ is defined and $[\![A_2]\!]\rho''\Delta''$ is defined for any $\rho'' \in [\![\Gamma, x{:}A_1]\!]\Delta''$. If $[\![\Gamma, x{:}A_1]\!]\Delta''$ is defined then by definition $[\![\Gamma]\!]\Delta''$ is defined and $[\![A_1]\!]\rho'\Delta''$ is defined for any $\rho' \in [\![\Gamma]\!]\Delta''$.

  Suppose ρ is a valuation from Δ to Γ. Then $\Delta \vdash^S \widehat{\phi}(A_1) \to^{\mathrm{nf}} B_1$ and $[\![A_1]\!]\rho\Delta \in \mathrm{SAT}_\Delta(B_1)$ by Well-Definedness, so $\vdash^S \Delta, y{:}\widehat{\phi}(A_1)$ for any y fresh in Δ, and $\mathrm{weak}_\Delta^{\Delta, y:\widehat{\phi}(A_1)}$ is a renaming from $\Delta, y{:}\widehat{\phi}(A_1)$ to Δ. We know by *(S1)* that

  $$(y, *) \in [\![A_1]\!]\rho\Delta(\mathrm{weak}_\Delta^{\Delta, y:\widehat{\phi}(A_1)})$$

  and $\mathrm{mon}_{\mathrm{weak}_\Delta^{\Delta, y:\widehat{\phi}(A_1)}}(\rho) \in [\![\Gamma]\!]\Delta, y{:}\widehat{\phi}(A_1)$ by Lemma 6.6.5, so

  $$\mathrm{mon}_{\mathrm{weak}_\Delta^{\Delta, y:\widehat{\phi}(A_1)}}(\rho)[x := (y, *)] \in [\![\Gamma, x{:}A_1]\!]\Delta, y{:}\widehat{\phi}(A_1)$$

  Hence $\Delta, y{:}\widehat{\phi}(A_1) \vdash^S \widehat{\phi[x := y]}(A_2) \to^{\mathrm{nf}} B_2$.

  Now suppose δ is a renaming from Δ' to Δ. By Lemma 6.6.5 we know that $\mathrm{mon}_\delta(\rho) \in [\![\Gamma]\!]\Delta'$, so $[\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'$ is defined, uniform and a saturated family of sets for Δ' and B_1. Also, if $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'$ then $\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)] \in [\![\Gamma, x{:}A_1]\!]\Delta'$. Hence $[\![A_2]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'$

is defined and a saturated family of sets for Δ' and B_2' by inductive hypothesis and Well-Definedness, where $\Delta' \vdash^S \phi[\widehat{x := M_2}](A_2) \rightarrow^{\mathrm{nf}} B_2'$. Finally, we know that $\phi[\widehat{x := M_2}](A_2) \rhd^* [M_2/y]B_2$ by Lemma 4.7.13, so $\Delta' \vdash^S [M_2/y]B_2 \rightarrow^{\mathrm{nf}} B_2'$ by Subject Reduction.

- (λ). By similar reasoning as for (Π) we know that $[\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'$ is defined, uniform and a saturated family of sets for Δ' and B_1 for any renaming δ from Δ' to Δ, $\Delta \vdash^S \widehat{\phi}(A_1) \rightarrow^{\mathrm{nf}} B_1$ and $\Delta, y{:}\widehat{\phi}(A_1) \vdash^S \phi[\widehat{x := y}](M_0) \rightarrow^{\mathrm{nf}} P_0 : B_2$.

  Suppose δ is a renaming from Δ' to Δ. Then if $(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$ then we again can show that $\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)] \in [\![\Gamma, x{:}A_1]\!]\Delta'$, so by inductive hypothesis $[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'$ is defined and in $[\![A_2]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\mathrm{id}_{\Delta'})$, and $\Delta' \vdash^S \phi[\widehat{x := M_2}](A_2) \rightarrow^{\mathrm{nf}} B_2'$. This again implies that $\Delta' \vdash^S [M_2/x]B_2 \rightarrow^{\mathrm{nf}} B_2'$, so $[\![[x{:}A_1]M_0]\!]\rho\Delta$ is defined.

  Finally,

$$\begin{aligned} & (\phi[\widehat{x := M_2}](M_0), V[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta') \\ = \;& [\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta' \\ \in \;& [\![A_2]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

  If P_0 is not a base term then

$$\begin{aligned} & V[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta' \\ = \;& \lambda\delta'.V[\![M_0]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\delta') \\ = \;& \lambda\delta'.V[\![M_0]\!]\mathrm{mon}_{\delta'}(\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)])\Delta''(\mathrm{id}_{\Delta''}) \\ = \;& \lambda\delta'.V[\![M_0]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)[x := \mathrm{mon}_{\delta'}(M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''}) \\ = \;& \lambda\delta'.(\lambda(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''(\mathrm{id}_{\Delta''}). \\ & \qquad V[\![M_0]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)[x := (M_2, v_2)]\Delta''(\mathrm{id}_{\Delta''}))(\mathrm{mon}_{\delta'}(M_2, v_2)) \\ = \;& \lambda\delta'.V[\![[x{:}A_1]M_0]\!]\mathrm{mon}_{\delta'\circ\delta}(\rho)\Delta''(\mathrm{id}_{\Delta''})(\mathrm{mon}_{\delta'}(M_2, v_2)) \\ = \;& \lambda\delta'.V[\![[x{:}A_1]M_0]\!]\rho\Delta(\delta' \circ \delta)(\mathrm{mon}_{\delta'}(M_2, v_2)) \end{aligned}$$

  so by *(S2)* we know that

$$\Delta' \vdash^S \widehat{\phi}([x{:}A_1]M_0)(M_2) \rightarrow^{\mathrm{wh}} \phi[\widehat{x := M_2}](M_0) : B_2'$$

implies that

$$\begin{array}{ll} & \mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta([\![[x{:}A_1]M_0]\!]\rho\Delta),(M_2,v_2)) \\ = & (\widehat{\phi}([x{:}A_1]M_0)(M_2),\lambda\delta'.V[\![[x{:}A_1]M_0]\!]\mathrm{mon}_\delta(\rho)\Delta'(\delta'\circ\delta)(\mathrm{mon}_{\delta'}(M_2,v_2))) \\ \in & [\![A_2]\!]\mathrm{mon}_\delta(\rho)[x:=(M_2,v_2)]\Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

If P_0 is a base term then the last steps apply as well.

- (*App*). By inductive hypothesis and Well-Definedness we know that
  - $[\![M_1]\!]\rho\Delta \in [\![(x{:}A_1)A_2]\!]\rho\Delta(\mathrm{id}_\Delta)$, $\Delta \vdash^S \widehat{\phi}((x{:}A_1)A_2) \to^{\mathrm{nf}} (x{:}B_1)B_2$ and $[\![(x{:}A_1)A_2]\!]\rho\Delta$ is a saturated family of sets for Δ and $(x{:}B_1)B_2$, so $\Delta \vdash^S \widehat{\phi}(M_1) \to^{\mathrm{nf}} P_1 : (x{:}B_1)B_2$ and if P_1 is not a base term then $dom(V[\![M_1]\!]\rho\Delta) = [\![A_1]\!]\rho\Delta$, and
  - $[\![M_2]\!]\rho\Delta \in [\![A_1]\!]\rho\Delta(\mathrm{id}_\Delta)$, $\Delta \vdash^S \widehat{\phi}(A_1) \to^{\mathrm{nf}} B_1$, and $[\![A_1]\!]\rho\Delta$ is a saturated family of sets for Δ and B_1.

  If $[\![(x{:}A_1)A_2]\!]\rho\Delta$ is defined then we know that $\Delta \vdash^S \widehat{\phi}([M_2/x]A_2) \to^{\mathrm{nf}} B_2'$, since $[\![M_2]\!]\rho\Delta \in [\![A_1]\!]\rho\Delta(\mathrm{id}_{\Delta'})$. Hence $[\![M_1(M_2)]\!]\rho\Delta$ is defined, and by definition of $[\![(x{:}A_1)A_2]\!]\rho\Delta$ and Lemma 6.6.4

$$[\![M_1(M_2)]\!]\rho\Delta \in [\![A_2]\!]\rho[x:=[\![M_2]\!]\rho\Delta]\Delta(\mathrm{id}_\Delta) = [\![[M_2/x]A_2]\!]\rho\Delta(\mathrm{id}_\Delta)$$

- (Type). Immediate.

- (El). By inductive hypothesis $[\![A]\!]\rho\Delta$ is defined and in $[\![\mathrm{Type}]\!]\rho\Delta(\mathrm{id}_\Delta) = \mathrm{USO}_\Delta(\mathrm{Type})$. Hence $[\![\mathrm{El}(A)]\!]\rho\Delta$ is defined by Uniformity.

- ($=T$). By inductive hypothesis we know that $[\![M]\!]\rho\Delta$ and $[\![A]\!]\rho\Delta$ are defined, $[\![M]\!]\rho\Delta \in [\![A]\!]\rho\Delta(\mathrm{id}_\Delta)$ and $[\![A]\!]\rho\Delta \downarrow [\![B]\!]\rho\Delta$. The result follows trivially.

- ($=R$). By inductive hypothesis we know that $[\![M]\!]\rho\Delta$, $[\![N]\!]\rho\Delta$ and $[\![A]\!]\rho\Delta$ are defined, $[\![M]\!]\rho\Delta = [\![N]\!]\rho\Delta \in [\![A]\!]\rho\Delta(\mathrm{id}_\Delta)$ and $[\![A]\!]\rho\Delta \downarrow [\![B]\!]\rho\Delta$. The result follows trivially.

- (*KRefl*)(*Refl*)(*KSym*)(*Sym*). By inductive hypothesis.

- (*KTrans*)(*Trans*). By inductive hypothesis and Uniqueness of Normal Forms.

- (Π-*Eq*). By inductive hypothesis, where

$$\Delta \vdash^S \widehat{\phi}((x{:}A_1)A_2) \downarrow \widehat{\phi}((x{:}B_1)B_2)\ [(x{:}C_1)C_2]$$

  follows by reasoning similar to (Π).

- (λ-*Eq*). By inductive hypothesis, where

$$\Delta \vdash^S \widehat{\phi}([x{:}A_1]M_0) \downarrow \widehat{\phi}([x{:}B_1]N_0)\ [P] :\ (x{:}C_1)C_2$$

  follows by reasoning similar to (λ).

- (*App-Eq*). By inductive hypothesis we know that if $\rho \in [\![\Gamma]\!]\Delta$ then

$$\begin{array}{rcl}
[\![M_1]\!]\rho\Delta \downarrow [\![N_1]\!]\rho\Delta & \in & [\![(x{:}A_1)A_2]\!]\rho\Delta \\
[\![M_2]\!]\rho\Delta \downarrow [\![N_2]\!]\rho\Delta & \in & [\![A_1]\!]\rho\Delta \\
[\![M_1(M_2)]\!]\rho\Delta & \in & [\![[M_2/x]A_2]\!]\rho\Delta \\
[\![N_1(N_2)]\!]\rho\Delta & \in & [\![[N_2/x]A_2]\!]\rho\Delta
\end{array}$$

  Hence $\Delta \vdash^S \widehat{\phi}(M_1) \downarrow \widehat{\phi}(N_1)\ [P_1] :\ (x{:}B_1)B_2$, $\Delta \vdash^S \widehat{\phi}(M_2) \downarrow \widehat{\phi}(N_2)\ [P_2] :\ B_1$, $\Delta \vdash^S \widehat{\phi}(M_1(M_2)) \to^{\mathrm{nf}} Q :\ B$ and $\Delta \vdash^S \widehat{\phi}(N_1(N_2)) \to^{\mathrm{nf}} R :\ C$. This means that $\widehat{\phi}(M_1(M_2)) \rhd^* P_1(P_2)$ and $\widehat{\phi}(N_1(N_2)) \rhd^* P_1(P_2)$, so $\Delta \vdash^S P_1(P_2) \to^{\mathrm{nf}} Q :\ B$ and $\Delta \vdash^S P_1(P_2) \to^{\mathrm{nf}} R :\ C$ by Subject Reduction. Hence $Q \equiv R$ and $B \equiv C$ by Uniqueness of Normal Forms.

  Furthermore, if P_1 is not a base term then

$$\begin{array}{rcl}
V[\![M_1]\!]\rho\Delta(\mathrm{id}_\Delta)([\![M_2]\!]\rho\Delta) & = & V[\![M_1]\!]\rho\Delta(\mathrm{id}_\Delta)([\![N_2]\!]\rho\Delta) \\
 & = & V[\![N_1]\!]\rho\Delta(\mathrm{id}_\Delta)([\![N_2]\!]\rho\Delta)
\end{array}$$

  by inductive hypothesis and the definition of value sets.

- (β). We know that

$$[\![([x{:}A_1]M_0)(M_2)]\!]\rho\Delta \in [\![A_2]\!]\rho[x := [\![M_2]\!]\rho\Delta]\Delta$$

and

$$\begin{aligned} [\![M_0]\!]\rho[x := [\![M_2]\!]\rho\Delta]\Delta &= (\widehat{\phi}([M_2/x]M_0), V[\![M_0]\!]\rho[x := [\![M_2]\!]\rho\Delta]\Delta) \\ &\in [\![A_2]\!]\rho[x := [\![M_2]\!]\rho\Delta]\Delta \end{aligned}$$

where $[\![A_2]\!]\rho[x := [\![M_2]\!]\rho\Delta]\Delta = [\![[M_2/x]A_2]\!]\rho\Delta$ by Substitution.

We then know that

$$V[\![[M_2/x]M_0]\!]\rho\Delta = V[\![M_0]\!]\rho[x := [\![M_2]\!]\rho\Delta]\Delta = V[\![([x{:}A_1]M_0)(M_2)]\!]\rho\Delta$$

and

$$\Delta \vdash^S \widehat{\phi}(([x{:}A_1]M_0)(M_2)) \downarrow \widehat{\phi}([M_2/x]M_0)\ [P] :\ B_2'$$

where $\Delta \vdash^S \widehat{\phi}([M_2/x]A_2) \to^{\mathrm{nf}} B_2'$, by reasoning similar to that for (λ).

- (η). We know that
  - $[\![M]\!]\rho\Delta \in [\![(x{:}A_1)A_2]\!]\rho\Delta$,
  - $\Delta \vdash^S \widehat{\phi}((x{:}A_1)A_2) \to^{\mathrm{nf}} (x{:}B_1)B_2$,
  - $[\![(x{:}A_1)A_2]\!]\rho\Delta \in \mathrm{SAT}_\Delta((x{:}B_1)B_2)$ and
  - $\mathrm{FV}(M) \subseteq dom(\Gamma)$ by Contexts for *UTT*.

  Then:

$$\begin{aligned} & V[\![M]\!]\rho\Delta \\ = {} & \lambda\delta.V[\![M]\!]\rho\Delta(\delta) \\ = {} & \lambda\delta.V[\![M]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ = {} & \lambda\delta.\lambda(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}).V[\![M]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})(M_2, v_2) \\ = {} & \lambda\delta.\lambda(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}). \\ & \quad V[\![M]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\mathrm{id}_{\Delta'})(M_2, v_2) \\ = {} & \lambda\delta.\lambda(M_2, v_2) \in [\![A_1]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}). \\ & \quad (\lambda\delta'.V[\![M]\!]\mathrm{mon}_\delta(\rho)[x := (M_2, v_2)]\Delta'(\delta')(\mathrm{mon}_{\delta'}(M_2, v_2)))(\mathrm{id}_{\Delta'}) \\ = {} & V[\![[x{:}A_1]M(x)]\!]\rho\Delta \end{aligned}$$

and by Lemma 4.9.16 we know that $\Delta \vdash^S \widehat{\phi}([x{:}A_1]M(x)) \to^{\mathrm{nf}} P \,:\, (x{:}B_1)B_2$ implies that $\Delta \vdash^S \widehat{\phi}(M) \to^{\mathrm{nf}} P \,:\, (x{:}B_1)B_2$, so the result follows by Uniqueness of Normal Forms.

- (El-*Eq*). Straightforward by inductive hypothesis and (S-El).

- (*Ctxt Subst*). By inductive hypothesis $[\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ is defined, $[\![\Gamma_0]\!]\Delta$ is defined and if $\rho_0 \in [\![\Gamma_0]\!]\Delta$ then $[\![P]\!]\rho_0\Delta$ and $[\![C]\!]\rho_0\Delta$ are defined and $[\![P]\!]\rho_0\Delta \in [\![C]\!]\rho_0\Delta$. Hence $[\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ is defined by Lemma 6.6.6.

- (*Kind Subst*). By inductive hypothesis

  - $[\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ is defined,
  - if $\rho \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ then $[\![A]\!]\rho\Delta$ is defined and
  - if $\rho_0 \in [\![\Gamma_0]\!]\Delta$ then $[\![P]\!]\rho_0\Delta$ and $[\![C]\!]\rho_0\Delta$ are defined and $[\![P]\!]\rho_0\Delta \in [\![C]\!]\rho_0\Delta$.

  Hence $[\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ is defined by Lemma 6.6.6, and if $\rho_0\rho_1 \in [\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ then $\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1 \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$. Therefore $[\![A]\!]\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1\Delta = [\![[P/z]A]\!]\rho_0\rho_1\Delta$.

- (*Term Subst*)(*Kind-Eq Subst*)(*Term-Eq Subst*). Similar to (*Kind Subst*).

- (*Kind Subst-Eq*). By inductive hypothesis $[\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ is defined, if $\rho \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$ then $[\![A]\!]\rho\Delta$ is defined and if $\rho_0 \in [\![\Gamma_0]\!]\Delta$ then $[\![P]\!]\rho_0\Delta$, $[\![Q]\!]\rho_0\Delta$ and $[\![C]\!]\rho_0\Delta$ are defined and $[\![P]\!]\rho_0\Delta \downarrow [\![Q]\!]\rho_0\Delta \in [\![C]\!]\rho_0\Delta$.

  Hence $[\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ is defined by Lemma 6.6.6, and if $\rho_0\rho_1 \in [\![\Gamma_0, [P/z]\Gamma_1]\!]\Delta$ then $\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1 \in [\![\Gamma_0, z{:}C, \Gamma_1]\!]\Delta$. Furthermore, $\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1 \downarrow \rho_0[z := [\![Q]\!]\rho_0\Delta]\rho_1$, so

  $$[\![A]\!]\rho_0[z := [\![P]\!]\rho_0\Delta]\rho_1\Delta \downarrow [\![A]\!]\rho_0[z := [\![Q]\!]\rho_0\Delta]\rho_1\Delta$$

  by Uniformity and

  $$[\![[P/z]A]\!]\rho_0\rho_1\Delta \downarrow [\![[Q/z]A]\!]\rho_0\rho_1\Delta$$

  by Substitution.

- (*Term Subst-Eq*). Similar to (*Kind Subst-Eq*).

- (Prop). By Well-Definedness and Uniformity

$$[\![\mathrm{Prop}]\!]\rho\Delta \in \mathrm{USO}_{\Delta}(\mathrm{Type}) = [\![\mathrm{Type}]\!]\rho\Delta$$

- (Prf). $[\![\mathrm{Prf}]\!]\rho\Delta$ is a uniform semantic object by Well-Definedness and Uniformity.

  Suppose δ is a renaming from Δ' to Δ and $(P, v_P) \in \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop})) = [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_{\delta}(\rho)\Delta'(\mathrm{id}_{\Delta'})$. Then

$$\begin{aligned} & \mathrm{APP}_{\Delta'}(\mathrm{mon}_{\delta}([\![\mathrm{Prf}]\!]\rho\Delta), (P, v_P)) \\ = \; & \mathrm{APP}_{\Delta'}([\![\mathrm{Prf}]\!]\mathrm{mon}_{\delta}(\rho)\Delta', (P, v_P)) \\ \in \; & \mathrm{USO}_{\Delta'}(\mathrm{Type}) = [\![\mathrm{Type}]\!]\mathrm{mon}_{\delta}(\rho)\Delta' \end{aligned}$$

  by Coherence and the definition of the interpretation, so

$$[\![\mathrm{Prf}]\!]\rho\Delta \in [\![(\mathrm{El}(\mathrm{Prop}))\mathrm{Type}]\!]\rho\Delta(\mathrm{id}_{\Delta})$$

  by the definition of $[\![(\mathrm{El}(\mathrm{Prop}))\mathrm{Type}]\!]\rho\Delta(\mathrm{id}_{\Delta})$.

- ($\forall$). $[\![\forall]\!]\rho\Delta$ is a uniform semantic object by Well-Definedness and Uniformity. Also, $[\![(A{:}\mathrm{Type})((\mathrm{El}(A))\mathrm{El}(\mathrm{Prop}))\mathrm{El}(\mathrm{Prop})]\!]\rho\Delta$ is defined by inductive hypothesis.

  Suppose δ is a renaming from Δ' to Δ and

$$\begin{aligned} (A, v_A) \; & \in \; [\![\mathrm{Type}]\!]\mathrm{mon}_{\delta}(\rho)\Delta' \\ (P, v_P) \; & \in \; [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_{\delta}(\rho)[A := (A, v_A)]\Delta' \end{aligned}$$

  Then we know that

$$\begin{aligned} & \Delta'(\Delta'(\mathrm{mon}_{\delta}([\![\forall]\!]\rho\Delta))(A, v_A))(P, v_P) \\ \in \; & \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop})) \\ = \; & [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_{\delta}(\rho)\Delta'(\mathrm{id}_{\Delta'}) \end{aligned}$$

  so

$$[\![\forall]\!]\rho\Delta \in [\![(A{:}\mathrm{Type})((\mathrm{El}(A))\mathrm{El}(\mathrm{Prop}))\mathrm{El}(\mathrm{Prop})]\!]\rho\Delta(\mathrm{id}_{\Delta'})$$

  by Lemma 6.7.1.

- (Λ). $[\![\Lambda]\!]\rho\Delta$ is a uniform semantic object by Well-Definedness and Uniformity.

  Suppose δ is a renaming from Δ' to Δ and

  $$\begin{array}{rcl}(A,v_A) & \in & [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ (P,v_P) & \in & [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)]\Delta'(\mathrm{id}_{\Delta'}) \\ (g,v_g) & \in & [\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)][P:=(P,v_P)]\Delta'(\mathrm{id}_{\Delta'})\end{array}$$

  which means that $\Delta' \vdash^S A \downarrow A\ [A'] : \mathrm{Type}$ and $\Delta' \vdash^S P \downarrow P\ [P'] : (\mathrm{El}(A'))\mathrm{El}(\mathrm{Prop})$. By definition of $[\![\forall]\!]\rho\Delta$, $[\![\mathrm{Prf}]\!]\rho\Delta$ and $[\![\mathrm{El}]\!]\rho\Delta$ we know that

  $$\begin{array}{cl} & (\Lambda(A,P,g),\lambda\delta'.*) \\ \in & V[\![\forall]\!]\rho\Delta(\delta)((A,v_A),(P,v_P)) \\ = & V[\![\forall(A,P)]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)][P:=(P,v_P)]\Delta'(\mathrm{id}_{\Delta'}) \\ = & [\![\mathrm{El}(\mathrm{Prf}(\forall(A,P)))]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)][P:=(P,v_P)]\Delta'(\mathrm{id}_{\Delta'})\end{array}$$

  so $[\![\Lambda]\!]\rho\Delta$ is in the interpretation of its kind by Lemma 6.7.1.

- $(\mathrm{E}_\forall)$. $[\![\mathrm{E}_\forall]\!]\rho\Delta$ is a uniform semantic object by Well-Definedness and Uniformity.

  Suppose δ is a renaming from Δ' to Δ and

  $$\begin{array}{rcl}(A,v_A) & \in & [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) \\ (P,v_P) & \in & [\![(\mathrm{El}(A))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)]\Delta'(\mathrm{id}_{\Delta'})\end{array}$$

  and that (R,v_R), (f,v_f) and (M,v_M) are in the interpretations of the appropriate kinds.

  First, since

  $$(R,v_R) \in [\![(\mathrm{El}(\mathrm{Prf}(\forall(A,P))))\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)[A:=(A,v_A)][P:=(P,v_P)]\Delta'(\mathrm{id}_{\Delta'})$$

  we know that

  $$\mathrm{APP}_{\Delta'}((R,v_R),(M,v_M)) \in [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) = \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop}))$$

so $v_R(\mathrm{id}_{\Delta'})(M, v_M) \in \mathrm{SAT}_{\Delta'}(\mathrm{El}(\mathrm{Prf}(S)))$, where $\Delta' \vdash^S R(M) \to^{\mathrm{nf}} S$: $\mathrm{El}(\mathrm{Prop})$. Similar to Well-Definedness, we again know that if $(M, v_M) \downarrow (N, v_N)$ then $\Delta' \vdash^S R(M) \downarrow R(N)\ [S] : \mathrm{El}(\mathrm{Prop})$.

We show that

$$\begin{array}{ll} & (\mathrm{E}_\forall(A, P, R, f, M), \lambda\delta'.*) \\ \in & [\![\mathrm{El}(\mathrm{Prf}(R(M)))]\!]\mathrm{mon}_\delta(\rho)[R := (R, v_R)][M := (M, v_M)]\Delta'(\mathrm{id}_{\Delta'}) \\ = & v_R(\mathrm{id}_{\Delta'})(M, v_M) \end{array}$$

by induction on the construction of

$$[\![\mathrm{El}(\mathrm{Prf}(\forall(A, P)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'(\mathrm{id}_{\Delta'})$$

from which by Lemma 6.7.1 it follows that $[\![\mathrm{E}_\forall]\!]\rho\Delta$ is in the interpretation of its kind:

- M is a base term such that $\Delta' \vdash^S M \to^{\mathrm{nf}} Q$: $\mathrm{El}(\mathrm{Prf}(\forall(A', P')))$. Then $\mathrm{E}_\forall(A, P, R, f, M)$ is a base term, so $(\mathrm{E}_\forall(A, P, R, f, M), \lambda\delta'.*) \in v_R(\mathrm{id}_{\Delta'})(M, v_M)$ by *(S1)*.
- $\Delta' \vdash^S M \to^{\mathrm{wh}} N : \mathrm{El}(\mathrm{Prf}(\forall(A', P')))$ and

  $$(N, v_N) \in [\![\mathrm{El}(\mathrm{Prf}(\forall(A, P)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'$$

  By inductive hypothesis $\mathrm{E}_\forall(A, P, R, f, N) \in v_R(\mathrm{id}_{\Delta'})(N, v_N)$, and $\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, M) \to^{\mathrm{wh}} \mathrm{E}_\forall(A, P, R, f, N)$: $\mathrm{El}(\mathrm{Prf}(S))$, by *(S-base)*. Hence $(\mathrm{E}_\forall(A, P, R, f, M), \lambda\delta'.*) \in v_R(\mathrm{id}_{\Delta'})(N, v_N)$ by *(S2)*. Finally, since $\Delta' \vdash^S M \downarrow N\ [Q] : \mathrm{El}(\mathrm{Prf}(\forall(A', P')))$ and $v_M = \lambda\delta'.* = v_N$, we know that $v_R(\mathrm{id}_{\Delta'})(N, v_N) = v_R(\mathrm{id}_{\Delta'})(M, v_N)$ by definition of uniformity.
- $M \equiv \Lambda(A', P', g)$, where $\Delta' \vdash^S A \downarrow A'\ [A'']$: Type, $\Delta' \vdash^S P \downarrow P'\ [P''] : (\mathrm{El}(A''))\mathrm{El}(\mathrm{Prop})$ and

  $$(g, v_g) \in [\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'(\mathrm{id}_{\Delta'})$$

Then

$$\begin{array}{rl} & \mathrm{APP}_{\Delta'}((f, v_f), (g, v_g)) \\ \in & [\![\mathrm{El}(\mathrm{Prf}(R(\Lambda(A, P, g))))]\!]\rho'\Delta'(\mathrm{id}_{\Delta'}) \\ = & v_R(\mathrm{id}_{\Delta'})(\Lambda(A, P, g), \lambda\delta'.*) \end{array}$$

where ρ' is $\mathrm{mon}_\delta(\rho)$ extended by A, P, R, f and g, and

$$\Delta' \vdash^S \mathrm{E}_\forall(A, P, R, f, \Lambda(A', P', g)) \to^{\mathrm{wh}} f(g) : \mathrm{El}(\mathrm{Prf}(S))$$

so

$$(\mathrm{E}_\forall(A, P, R, f, \Lambda(A', P', g)), \lambda\delta'.*) \in v_R(\mathrm{id}_{\Delta'})(\Lambda(A, P, g), \lambda\delta'.*)$$

Since $[\![(x{:}\mathrm{El}(A))\mathrm{El}(\mathrm{Prf}(P(x)))]\!]\mathrm{mon}_\delta(\rho)[A := (A, v_A)][P := (P, v_P)]\Delta'$ is a family of saturated sets by inductive hypothesis, we know that $\Delta' \vdash^S g \to^{\mathrm{nf}} g' : (x{:}\mathrm{El}(A'))\mathrm{El}(\mathrm{Prf}(P'(x)))$, so

$$\Delta' \vdash^S \Lambda(A, P, g) \downarrow \Lambda(A', P', g)\ [\Lambda(A'', P'', g')] : \mathrm{El}(\mathrm{Prf}(\forall(A', P')))$$

by (*S-App*). Hence

$$v_R(\mathrm{id}_{\Delta'})(\Lambda(A, P, g), \lambda\delta'.*) = v_R(\mathrm{id}_{\Delta'})(\Lambda(A', P', g), \lambda\delta'.*)$$

by definition of uniformity.

- (Prop-*Eq*). Clearly

$$\Delta \vdash^S \widehat{\phi}(\mathrm{E}_\forall(A, P, R, f, \Lambda(A, P, g))) \downarrow \widehat{\phi}(f(g))\ [N] : \mathrm{El}(\mathrm{Prf}(S))$$

where $\Delta \vdash^S R(\Lambda(A, P, g)) \to^{\mathrm{nf}} S : \mathrm{El}(\mathrm{Prop})$. Also, $*$ is the unique value for proofs.

- $(\mathcal{M}^X[\bar{\Theta}])$. By inductive hypothesis $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \to^{\mathrm{nf}} \Theta_j'$, using the same construction as for (Π) and (λ), so $[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta$ is defined.

  By Well-Definedness and Uniformity

$$[\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta \in \mathrm{USO}_\Delta(\mathrm{Type}) = [\![\mathrm{Type}]\!]\rho\Delta(\mathrm{id}_\Delta)$$

- $(\iota_i^X[\bar{\Theta}])$. By inductive hypothesis $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \to^{\mathrm{nf}} \Theta'_j$, so $[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta$ is defined.

  Next, suppose δ is a renaming from Δ' to Δ and

$$\begin{array}{rcl} (N_1, v_1) & \in & [\![\mathrm{El}(M_1)]\!]\mathrm{mon}_\delta(\rho[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])\Delta'(\mathrm{id}_{\Delta'}). \\ & & \ldots \\ (N_n, v_n) & \in & [\![\mathrm{El}(M_n)]\!]\mathrm{mon}_\delta(\rho[X := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta])[x_1 := (N_1, v_1)] \ldots \Delta'(\mathrm{id}_{\Delta'}) \end{array}$$

  By construction we have

$$(\iota_i^X[\bar{\Theta}](N_1, \ldots, N_n), \lambda\delta'.{<}v_1(\delta'), \ldots, v_n(\delta'){>}) \in [\![\mathcal{M}^X[\bar{\Theta}]]\!]\mathrm{mon}_\delta(\rho)\Delta'$$

  so by Lemma 6.7.1 we know that $[\![\iota_i^X[\bar{\Theta}]]\!]\rho\Delta$ is in the interpretation of its kind.

- $(\mathrm{E}^X[\bar{\Theta}])$. By inductive hypothesis $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \to^{\mathrm{nf}} \Theta'_j$.

  We show by transfinite induction on the complexity of $\mathcal{M}^X[\bar{\Theta}]$ that $R^\alpha(M)$ is defined and $R^\alpha(M, v_M) \in v_C(M, v_M)$ for $(M, v_M) \in [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta^\alpha$, and furthermore that

$$\begin{array}{rl} & [\![\Phi^\natural[\mathcal{M}^X[\bar{\Theta}], C, f, z]]\!]\mathrm{mon}_\delta(\rho)[C := (C, v_C)][f := R^\alpha][z := (M, v_M)]\Delta' \\ \in & [\![\Phi^\circ[\mathcal{M}^X[\bar{\Theta}], C, z]]\!]\rho[A := [\![\mathcal{M}^X[\bar{\Theta}]]\!]\rho\Delta][C := (C, v_C)][z := (M, v_M)]\Delta' \end{array}$$

  This proof is similar to that in Lemma 5.3.1. Then by Lemma 6.7.1 we know that $[\![\mathrm{E}^X[\bar{\Theta}]]\!]\rho\Delta$ is in the interpretation of its kind.

- $(\bar{\Theta}\text{-}Eq)$. We need to show that

$$\begin{array}{rl} & [\![\Phi^\natural[\mathcal{M}^X[\bar{\Theta}], C, f, z]]\!]\mathrm{mon}_\delta(\rho)[C := (C, v_C)][f := R^\beta][z := (N_{i_k}, v_{i_k})]\Delta' \\ = & [\![\Phi^\natural[\mathcal{M}^X[\bar{\Theta}], C, f, z]]\!]\mathrm{mon}_\delta(\rho)[C := (C, v_C)][f := R^{\alpha_0}][z := (N_{i_k}, v_{i_k})]\Delta' \end{array}$$

  for $\beta < \alpha_0$, which follows by transfinite induction on α_0.

- $(\kappa\text{-}Eq)$. We know that $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_i) \downarrow \phi[\widehat{X := Y}](\Theta'_i)\ [\Theta''_i]$ for $1 \leq i \leq n$, so $\Delta \vdash^S \widehat{\phi}(\kappa^X[\bar{\Theta}]) \downarrow \widehat{\phi}(\kappa^X[\bar{\Theta}']) \ [\kappa^X[\bar{\Theta}'']] : A^X[\bar{\Theta}'']$. Furthermore, we know for any $\rho_{i,j}$ from Δ to $\Gamma, X{:}\mathrm{Type}, x_1{:}M_{i,1}, \ldots, x_{j-1}{:}M_{i,j-1}$

that

$$[\![M_{i,j}]\!]\rho_{i,j}\Delta = [\![M'_{i,j}]\!]\rho_{i,j}\Delta$$

For $\mathcal{M}^X[\bar{\Theta}]$ and $\mu^X[\bar{\Theta}]$, we know that $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_i) \downarrow \Theta''_i\ [\Theta'''_i]$ if and only if $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta'_i) \downarrow \Theta''_i\ [\Theta'''_i]$, so the interpretations are equal.

For $\mathrm{E}^X[\bar{\Theta}]$, we have the added premisses that

$$[\![\Theta_i^\circ[A, C, z]]\!]\rho'\Delta' = [\![(\Theta'_i)^\circ[A, C, z]]\!]\rho'\Delta'$$

and

$$[\![\Theta_i^\natural[A, C, f, z]]\!]\rho'\Delta' = [\![(\Theta'_i)^\natural[A, C, f, z]]\!]\rho'\Delta'$$

for any valuation ρ' from Δ' to $\Gamma, A{:}\mathrm{Type}, C{:}(A)\mathrm{Type}, f{:}(x{:}A)C(x), z{:}\Theta(A)$, so the interpretations $[\![\mathrm{E}^X[\bar{\Theta}]]\!]\rho\Delta$ and $[\![\mathrm{E}^X[\bar{\Theta}']]\!]\rho\Delta$ are equal.

- (U). By Well-Definedness and Uniformity

  $$[\![\mathrm{U}]\!]\rho\Delta \in \mathrm{USO}_\Delta(\mathrm{Type}) = [\![\mathrm{Type}]\!]\rho\Delta(\mathrm{id}_\Delta)$$

- (prop). By definition of $[\![\mathrm{U}]\!]\rho\Delta$.

- (prf). $[\![\mathrm{prf}]\!]\rho\Delta$ is a uniform semantic object by Well-Definedness and Uniformity.

  Suppose δ is a renaming from Δ' to Δ and

  $$(P, v_P) \in [\![\mathrm{El}(\mathrm{Prop})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'}) = \mathrm{USO}_{\Delta'}(\mathrm{El}(\mathrm{Prop}))$$

  Then by definition of value sets, uniformity and $[\![\mathrm{El}(\mathrm{U})]\!]\mathrm{mon}_\delta(\rho)\Delta'$

  $$\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta([\![\mathrm{prf}]\!]\rho\Delta), (P, v_P)) = (\mathrm{prf}(P), v_P) \in [\![\mathrm{El}(\mathrm{U})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$$

  so the result follows by Lemma 6.7.1.

- ($\mu^X[\bar{\Theta}]$). By inductive hypothesis $\Delta, Y{:}\mathrm{Type} \vdash^S \phi[\widehat{X := Y}](\Theta_j) \downarrow \phi[\widehat{X := Y}](\Theta'_j)\ [\Theta''_j]$, where for all $(\Gamma', A) \in \textsc{Types}_\Gamma(\Theta'_j)$ there is an a such that $\mathrm{T}(a) \equiv A$. This implies that for all $(\Gamma', A) \in \textsc{Types}_\Delta(\phi[\widehat{X := Y}](\Theta'_j))$ there is an a such that $\mathrm{T}(a) \equiv A$. Hence $[\![\mu^X[\bar{\Theta}]]\!]\rho\Delta$ is defined.

- (T). We know that $[\![\mathrm{T}]\!]\rho\Delta$ is a uniform semantic object by Well-Definedness and Uniformity.

  Suppose δ is a renaming from Δ' to Δ and $(S, v_S) \in [\![\mathrm{El}(\mathrm{U})]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$. Then by definition of value sets and uniformity

  $$\mathrm{APP}_{\Delta'}(\mathrm{mon}_\delta([\![\mathrm{T}]\!]\rho\Delta), (S, v_S)) \in \mathrm{USO}_{\Delta'}(\mathrm{Type}) = [\![\mathrm{Type}]\!]\mathrm{mon}_\delta(\rho)\Delta'(\mathrm{id}_{\Delta'})$$

  so the result follows by Lemma 6.7.1.

- (T-prop)(T-prf)(T-$\mu^X[\bar{\Theta}]$). These follow by construction of the interpretation.

$\square$

Lemma 6.7.3 *If* $\vdash \Gamma$ *then* $\lambda x.(x, \lambda\delta.*) \in [\![\Gamma]\!]\Gamma$.

Proof By induction on the structure of Γ:

- $\Gamma \equiv ()$. Immediate.

- $\Gamma \equiv \Gamma_0, x{:}A$. By Context Validity (Lemma 4.5.2) and inductive hypothesis we know that $\lambda x.(x, \lambda\delta.*) \in [\![\Gamma_0]\!]\Gamma_0$. Hence $\Gamma_0 \vdash A$ kind trivially, so by Soundness $[\![A]\!]\lambda x.(x, \lambda\delta.*)\Gamma_0$ is defined, which means that there is a B such that $\Gamma_0 \vdash^S A \rightarrow^{\mathrm{nf}} B$ and $[\![A]\!]\lambda x.(x, \lambda\delta.*)\Gamma_0 \in \mathrm{SAT}_{\Gamma_0}(B)$. We then know that $\mathrm{weak}^{\Gamma}_{\Gamma_0}$ is a renaming and

  $$\mathrm{mon}_{\mathrm{weak}^{\Gamma}_{\Gamma_0}}(\lambda x.(x, \lambda\delta.*)) \in [\![\Gamma_0]\!]\Gamma$$

  by Lemma 6.6.5. Finally, $(x, \lambda\delta.*) \in [\![A]\!]\mathrm{mon}_{\mathrm{weak}^{\Gamma}_{\Gamma_0}}(\lambda x.(x, \lambda\delta.*))\Gamma$ by Soundness and *(S1)*, so the result follows.

$\square$

Corollary 6.7.4

- *If* $\Gamma \vdash A$ kind *then there is a* B *such that* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$.

- *If* $\Gamma \vdash M : A$ *then there are* B, P *such that* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ *and* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : B$.

- *If* $\Gamma \vdash A = B$ *then there is a* C *such that* $\Gamma \vdash^S A \downarrow B\ [C]$.

- *If* $\Gamma \vdash M = N : A$ *then there are* B, P *such that* $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$ *and* $\Gamma \vdash^S M \downarrow N\ [P] : B$.

Proof We consider the case of $\Gamma \vdash A$ kind. By Lemma 6.7.3 there is a valuation from Γ to Γ, so by Soundness we know that $[\![A]\!]\rho\Delta$ is defined. By Well-Definedness there is a B such that $\Gamma \vdash^S A \rightarrow^{\mathrm{nf}} B$.

6.8 Consequences of Soundness

Several important results for *UTT* follow from soundness and completeness for UTT^S and the metatheory developed in Chapter 4.

Corollary 6.8.1 (Unicity of Types) *If* $\Gamma \vdash M : A$ *and* $\Gamma \vdash M : B$ *then* $\Gamma \vdash A = B$.

Proof By Soundness, Uniqueness of Normal Forms and Completeness. □

Corollary 6.8.2 (Uniqueness of Product Formation) *If* $\Gamma \vdash (x{:}A_1)A_2 = (x{:}B_1)B_2$ *then* $\Gamma \vdash A_1 = B_1$ *and* $\Gamma, x{:}A_1 \vdash A_2 = B_2$.

Proof By Soundness, Generation, Context Replacement and Completeness. □

Corollary 6.8.3 (Kind Correctness) *If* $\Gamma \vdash M : A$ *then* $\Gamma \vdash A$ kind.

Corollary 6.8.4 (Strong Normalization) *If* $\Gamma \vdash M : A$ *then* M *is strongly normalizing.*

Corollary 6.8.5 (Subject Reduction) *If* $\Gamma \vdash M : A$ *and* $M \rhd^* N$ *then* $\Gamma \vdash N : A$.

Corollary 6.8.6 (Church–Rosser) *If* $\Gamma \vdash M = N : A$ *then there is a* P *such that* $M \rhd^* P$ *and* $N \rhd^* P$.

We can also use the above results to show the decidability of type-checking and the equality reflection theorem (Theorem 3.4.1), using standard techniques [51,47].

Chapter 7

Conclusions and Further Work

7.1 Conclusions

We have introduced a new class of formal systems, typed operational semantics, which present type theory from a computational perspective. In doing so, we have arrived at a strategy for studying computation in type theory opposite to the frequently adopted approach of removing type information from the proof of normalization. We have demonstrated that a specific such system gives a new treatment of fundamental results which have been difficult to establish for systems with dependent types, including strong normalization, Church–Rosser and subject reduction. In our opinion, the benefits of our approach in the study of the basic metatheory justify the added complexity in the proof of soundness. Furthermore, the specific typed operational semantics that we study itself gives a precise, coherent description of the relationship between typing and reduction.

We have studied the idea of typed operational semantics in the context of two type theories. The basic similarity of the treatment of these two theories suggests that our approach may provide a uniform treatment of the metatheory for normalizing type theories.

We have developed a basic metatheoretic understanding of a sophisticated type theory with the Logical Framework, impredicativity, inductive types and a predicative universe. We hope that the theoretical understanding of inductive types as presented in this thesis and elsewhere will provide a foundation for a more practical investigation of the strength of type theory as a specification and programming language.

7.2 Further Work

In the semantic explanation of judgements, two terms are equal in a type if their canonical forms are equal relative to a decidable notion of equality for that type. Coquand's algorithm for testing conversion [15] uses a bisimilarity relation to compare terms which are η-convertible, and the normalization proof uses a logical relation to show that equal terms are bisimilar. Although we have demonstrated in this thesis that the additional machinery of bisimilarity is not necessary for η-equality, there seem to be situations where an equality other than syntactic identity of normal forms may be essential.

One class of such systems is typed calculi with added equalities or reductions representing uniqueness conditions which are known not to have the Church–Rosser property. A well-known system with this problem is the simply typed lambda calculus with product types and a unit type. Goguen and Luo [35] present another system whose metatheoretic properties could be better understood using the technique of bisimilarity and logical relations, with sums, products, the unit type and Martin-Löf's W-type. This system has many of the inductive types which are interesting in practice presented without the added complexity of the kind schema of UTT.

There may also be some inductive types over which we would like a coarser notion of equality. An example of such a type is the type of multisets, which could be represented as lists with the added equality that

$$\mathrm{cons}(a, \mathrm{cons}(b, l)) = \mathrm{cons}(b, (\mathrm{cons}(a, l))$$

It is clear that notions of equality up to associativity and commutativity cannot be modeled simply by reduction, and we need an equivalence on the canonical forms.

The notion of $\beta\eta$-long normal form seems to be preferable when we use the Logical Framework as a framework for defining logics. It would be interesting to see whether the system UTT^S can be changed to incorporate such long normal forms.

Explicit definitions, or *let*-expressions, are a useful tool for introducing abbreviations in ML-like languages which can also be added to type theories. Severi and Poll [71] show that many Pure Type Systems which are strong normalization under β-reduction are also strong normalization if we include definitions, in particular Calculus of Constructions. However, this proof uses considerable metatheoretic power because it relies on ECC being strongly normalizing. It seems that our system UTT^S can be extended to handle explicit definitions and give normalization proofs for systems with explicit definitions directly.

We have not proved the equivalence of the presentation of UTT with judgemental equality and the presentation with conversion. It is easy to show that any judgement which holds in the system with judgemental equality holds in the system with conversion. To show the other direction we need to know Church–Rosser for well-typed terms in the system with conversion, which Coquand [15] overlooks. A solution to this is to follow Salvesen's technique [70] and consider the presentations of these two systems with a type label on application corresponding to the domain of the application: that is, application is of the form $\mathrm{App}(A_1, M_1, M_2)$, and the rule of inference for application is

$$(\mathrm{App})\quad \frac{\Gamma \vdash M_1 \,:\, (x{:}A_1)A_2 \qquad \Gamma \vdash M_2 \,:\, A_1}{\Gamma \vdash \mathrm{App}(A_1, M_1, M_2) \,:\, [M_2/x]A_2}$$

The reduction rule is similar to our equalities for constants in UTT, requiring the labels to be syntactically equivalent for β or η-reduction to take place:

$$\begin{array}{llcll} (\beta) & \mathrm{App}(A_1, [x{:}A_1]M_0, M_2) & \rhd & [M_2/x]M_0 & \\ (\eta) & [x{:}A_1]\mathrm{App}(A_1, M, x) & \rhd & M & \textit{if } x \notin \mathrm{FV}(M) \end{array}$$

Because this system has the weak confluence property, if terms are normalizing then they are also confluent by Newman's lemma, as proved for example by Barendregt [7]. Using this technique to prove equivalence will only be possible in systems where all well-typed terms are normalizing, but the representation of the system UTT^S is already dependent on strong normalization.

The treatment of variables in this thesis does not seem satisfactory, and in particular identifying terms which are equivalent up to change of bound variables seems unacceptable. We believe that the correct syntax-directed presentation of a system should use some representation for bound variables which is unique, such as de Bruijn indices [21] or Stoughton's α-normal forms [74], while the canonical presentation should use variables with names and full α-equality as a rule of inference. In fact we believe that for studying the metatheory the system should be presented as explicitly as possible, which should include rules of equality for weakening and substitution.

We have needed to use a complicated definition of the interpretation function in our proof of soundness. The reason for this is that we want to interpret judgements rather than derivations, since there may be several derivations of the same judgement. However, systems which are syntax-directed, such as UTT^S, do not have this problem, since there is a unique derivation associated with any judgement. It would be interesting to investigate model construction for syntax-directed systems to see whether this is easier than the technique of using a partial interpretation. This may require having a more explicit term structure which reflects the structure of derivations more accurately: for example, using explicit substitutions [55, 78]. In any case, it is not clear that UTT^S is the right syntax-directed system, and Huet's constructive engine [40] or some other system may be better suited to constructing interpretations.

Appendix A

A Complexity Measure for UTT^{-}

In this section we justify the induction principle over the types of UTT^{-} by giving a complexity measure. This principle plays an important role in our proof of normalization, because it allows us to define the range of the interpretation we give in the proof in Chapter 6. This is in contrast to the set-theoretic interpretation, for example, where the range of the interpretation of $\Gamma \vdash^{-} M : A$ is just the universe of sets.

Luo's thesis [44] defines a measure on the types of ECC which is considerably simpler than ours. This is because he is able to show the important substitution property by analysis of the syntax, that

$$\Gamma_0, x{:}A, \Gamma_1 \vdash^{-} B \text{ kind} \quad \text{and} \quad \Gamma_0 \vdash^{-} N : A \quad \text{imply} \quad c([N/x]B) \leq c(B)$$

This property seems to be false for any simple complexity measure for the types of UTT^{-}, in light of examples such as the type

$$\begin{aligned} \Gamma \;\; \vdash^{-} \;\; & [\mathrm{succ}(y)/x]\mathrm{T}(\mathrm{E}_{\mathrm{N}}([x{:}\mathrm{N}]\mathrm{type}_0, \mathrm{N}, [v{:}\mathrm{type}_0]\mathrm{N} \to v, x)) \\ = \;\; & \mathrm{N} \to \mathrm{T}(\mathrm{E}_{\mathrm{N}}([x{:}\mathrm{N}]\mathrm{type}_0, \mathrm{N}, [v{:}\mathrm{type}_0]\mathrm{N} \to v, y)) \end{aligned}$$

Here the first type and the domain of the second type are equivalent except for the change of a variable name, and a simple measure based on the structure of types would have to assign the same value to those two types and hence a higher value for the second type.

Our solution to this interaction between inductive types and universes is to use a variant of the set-theoretic model to provide a complexity measure. The intuition is that in this model, the rank of the interpretations of the constituent types of an inductive type are necessarily smaller than that of the type. However, this is only true in the absence of empty types: for example, the rank of the set of ordered pairs with one element in the empty set is equal to the rank of the empty set. We therefore consider an interpretation where propositions are always non-empty, and we restrict the syntax of inductive types to exclude the empty types.

A better solution would have been to define the interpretation of universes inductively, similar to our definition of inductive types. This would allow us to define a measure on types which simply follows the inductive definition of the universe. Another alternative would have been to use the positive inductive definitions introduced by Allen [3].

A.1 Set-Theoretic Preliminaries

Definition A.1.1 *The* rank *of a set S,* $\mathrm{rank}(S)$*, is the unique operation satisfying the recursive equation* $\mathrm{rank}(S) = \{\mathrm{rank}(T) | T \in S\}$.

Lemma A.1.2 *The following properties hold for the rank function:*

- *If A and B are sets then* $\max\{\mathrm{rank}(A), \mathrm{rank}(B)\} \leq \mathrm{rank}(A + B)$.
- *If A and B are non-empty sets then* $\max\{\mathrm{rank}(A), \mathrm{rank}(B)\} < \mathrm{rank}(A \times B)$.
- *If A is a non-empty set and B is a family of non-empty sets over A then* $\max(\{\mathrm{rank}(A)\} \cup \{\mathrm{rank}(B(a)) | a \in A\}) < \mathrm{rank}(Fa \in A.B(a))$*, for $F \in \{\Sigma, \Pi\}$.*

Proof Straightforward. $\square$

A.2 An Alternative Interpretation of Propositions

The proof-irrelevant semantics we gave in chapter 5 interprets false propositions as the empty set. Although this is an intuitive interpretation for the calculus, for the purpose of defining a complexity measure we wish to avoid giving an empty interpretation to any type. Hence we now give a simple alternative semantics in which there is only one proposition, with a unique proof. This is simply an inconsistent interpretation of propositions.

Definition A.2.1 *We define the following constants:*

$$
\begin{aligned}
[\![\mathrm{Prop}]\!] &=_{\mathrm{df}} \{0\} \\
[\![\mathrm{Prf}]\!] &=_{\mathrm{df}} \lambda v \in [\![\mathrm{Prop}]\!].\{0\} \\
[\![\forall]\!] &=_{\mathrm{df}} \lambda A \in [\![\mathrm{Type}]\!].\lambda P \in A \to [\![\mathrm{Prop}]\!].0 \\
[\![\Lambda]\!] &=_{\mathrm{df}} \lambda A \in [\![\mathrm{Type}]\!].\lambda P \in A \to [\![\mathrm{Prop}]\!].\lambda f \in (\Pi a \in A.[\![\mathrm{Prf}]\!](P(a))).\emptyset \\
[\![\mathrm{E}_\forall]\!] &=_{\mathrm{df}} \lambda A \in [\![\mathrm{Type}]\!].\lambda P \in A \to [\![\mathrm{Prop}]\!].\lambda R \in [\![\forall]\!](A,P) \to [\![\mathrm{Prop}]\!]. \\
&\qquad \lambda f \in (\Pi g \in (\Pi a \in A.[\![\mathrm{Prf}]\!](P(a))).[\![\mathrm{Prf}]\!](R([\![\Lambda]\!](A,P,g)))). \\
&\qquad \lambda z \in [\![\mathrm{Prf}]\!]([\![\forall]\!](A,P)).\emptyset
\end{aligned}
$$

Lemma A.2.2 *If* $\vdash^- \Gamma$, $[\![\Gamma]\!]$ *is defined,* $\rho \in [\![\Gamma]\!]$ *and* $\Gamma \vdash^- M : A$ *then* $[\![M]\!](\rho) \in [\![A]\!](\rho)$.

Proof Straightforward by induction on derivations. □

A.3 UTT^*

We introduce the calculus UTT^*, which corresponds to the sublanguage of UTT^- with no empty types.

Definition A.3.1 (Non-Empty Kind Schema) *Let Γ be a valid context. We say that $\bar{\Theta}$ is a* non-empty Γ-schema, *notation* $\textsc{Sch}^*_{\Gamma;X}(\bar{\Theta})$, *if there is a Θ_i whose arity is* 0.

Recall that the arity of a schema is the sequence of strictly positive operators which occur in the schema. The condition of being a non-empty kind schema says that there must be a constructor for the type which is not recursively defined. This non-recursive constructor gives a base element for the type.

Definition A.3.2 (UTT^*) *The calculus UTT^* is the calculus UTT^- with all occurrences of $\textsc{Sch}^-_{\Gamma;X}(\bar{\Theta})$ replaced by $\textsc{Sch}^*_{\Gamma;X}(\bar{\Theta})$. We use the symbol $\vdash^*$ for the judgements of UTT^*.*

There is an obvious map from derivations in UTT^* to derivations in UTT^-. It is clear that the set-theoretic interpretation lifts directly to this calculus.

Lemma A.3.3 *Under the semantics introduced in section A.2, the following properties hold for UTT^*:*

- *For any valid context Γ there exists a $\rho \in [\![\Gamma]\!]$.*
- *For any valid kind $\Gamma \vdash^* A$* kind *and any $\rho \in [\![\Gamma]\!]$, $[\![A]\!](\rho)$ is non-empty.*

Proof By induction on derivations.

The only rule which could introduce empty interpretations is $(\mathcal{M}^X[\bar{\Theta}])$, but because there must be some Θ_i whose arity is 0 it is clear that $\bar{\Theta}^{\#}_{\rho}(S)$ is strictly larger than S for any set S. $\square$

We can also show the converse, that any well-typed term in UTT whose interpretation involves no empty types is also a well-typed term in UTT^*. However, although this fact gives us an understanding of the relationship between UTT and UTT^*, it plays no role in our development of the syntax, so we omit the details.

Lemma A.3.4 *Suppose* $\textsc{Sch}^*_{\Gamma;X}(\bar{\Theta})$*. If* S *is a non-empty set then for all* $(\Gamma, \Gamma'; A) \in \textsc{Types}_\Gamma(\bar{\Theta})$*, for all* $(\rho, \rho') \in [\![\Gamma, \Gamma']\!]$*,* $\mathrm{rank}([\![A]\!](\rho, \rho')) < \mathrm{rank}(\bar{\Theta}^\#_\rho(S))$*.*

Proof By induction on $\textsc{Sch}^*_{\Gamma;X}(\bar{\Theta})$, where for $\textsc{Sch}^*_{\Gamma,\Gamma';X}(\Theta)$ and $\textsc{Pos}^*_{\Gamma,\Gamma';X}(\Phi)$ we also show that $\Theta^\#_{\rho,\rho'}(\mathrm{S})$ and $\Phi^\#_{\rho,\rho'}(\mathrm{S})$ are non-empty:

- $\textsc{Sch}^*_{\Gamma,\Gamma';X}(\Theta_1, \ldots, \Theta_n)$. By inductive hypothesis and Lemma A.1.2.

- $\textsc{Sch}^*_{\Gamma,\Gamma';X}(X)$. Immediate.

- $\textsc{Sch}^*_{\Gamma,\Gamma';X}((\Phi)\Theta_0)$. By inductive hypothesis and Lemmas A.3.3 and A.1.2.

- $\textsc{Sch}^*_{\Gamma,\Gamma';X}((x{:}A)\Theta_0)$. Suppose $(\rho, \rho') \in [\![\Gamma, \Gamma']\!]$. First,

  $$(\Gamma, \Gamma'; A) \in \textsc{Types}_{\Gamma,\Gamma'}((x{:}A)\Theta_0)$$

  and by Lemma A.1.2 we know that

  $$\begin{array}{rcl} \mathrm{rank}([\![A]\!](\rho, \rho')) & < & \mathrm{rank}(\Sigma a \in [\![A]\!](\rho, \rho').{\Theta_0}^\#_{\rho,\rho',a}(S)) \\ & = & \mathrm{rank}((x{:}A){\Theta_0}^\#_{\rho,\rho'}(S)) \end{array}$$

  where $[\![A]\!](\rho, \rho')$ is non-empty by Lemma A.3.3 and ${\Theta_0}^\#_{\rho,\rho',a}(\mathrm{S})$ is non-empty by inductive hypothesis.

  Furthermore, by inductive hypothesis we know that for any $(\Gamma, \Gamma', x{:}A, \Gamma''; B) \in \textsc{Types}_{\Gamma,\Gamma',x:A}(\Theta_0)$ and any $(\rho, \rho', a, \rho'') \in [\![\Gamma, \Gamma', x{:}A, \Gamma'']\!]$,

  $$\begin{array}{rcl} \mathrm{rank}([\![B]\!](\rho, \rho', a, \rho'')) & < & \mathrm{rank}({\Theta_0}^\#_{\rho,\rho',a}(S)) \\ & < & \mathrm{rank}(\Sigma a \in [\![A]\!](\rho, \rho').{\Theta_0}^\#_{\rho,\rho',a}(S)) \\ & = & \mathrm{rank}((x{:}A){\Theta_0}^\#_{\rho,\rho'}(S)) \end{array}$$

- $\textsc{Pos}^*_{\Gamma,\Gamma';X}((x{:}A)\Phi_0)$. Similar to the previous case.

- $\textsc{Pos}^*_{\Gamma,\Gamma';X}(X)$. Immediate.

□

Definition A.3.5 *We define the complexity of derivable judgements* $\Gamma \vdash^* A$ kind, $c(\Gamma \vdash^* A \text{ kind})$, *as*

$$c(\Gamma \vdash^* A \text{ kind}) =_{\text{df}} \mu i.(\forall \rho \in [\![\Gamma]\!].\text{rank}([\![A]\!](\rho)) \leq i)$$

Lemma A.3.6 *If* $\Gamma \vdash^* A = B$ *then* $c(\Gamma \vdash^* A \text{ kind}) = c(\Gamma \vdash^* B \text{ kind})$.

Proof By Theorem 5.3.1, for any $\rho \in [\![\Gamma]\!]$, if $\Gamma \vdash^* A = B$ then $[\![A]\!](\rho) = [\![B]\!](\rho)$. □

Lemma A.3.7 *If* $\Gamma \vdash^* A$ kind, $\vdash^* \Delta$, *and* $\Gamma \subseteq \Delta$, *then* $c(\Gamma \vdash^* A \text{ kind}) = c(\Delta \vdash^* A \text{ kind})$.

Proof By 5.2.3. □

Lemma A.3.8 *Suppose* $\Gamma, x{:}A \vdash^* B$ kind *and* $\Gamma \vdash^* N : A$. *Then*

$$c(\Gamma \vdash^* [N/x]B \text{ kind}) \leq c(\Gamma, x{:}A \vdash^* B \text{ kind})$$

Proof For any $\rho \in [\![\Gamma]\!]$,

$$\text{rank}([\![[N/x]B]\!](\rho)) = \text{rank}([\![B]\!](\rho[x := [\![N]\!](\rho)]))$$

by Lemma 5.2.4, from which the result follows directly. □

Lemma A.3.9

- *If* $\textsc{Sch}^*_{\Gamma;X}(\bar{\Theta})$ *and for any* $(\Gamma', A) \in \textsc{Types}_\Gamma(\bar{\Theta})$, *where* $\Gamma' \equiv \Gamma, x_1{:}A_1, \ldots, x_n{:}A_n$, *if for all* A_i *we have that* $\Gamma, x_1{:}A_1, \ldots, x_{i-1}{:}A_{i-1} \vdash^* N_i : A_i$, *then*

$$c(\Gamma \vdash^* [\bar{N}/\Gamma']A \text{ kind}) < c(\Gamma \vdash^* \text{El}(\mathcal{M}^X[\bar{\Theta}]) \text{ kind})$$

- *If* $\Gamma \vdash^* (x{:}A_1)A_2$ kind *then* $c(\Gamma \vdash^* A_1 \text{ kind}) < c(\Gamma \vdash^* (x{:}A_1)A_2 \text{ kind})$ *and*

$$c(\Gamma \vdash^* [N/x]A_2 \text{ kind}) < c(\Gamma \vdash^* (x{:}A_1)A_2 \text{ kind})$$

  for any N *such that* $\Gamma \vdash^* N : A_1$.

•

  - $c(\Gamma \vdash^* \mathrm{El}(\mathrm{Prop})\ \mathrm{kind}) < c(\Gamma \vdash^* \mathrm{El}(\mathrm{U})\ \mathrm{kind})$.
  - $c(\Gamma \vdash^* \mathrm{El}(\mathrm{Prf}(P))\ \mathrm{kind}) < c(\Gamma \vdash^* \mathrm{El}(\mathrm{U})\ \mathrm{kind})$ *for any P such that* $\Gamma \vdash^* P : \mathrm{Prop}$.
  - $c(\Gamma \vdash^* \mathrm{El}(\mathcal{M}^X[\bar{\Theta}])\ \mathrm{kind}) < c(\Gamma \vdash^* \mathrm{El}(\mathrm{U})\ \mathrm{kind})$ *if* $\Gamma \vdash^* \mu^X[\bar{\Theta}] : \mathrm{El}(\mathrm{U})$.

Proof The first statement follows from lemmas A.3.8 and A.3.4.

The second statement follows from lemmas A.3.8, A.3.3, and A.1.2 and the definition of $[\![(x{:}A_1)A_2]\!](\rho)$.

The last statement follows by definition of the interpretation of $[\![\mathrm{U}]\!](\rho)$. □

Definition A.3.10 (Inductive Complexity) *Let the complexity for M such that $\Gamma \vdash^- M : \mathcal{M}^X[\bar{\Theta}]$ be defined as*

$$k(\Gamma \vdash^* M : \mathcal{M}^X[\bar{\Theta}]) =_{\mathrm{df}} \mu\alpha.\forall\rho.[\![M]\!](\rho) \in (\bar{\Theta}^{\#}_{\rho})^{\alpha}$$

Lemma A.3.11 $k(\Gamma \vdash^* N_j(P_1, \ldots, P_p) : \mathcal{M}^X[\bar{\Theta}]) < k(\Gamma \vdash^* \iota^X_i[\bar{\Theta}](N_1, \ldots, N_n) : \mathcal{M}^X[\bar{\Theta}])$ *for any $P_1, \ldots, P_p$ of appropriate type and $1 \leq j \leq n$.*

Proof Let ρ be a valuation such that $[\![N_j(P_1, \ldots, P_p)]\!](\rho) \in (\bar{\Theta}^{\#}_{\rho})^{\alpha}$ but not for any $\beta < \alpha$. It is clear by definition of $\iota^X_i[\bar{\Theta}]$ that $k(\Gamma \vdash^* \iota^X_i[\bar{\Theta}](N_1, \ldots, N_n) :)\mathcal{M}^X[\bar{\Theta}]$ is larger than α. □

We recall that this definition of the complexity measure is for a subsystem of *UTT* without empty types. This subsystem includes all of the types we defined using the schemas in Section 3.3.2 except the empty type and the W-type. Modifying the proof to include these types would involve either defining the interpretation of the universe inductively or showing that there is a reduction-preserving map from *UTT* to the subsystem UTT^{*}. For the latter technique, we can simply map a schema family to that schema family extended by the schema X.

Bibliography

[1] Peter Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of mathematical logic*. North-Holland, Amsterdam, 1977.

[2] Peter Aczel. Frege structures and the notions of proposition, truth and set. In *The Kleene symposium*. North-Holland, Amsterdam, 1980.

[3] Stuart Allen. A non–type–theoretic definition of Martin-Löf's types. In *Proceedings of the Symposium on Logic in Computer Science*, Ithaca, New York, June 1987.

[4] Thorsten Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, University of Edinburgh, 1993.

[5] L. Augustsson, Th. Coquand, and B. Nordström. A short description of another logical framework. In G. Huet and G. Plotkin, editors, *Preliminary Proceedings of Logical Frameworks*, 1990.

[6] R. Backhouse. On the meaning and construction of the rules in Martin-Löf's theory of types. In A. Avron et al, editor, *Workshop on General Logic*. LFCS Report Series, ECS-LFCS-88-52, Dept. of Computer Science, University of Edinburgh, 1988.

[7] Henk Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North Holland, revised edition, 1984.

[8] Henk Barendregt. Lambda calculi with types. In Samson Abramsky, D. M. Gabbai, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1991.

[9] S. Berardi. *Type Dependence and Constructive Mathematics.* PhD thesis, Universita di Torino, Italy, 1990.

[10] E. Bishop. *Foundations of Constructive Analysis.* McGraw-Hill, 1967.

[11] C. Böhm and A. Berarducci. Automatic synthesis of typed λ-programs on term algebras. *Theoretical Computer Science*, 39, 1985.

[12] Robert L. Constable, *et al. Implementing Mathematics with the NuPRL Proof Development System.* Prentice–Hall, Englewood Cliffs, NJ, 1986.

[13] Catarina Coquand. From semantics to rules: A machine assisted analysis, 1993. To appear as Chalmers Technical Report.

[14] Thierry Coquand. An analysis of Girard's paradox. In *Proc. of the Symposium on Logic in Computer Science*, pages 227–236, Boston, June 1986.

[15] Thierry Coquand. An algorithm for testing conversion in type theory. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks.* Cambridge University Press, 1991.

[16] Thierry Coquand. Pattern matching with dependent types. In *Workshop on Logical Frameworks*, 1992. Preliminary Proceedings.

[17] Thierry Coquand and Jean Gallier. A proof of strong normalization for the theory of constructions using a Kripke-like interpretation. In *Workshop on Logical Frameworks–Preliminary Proceedings*, 1990.

[18] Thierry Coquand and Gerard Huet. The calculus of constructions. *Information and Computation*, 76:95 – 120, 1988.

[19] Thierry Coquand and Christine Paulin-Mohring. Inductively defined types. *Lecture Notes In Computer Science*, 417, 1990.

[20] Haskell Curry and Robert Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[21] N.G. de Bruijn. Lambda calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the church-rosser theorem. Indag. Mathematics 34, 1972.

[22] N.G. de Bruijn. A survey of the project AUTOMATH. In J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[23] G. Dowek et al. *The Coq Proof Assistent: User's Guide (version 5.6)*. INRIA-Rocquencourt and CNRS-ENS Lyon, 1991.

[24] Peter Dybjer. Inductively defined sets in Martin-Löf's set theory. In Arnon Avron et.al., editor, *Workshop on General Logic*. Dept. of Computer Science, University of Edinburgh, 1988.

[25] Peter Dybjer. An inversion principle for Martin-Löf's type theory. In Peter Dybjer et al, editor, *Workshop on Programming Logic*. Programming Methodology Group, Report 54, 1989.

[26] Peter Dybjer. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*. Cambridge University Press, 1991.

[27] Yuxi Fu. Topics in type theory. Technical report, Dept. of Computer Science, University of Manchester, 1992.

[28] Jean Gallier. On Girard's "Candidats de Reducibilité". In Piergiogio Oddifreddi, editor, *Logic and Computer Science*. Academic Press, 1990.

[29] G. Gentzen. Untersuchungen über das logische schliessen. *Mathematische Zeitschrifte*, 39, 1935.

[30] Herman Geuvers. *Logics and Type Systems*. PhD thesis, Katholieke Universiteit Nijmegen, September 1993.

[31] Herman Geuvers and Mark-Jan Nederhof. A modular proof of strong normalization for the Calculus of Constructions. *Journal of Functional Programming*, 1(2):155–189, April 1991.

[32] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, Université Paris VII, 1972.

[33] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types.* Cambridge University Press, 1989.

[34] Kurt Gödel. Über eine bisher noch nicht benutze erweitrung des finiten standpunktes. *Dialectica*, 12, 1958.

[35] Healfdene Goguen and Zhaohui Luo. Inductive data types: Well-ordering types revisited. In G. Huet and G. Plotkin, editors, *Logical Environments.* Cambridge University Press, 1993.

[36] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Proc. 2nd Ann. Symp. on Logic in Computer Science*, 1987.

[37] Robert Harper, David MacQueen, and Robin Milner. Standard ML. Technical Report ECS–LFCS–86–2, Laboratory for Foundations of Computer Science, March 1986.

[38] Robert Harper and Robert Pollack. Type checking with universes. *Theoretical Computer Science*, 89, 1991.

[39] William Howard. The formulae-as-types notion of construction. In J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic.* Academic Press, 1980.

[40] Gérard Huet. The constructive engine. In R. Narasimhan, editor, *A Perspective in Theoretical Computer Science.* World Scientific Publishing, 1989. Commemorative Volume for Gift Siromoney.

[41] G. Koletsos. Church–Rosser theorem for typed functional systems. *Journal of Symbolic Logic*, 50(3):782–790, 1985.

[42] A. Levy. *Basic Set Theory*. Springer–Verlag, 1979.

[43] Clas Löfwall and Gunnar Sjödin. Strong normalisability in Martin-Löf's type theory. Technical Report R91:09, SICS, 1991.

[44] Zhaohui Luo. *An Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, November 1990.

[45] Zhaohui Luo. A problem of adequacy: conservativity of calculus of constructions over higher-order logic. Technical Report ECS-LFCS-90-121, University of Edinburgh, 1990.

[46] Zhaohui Luo. Program specification and data refinement in type theory. *Mathematical Structures in Computer Science*, 3(3), 1993.

[47] Zhaohui Luo. *Computation and Reasoning*. Oxford University Press, 1994.

[48] Zhaohui Luo and Robert Pollack. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, University of Edinburgh, 1992.

[49] David MacQueen. Using dependent types to express modular structure. In *Proceedings of the 13th ACM Symposium on the Principles of Programming Languages*, 1986.

[50] P. Martin-Löf. Constructive mathematics and computer programming. In L.J. Cohen et.al, editor, *Logic, Methodology and Philosophy of Science VI*, Amsterdam, 1982. North-Holland.

[51] Per Martin-Löf. A theory of types. Unpublished manuscript, 1971.

[52] Per Martin-Löf. An intuitionistic theory of types, 1972. Unpublished manuscript.

[53] Per Martin-Löf. An intuitionistic theory of types: Predicative part. In H. Rose and J. C. Shepherdson, editors, *Logic Colloqium 1973*, pages 73 – 118, 1975.

[54] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[55] Per Martin-Löf. Substitution calculus. Handwritten notes, 1992.

[56] J. McKinna. *Deliverables: a categorical approach to program development in type theory*. PhD thesis, University of Edinburgh, 1992.

[57] J. McKinna and R. Burstall. Deliverables: a categorical approach to program development in type theory. *Proc of Mathematical Foundations of Computer Science, LNCS 711*, 1993.

[58] James McKinna and Robert Pollack. Pure type systems formalized. In M.Bezem and J.F.Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA'93*, pages 289–305. Springer–Verlag, LNCS 664, March 1993.

[59] N. Mendler. Recursive types and type constraints in second-order lambda calculus. In *Proceedings of POPL*, 1987.

[60] N. Mendler. *Recursive Definition in Type Theory*. PhD thesis, Cornell University, 1988.

[61] J.C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions. In *ACM Conference on LISP and Functional Programming*, pages 308–319, August 1986.

[62] John Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North Holland, 1990.

[63] Bengt Nordström, Kent Petersson, and Jan Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, 1990.

[64] Christian-Emil Ore. The Extended Calculus of Constructions (ECC) with inductive datatypes. *Information and Computation*, 1990.

[65] G. Pottinger. Strong normalization for the terms of the theory of constructions. Technical Report TR 11-7, Odyssey Research Associates, Inc., 1987.

[66] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study.* Almqvist and Wiksell, 1965.

[67] John C. Reynolds. Towards a theory of type structure. In *Colloq. sur la Programmation*, volume 19 of *Lecture Notes in Computer Science*, pages 408–423. Springer–Verlag, 1974.

[68] Eike Ritter. *Categorical Abstract Machine for Higher-Order Typed Lambda Calculus.* PhD thesis, University of Cambridge, September 1992.

[69] Anne Salvesen. Typeteori - en studie. Cand. scient. thesis, University of Oslo, 1984.

[70] Anne Salvesen. The Church-Rosser property for pure type systems with $\beta\eta$-reduction, November 1991. unpublished manuscript.

[71] Paula Severi and Erik Poll. Pure type systems with definitions, September 1993.

[72] Jan Smith. An interpretation of Martin-Löf's type theory in a type-free theory of propositions. *Journal of Symbolic Logic*, 49(3), 1984.

[73] Jan Smith. The independence of Peano's fourth axiom from Martin-Löf's type theory without universes. *Journal of Symbolic Logic*, 53(3), 1988.

[74] Allen Stoughton. Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.

[75] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness and Independence Results.* Birkhäuser, 1991.

[76] W. W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic*, 32, 1967.

[77] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium, Lecture Notes in Mathematics, 453*, pages 240–251, 1975.

[78] A. Tasistro. Formulation of Martin-Löf's theory of types with explicit substitutions. Licentiate thesis, 1993.

[79] L. S. van Bentham Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH system.* PhD thesis, Technische Hogeschool Eindhoven, 1977.

[80] L. van Benthem Jutting, James McKinna, and Robert Pollack. Typechecking in pure type systems. submitted for publication, 1993.

[81] D. van Dalen. *The Language Theory of Automath.* PhD thesis, Technical University of Eindhoven, 1980.

[82] D. van Dalen, H.C. Doets, and H. de Swart. *Sets: Naive, Axiomatic and Applied.* International Series in Pure and Applied Mathematics. Pergammon Press, 1978.

[83] Benjamin Werner. A normalization proof for an impredicative type system with large elimination over integers. In *Workshop on Logical Frameworks–Preliminary Proceedings*, pages 341–357, 1992.

Index