A MODULAR HIERARCHY OF LOGICAL FRAMEWORKS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2004

By
Robin Adams
School of Mathematics

Contents

List of Figures

List of Frameworks and Features

Abstract

Logical frameworks — formal systems for the specification and representation of other formal systems — are now a well-established field of research, and the number and variety of logical frameworks is large and growing continuously. In this thesis, I tie several examples of logical frameworks into a single hierarchy.

I begin by introducing an infinite family of new, weak, lambda-free logical frameworks. These systems do not use lambda-abstraction, local definition, or any similar feature; parameterisation, and the instantiation of parameterisation, is taken as basic. These frameworks form conservative extensions of one another; this structure of extension is what I call the modular hierarchy of logical frameworks.

I show how several examples of existing logical frameworks — specifically, the systems PAL and AUT-68 from the AUTOMATH family, the Edinburgh Logical Framework, Martin-Löf's Logical Framework, and Luo's system PAL$^+$ — can be fitted into this hierarchy, in the sense that one of the weak frameworks can be embedded in each as a conservative subsystem. I give several examples of adequacy theorems for object theories in the weak frameworks; these theorems are easier to prove than is usually the case for a logical framework. Adequacy theorems for the systems higher in the hierarchy follow as immediate corollaries.

In the second part of this thesis, I investigate an approach to the design of logical frameworks suggested by the existence of such a hierarchy: that a framework could be built by specifying a set of features, the result of adding any of which to a framework is a conservative extension of the same. I show how all of the weak frameworks from the first part, as well as two of the systems we gave there as examples, can indeed be built in this manner.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Acknowledgements

My thanks should first and foremost go to my supervisor, Peter Aczel, for first introducing me to type theory in what now feels like the dim and distant past, for help and support, both technical and , for encouraging me when it seemed everything I tried led to failure, and for restraining my ambition when I had once again found the single idea that would turn the type theory world upside-down.

Thanks to Zhaohui Luo for the many helpful suggestions and discussions, both in person and by e-mail, over the course of the last few years. Thanks also to the rest of the CARG group at Durham, especially Paul Callaghan and James McKinna, for their hospitality and helpful discussions on several visits. Likewise, thanks go to the Foundations group at Nijmegen, especially Henk Barendregt and Herman Geuvers, for receiving me as a visitor, and for their suggestions and opinions while I was there.

Thanks to many people with whom I have had several fruitful discussions at conferences and over e-mail, particularly Randy Pollack for substantial help during the first part of my PhD course.

I would like to thank my parents, Nêst and Chris, for moral and emotional support, and for providing a safe place I could retreat to now and again. I would like to thank Sunday morning cartoons for the same reasons.

I would finally like to thank a person I have never met: Raymond Smullyan, for helping me find my vocation. It was the accidental discovery of one of his puzzle books in the school library that first introduced me to mathematical logic. I immediately devoured as many of his books, both puzzle books and formal text books, that I could find. He expertly conveyed to me the possibilities, wonder and fun to be found in a mathematical model of mathematics itself. I do not think it has ever since ceased to amaze either of us.

This research was funded by EPSRC research grant number 00801108, and the same number of cups of coffee.

Chapter 1

Introduction

Both mathematical logic and computer science deal with a large number of *formal systems*: systems consisting of the formation and manipulation of syntactic expressions of various kinds. The theories that are the subject matter of mathematical logic are formal systems, as are progamming languages and many of the systems invented for giving semantics to the same. The systems known as type theories (typed lambda calculi, or typing systems) have proven thesmelves relevant to both fields, thanks to the minor miracle that goes by the name of the Curry-Howard isomorphism: that the same system can have two different interpretations, as a logical theory and as a functional programming language.

The field of *logical frameworks* stems from the (ostensibly rather pithy) observation that these formal systems are very similar to one another. Many systems employ the *binding* of variables; and what it means for a variable to be free or bound, for two expressions to be α-convertible (i.e. identical up to relabelling of bound variables), or for an expression to be substituted for a variable in such a way that variable capture is avoided, is essentially the same in each case. Likewise, while the rules of deduction vary from one system of logic to another, what it means for a proposition to be derivable is then the same; in particular, in the various systems of natural deduction, what it means for a hypothesis to be open or discharged is the same between the different systems.

Thanks to this resemblance, it is possible to define *logical frameworks*. A logical framework can be defined as a formal system within which (representations of) other formal systems are built. Another way of phrasing this would be that logical frameworks provide a meta-language for the specification of formal systems. The system being represented is known as the *object system* or *object logic*, to distinguish it from the framework, or *frame*, itself. Type theories have proven themselves very effective at this task, too: the features of the lambda calculus lend themselves well to the representation of both variable binding and substitution, and discharge of hypotheses.

There are several reasons for wishing to use logical frameworks. The most obvious is the saving of effort: rather than building implementations of many different systems separately, one need only implement a logical framework within which all the other systems can be defined, and we obtain implementations of the object systems 'for free'. While this may be something of an overstatement — there may be quite some work involved in defining a system within a framework in such a way that the implementation is useful — nevertheless, a logical framework is a much better starting point than implementing a system from scratch.

Secondly, some features of a system may be easier to specify when that system is defined within a logical framework than when it is defined on its own. For example, Dybjer [Dyb94], and Luo in the theory UTT [Luo94], give methods of defining arbitrary inductive types in systems built within variants of Martin-Löf's logical framework. In each case, essential use is made of the logical framework's product kinds in giving the inductive patterns used to define the types. Plastic [CL] is an example of a recent logical framework designed for representing type theories that feature coercive subtyping

The first logical frameworks were the systems that comprise the AUTOMATH family, which appeared in the late 60's and early 70's [NGV94]. The goal of the AUTOMATH system was to take a mathematical text that had been written in a formal language — a *book*, in AUTOMATH's terminology — and check its correctness. Rather than choose one fixed system of logic, however, AUTOMATH required the definition of the system of logic to be used — which connectives are present, and what the primitive rules of deduction are — to be part of the book itself. De Bruijn, the originator of the AUTOMATH systems, spoke often of the 'AUTOMATH café': like a café that allowed the customer to choose to eat kosher, halal or vegetarian if he or she wished, so AUTOMATH allowed the user to choose to use second-order or first-order, intuitionistic or classical logic as he or she wished. This is an excellent description of the spirit behind logical frameworks.

In 1986, Per Martin-Löf proposed his logical framework as a better setting for the definition of his type theory than the theory of arities which had previously been used. (This logical framework is presented in [NPS90]; the ideas behind it first appeared in [ML85].) The phrase 'logical framework' itself was coined with the appearance of the Edinburgh Logical Framework in 1987 [HHP93]. This system (then named LF but now more usually called ELF to distinguish it from the many other logical frameworks that have been developed since) has stood the test of time well and spawned many variants, such as the Linear Logical Framework LLF [Cer96] and the Concurrent Logical Framework CLF [WCPW03], specialised to handle certain particular forms of object theory. Martin-Löf's logical framework and ELF have been the two most important systems in the subsequent development of the field of logical frameworks by quite some distance.

Since then, the number and variety of logical frameworks has multiplied rapidly. Pfenning [Pfe96, Pfe] maintains an excellent overview of the history and current state of development of the field, to which we are indebted for the following examples and references. In the intervening years there have appeared the frameworks Isabelle, based on higher-order logic and resolution [Pau86, Pau89, NP92]; λProlog, based on logic programming [NM88, MNPS91]; FS_0 [Fef88, MSB93] and ALF [Mag95], based on inductive definitions; and Pi, based on partial inductive definitions [Hal91, Eri93, Eri94]. More frameworks, and more variants of each existing framework, are continually being invented; this situation shows no signs of changing any time soon.

In this thesis, we shall present a method which could conceivably bring a modicum of order to this presently rather chaotic field. We shall define an infinite family of very weak *lambda-free* logical frameworks, which extend one another conservatively in what we shall call a *modular hierarchy.* We shall show how many existing logical frameworks can be fitted into this hierarchy, in the sense that a weak framework can be embedded in each as a conservative subsystem. One advantage of this approach is that the adequacy theorem for an object theory can be proven more easily in a weak framework, and the corresponding result for every system that has been built on top of that weak framework follows as an immediate corollary.

The existence of such a hierarchy suggests an approach to the design of logical frameworks: building a framework by specifying a set of features, each of which is defined independently of the other. In the second part of this thesis, we investigate this possibility by reconstructing most of the frameworks with which we dealt in Part I by precisely this method.

1.1 Terminology

We shall deal in this thesis with several different object theories, many of which are type theories, and several different logical frameworks, all of which are type theories. We shall thus often be dealing with several different type theories at once, and it is inevitable that there shall be many opportunities for confusion, as several words, such as 'term', 'type' and 'expression', shall be used by each system, sometimes with subtly different meanings. Hopefully, the context will always resolve any ambiguities that do arise; but, to minimise the chance of confusion, we shall stay faithful to the following scheme of terminology. Where this involves deviating from the terminology usually associated with a particular system, we shall mention this fact explicitly when we first introduce the system.

When representing a formal system S within a logical framework F, S is the *object theory* or *object logic*, and F the *framework* or *frame.* S and F are both *systems.* The *entities* of S are the *terms*, *propositions* and *derivations* when S is a deductive system, and the *terms* and *types* when S is a type theory. These sets — the set of terms, the set of propositions, the set of types, and so forth — shall be called the *syntactic classes* of S. Their members are formed by *(grammatical) constructors*, operators which take entities and return entities, such as the connectives $\wedge$, $\vee$, $\rightarrow$ and quantifiers $\forall$, $\exists$ in deductive systems, or the *type constructors* Π, Σ, W and *term constructors* such as λ in the case of type theories.

We shall only be dealing with frameworks F that are themselves type theories. Each framework F uses a distinguished universe, for which we shall always use the symbol **Type**, whose terms are intended to represent the syntactic classes of S when S is a deductive system, or the types of S when S is a type theory. We shall call the expressions that can occur on the right of the typing judgements of F *kinds*, reserving the word *types* for the terms of kind **Type**; as **Type** is a universe à la Russell, these types shall themselves be kinds. The expressions on the left of the typing judgements of F shall be called the *terms* of F. Thus, when S is a type theory, the terms and types of S are represented by (subsets of) the terms and types of F; hopefully, this overloading of terminology shall clarify more than it confuses. Again, the set of terms, the set of types, and the set of kinds shall be the *syntactic classes* of F.

The one framework for which we deviate from the scheme in the above paragraph is ELF, whose grammar is organised differently to other frameworks; we use its terminology, speaking of *objects*, *families* and *kinds*.

An *expression* shall be any piece of syntax, whether a member of one of a system's syntactic classes, or merely a part of one.

Almost all of our systems use variable binding, and we shall handle the theory of variable binding and relabelling by the usual cowardly convention: we shall ignore it completely, except for the occasional token statement that "we identify —s up to α-conversion". We merrily proceed to assume within any definition or proof that bound variables match one another, or are distinct from free variables, as convenient. If one insists on an official position, we state now that all our variables are to be de Bruijn indices.

In particular, "M and N are *identical*", which we shall write $M \equiv N$ or $M \equiv_\alpha N$, shall always mean that M and N are α-convertible, while "M and N are *convertible*" shall indicate some weaker equivalence relation holds between M and N; this shall almost always be that the judgement $M = N : K$ is derivable, for the appropriate type or kind K. We shall try never to use the ambiguous phrase "M and N are equal", which *prima facie* could mean either.

1.2 The Use of a Logical Framework

We shall be concerned in this thesis with the representation of two classes of object theory within logical frameworks: *deductive systems*, of which we shall use propositional logic, predicate logic and the extension of natural deduction presented by Schroeder-Heister in [SH84] as examples; and *type theories*, of which we shall use Martin-Löf's Type Theory and the Calculus of Constructions as examples. As far as deductive systems are concerned, we restrict our attention to those frameworks in which derivations are modelled as objects, as opposed to those frameworks such as Isabelle in which the framework only records which formulas are derivable. We shall call the former *derivations-as-objects* frameworks, and the latter *derivability-as-primitive* frameworks. To be specific, the frameworks we use as examples are the systems PAL and AUT-68 from AUTOMATH [NGV94], the variant LF of Martin-Löf's Logical Framework [NPS90] developed by Luo [Luo94] in which the lambda-abstractions are typed, the Edinburgh Logical Framework ELF [HHP93], and Luo's recent framework PAL$^+$ [Luo03].

When representing a *deductive theory* S within a logical framework, we begin by declaring a *signature*, which shall typically be a sequence of *constant declarations*. The first part of the signature represents the grammar of S, by introducing a constant of kind **Type** for each syntactic class, and a constant of the appropriate kind for each constructor in the grammar of S. If we denote the constant corresponding to the syntactic class C by $\overline{C}$, we intend that the signature shall represent the grammar of S in such a way that:

> To each entity t of syntactic class C, there corresponds a term $\overline{t}$ of kind $\overline{C}$; further, every term of kind $\overline{C}$ is convertible with a unique term of this form. Further, this translation is injective: if $\overline{t} \equiv \overline{t'}$, then $t \equiv t'$.

The appropriate notion of 'convertible' in these two conditions depends on the framework; for example, it shall be β-convertibility in ELF, and $\beta\eta$-convertibility in Martin-Löf's Logical

Framework. In each of these two cases, the terms of the form $\overline{M}$ shall be the normal forms of the appropriate kind.

The second part of the signature then extends this correspondence to the derivations of S. We provide ourselves with a type for each formula ϕ, whose terms shall represent the derivations of ϕ; this is done by declaring a constant (which we shall always take to be the symbol **Prf**) which takes a term of type $\overline{\text{formula}}$ and returns a type. The signature then proceeds to declare a constant corresponding to each rule of deduction in S, in such a way that:

> To each derivation Δ of the formula ϕ, there corresponds a term $\overline{\Delta}$ of kind $\mathbf{Prf}[\overline{\phi}]$; every term of kind $\mathbf{Prf}[\overline{\phi}]$ is convertible with a unique such term.

We might ask whether a similar injectivity condition holds for derivations as held for entities:

> If $\overline{\Delta} \equiv \overline{\Delta'}$, then Δ and Δ' are the same derivation.

To answer this, we must decide what is meant by two derivations being 'the same'. In fact, the natural definition of identity on declarations is not reflected in the logical framework in this manner. It is an interesting task to make explicit what equivalence relation on derivations is represented by identity of terms in the logical framework; we shall investigate this matter in Chapter 2.

The *contexts* of the judgements of the logical framework under this signature then represent both the free variables in the entities of S, and the open hypotheses in the derivations of S. Thus, if we denote the signature that (we claim) represents S by Σ, we should hope that we are able to prove a result similar to the following:

- Σ is a valid signature. In all our logical frameworks, there shall be such a judgement form for signatures, and we shall always write this judgement as

$$\Sigma \text{ sig} .$$

- For every entity t of syntactic class C,

$$x_1 : \overline{\text{term}}, \ldots, x_n : \overline{\text{term}} \vdash_\Sigma \overline{t} : \overline{C} .$$

- If

$$x_1 : \overline{\text{term}}, \ldots, x_n : \overline{\text{term}} \vdash_\Sigma M : \overline{C} ,$$

  then there is a unique entity t of syntactic class C such that

$$x_1 : \overline{\text{term}}, \ldots, x_n : \overline{\text{term}} \vdash_\Sigma M = \overline{t} : \overline{C} .$$

- The formula ϕ is derivable from hypotheses $\psi_1, \ldots, \psi_m$ if and only if there is a term D such that

$$x_1 : \overline{\text{term}}, \ldots, x_n : \overline{\text{term}}, p_1 : \mathbf{Prf}[\overline{\psi_1}], \ldots, p_m : \mathbf{Prf}[\overline{\psi_m}] \vdash_\Sigma D : \mathbf{Prf}[\overline{\phi}] ,$$

  where $x_1, \ldots, x_n$ include all the free variables in $\psi_1, \ldots, \psi_m$, and ϕ.

The collection of such statements for a given object theory is called the *adequacy theorem* for the representation of that theory in the framework. The exact form that the theorm takes varies with the framework and the object theory. For example, in second-order logic, the context would also contain variables of type $\overline{\text{formula}}$, representing propositional variables, and variables of type $\overline{\text{term}} \to \cdots \to \overline{\text{term}} \to \overline{\text{term}}$ and $\overline{\text{term}} \to \cdots \to \overline{\text{term}} \to \overline{\text{formula}}$, representing function and predicate variables, respectively. Likewise, if the framework did not contain a judgement form for equality, instead using an externally defined relation such as β-convertibily, the third clause above would have to be modified accordingly. We mention specifically the fact that, in many frameworks, we would be able to prove:

> For every term M, if
> $$\Gamma \vdash_\Sigma M : \mathbf{Type}\,,$$
> either there is a unique syntactic class $\overline{C}$ such that
> $$\Gamma \vdash_\Sigma M = \overline{C} : \mathbf{Type}$$
> or there is a unique formula ϕ such that
> $$\Gamma \vdash_\Sigma M = \mathbf{Prf}[\overline{\phi}] : \mathbf{Type}\,.$$

This is not the case, however, in ELF, where the kind **Type** also includes the product types that can be formed from the types $\overline{C}$ and $\mathbf{Prf}[\overline{\phi}]$.

These adequacy theorems are notoriously difficult to prove, often relying on quite strong properties of the framework, such as normalization. Generally speaking, the stronger a logical framework is — that is, the more specialised it is, or the more 'features' it has — the more difficult its adequacy theorems are to prove. This is rarely a problem in practice, as we can prove the properties required once-and-for-all for a given framework. As an example, consider the adequacy theorems for ELF (for first-order logic and higher-order logic) proven in [HHP93]; while the weak normalization of ELF is quite difficult to prove, once it has been proven, the adequacy proofs are then quite straightforward.

The process of representing a type theory in a logical framework follows a similar pattern. Again, the first step is to give a *signature*. In this case, the signature shall consist of a sequence of constant declarations, one for each constructor in the type theory, together with a set of *equations*, one for each computation rule in the type theory. An *equation* is a declaration that two terms of the same kind are henceforth to be convertible; these equations may well be parametrized.

For example, the β-conversion rule would typically be represented by the equation

$$\begin{gathered}(A : \mathbf{Type}, B : A \to \mathbf{Type}, b : (x : A)Bx, a : A)\\ \mathrm{app}(A, B, \lambda(A, B, b), a) = b(a) : B(a)\,.\end{gathered}$$

If we have terms $\overline{A} : \mathbf{Type}$ representing the type A, $\overline{B} : \mathbf{Type}$ representing the type B, $\overline{b} : \overline{B}$ representing the term $b : B$, and $\overline{a} : \overline{A}$ representing the term a, then this equation ensures that the terms

$$\mathrm{app}(\overline{A}, [x:\overline{A}]\overline{B}, \lambda(\overline{A}, [x:\overline{A}]\overline{B}, [x:\overline{A}]\overline{b}), \overline{a}) = ([x:\overline{A}]\overline{b})\overline{a}$$

are convertible. The former shall be the term that represents $(\lambda x : A.b)a$, and the latter is convertible with

$$[\overline{a}/x]\overline{b}$$

which represents the term $[a/x]b$.

We hope again that, if we have constructed our signature correctly, we are able to establish a correspondence between the object theory S and its representation in the logical framework F of the following form:

- To each type A of S, there corresponds a term $\overline{A}$ of kind **Type**, and every term of kind **Type** is convertible with such a term. Further, if $\overline{A}$ and $\overline{B}$ are convertible in the framework (under the equations declared in the signature), then A and B are convertible in S.
- To each term M of type A in S, there corresponds a term $\overline{M}$ of kind $\overline{A}$; every term of kind $\overline{A}$ is convertible with such a term. Further, if $\overline{M}$ and $\overline{N}$ are convertible in the framework, then M and N are convertible in S.

Once again, the *adequacy theorem* for the representation of S consists of a formal statement of this correspondence. If Σ is the signature representing S, the adequacy theorem shall typically read as follows:

- Σ sig is derivable.
- The judgement

  $$x_1 : A_1, \ldots, x_n : A_n \vdash A \text{ type}$$

  is derivable in S if and only if

  $$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma \overline{A} : \mathbf{Type}$$

  is derivable in F.
- If

  $$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma M : \mathbf{Type}$$

  is derivable in F, then there exists a type A such that

  $$x_1 : A_1, \ldots, x_n : A_n \vdash A \text{ type}$$

  is derivable in S, and

  $$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma M = \overline{A} : \mathbf{Type}$$

  is derivable in F.

- The judgement
$$x_1 : A_1, \ldots, x_n : A_n \vdash A = B$$
is derivable in S if and only if
$$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma \overline{A} = \overline{B} : \mathbf{Type}$$
is derivable in F.

- The judgement
$$x_1 : A_1, \ldots, x_n : A_n \vdash a : A$$
is derivable in S if and only if
$$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma \overline{a} : \overline{A}$$
is derivable in F.

- If
$$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma M : \overline{A}$$
is derivable in F, then there is a term a such that
$$x_1 : A_1, \ldots, x_n : A_n \vdash_\Sigma a : A$$
is derivable in S, and
$$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma M = \overline{a} : \overline{A}$$
is derivable in F.

- The judgement
$$x_1 : A_1, \ldots, x_n : A_n \vdash a = b : A$$
is derivable in S if and only if
$$x_1 : \overline{A_1}, \ldots, x_n : \overline{A_n} \vdash_\Sigma \overline{a} = \overline{b} : \overline{A}$$
is derivable in F.

What we have here been calling the *signature* goes under different names in different frameworks. In the AUTOMATH systems, for example, these sequences of declarations were called the *book*; and in the literature on PAL$^+$, they are called *specifications* (of type theories). The term 'signature' comes from the ELF literature. For the purposes of this thesis, we shall use the term 'signature' whichever framework we are dealing with.

It is worthy of note that the role of these signatures has changed dramatically since the days of AUTOMATH. The intention was that AUTOMATH would be used to verify completed, prepared formal texts; these texts were the books, and consisted not merely of a specification of the system of logic to be used, but also definitions, representing the definitions made and theorems proved in the text. (The formula ϕ being proved as a theorem would be represented by defining a constant of type $\mathbf{Prf}[\overline{\phi}]$.) Thus, the judgements Σ sig — that the book Σ is 'OK' in AUTOMATH's terminology — were the only judgements of importance in the AUTOMATH systems. In a modern framework, these judgements are just the starting point; it is the typing judgements under a valid signature with which we are most concerned.

It is a striking fact that the same judgements can play these two very different roles. Any of the AUTOMATH systems could be used in the modern style without any modification, and any modern framework could be used (with greater or lesser efficiency) as a book-checker. These are two different ways of *using* a logical framework, not two different classes of framework.

1.3 The Case for Weaker Frameworks

As far back as 1991, de Bruijn [deB91] was making the case for logical frameworks to be weak type systems. Of the arguments he makes, the following two are relevant for our purposes:

1. It is not desirable to have a large number of incompatible logical frameworks in existence. Better to have a few simple systems than a large number of complex systems, each with a different set of features.

2. If we restrict the framework to only what cannot be included in the signature (book, in de Bruijn's terminology), then the distinction between framework and object theory is more clearly maintained.

To these, we may add that theoretical work is almost always easier when working with a weaker framework than a stronger. As one example, we have already stated that adequacy theorems generally become harder to prove as the framework grows in strength.

We have seen that the logical framework community did not exactly rally to de Bruijn's call. Logical frameworks have continued to multiply and become stronger and more specialised. Nevertheless, the spirit of de Bruijn's paper can arguably be seen to some extent in the lambda-free framework PAL$^+$ designed by Luo [Luo03], which does not have a primitive operation of lambda-abstraction, instead being built on the more basic operations of parameterisation and definition.

There are many good reasons for developing stronger frameworks. It may be that a particular specialised class of object theories is easier to represent if we make certain additions to the framework (this is the motivation behind the linear logical framework LLF [Cer96] and the concurrent logical framework CLF [WCPW03]). It may be that it is felt certain features of an object theory are better handled by making them part of the framework, rather than attempting to build them using the framework's current mechanisms; for example, in [Luo99] it is argued that coercive subtyping is better handled in this way. It may be that a particular way of strengthening the framework makes a particular implementation easier to use, by automating a task that previously had to be performed by the user.

So we argue for a more liberal approach than de Bruijn's. There is no reason that strong logical frameworks should not be developed, if they should prove useful. Nevertheless, we claim it is desirable to develop simultaneously the weak 'core' framework, a conservative subsystem of the strong framework that contains only those features that are essential to the representation of object theories, as opposed to those features that are desirable for implementation, but inessential theoretically.

We shall see that there is one major advantage to working with these weak frameworks: the representation of the object theory in a weak framework is much closer to the theory itself than is the case in a strong framework. For example, each entity of the object theory is represented by a term in the framework that is unique up to identity (α-convertibility), as opposed to convertibility in the framework. That is, we are able to show something stronger than the correspondence that we claimed in 1.2 above should hold. For a deductive system, we can show:

- To each syntactic class C of S, there corresponds a term $\overline{C}$ of kind **Type**. Further, $\overline{C}$ and $\overline{C'}$ are *identical* if and only if C and C' are the same syntactic class.
- To each entity M of syntactic class C, there corresponds a term $\overline{M}$ of kind $\overline{C}$; every term of kind $\overline{C}$ is of this form. Further, $\overline{M}$ and $\overline{M'}$ are *identical* if and only if $M \equiv M'$.
- To each derivation Δ of the formula ϕ, there corresponds a term $\overline{\Delta}$ of kind $\mathbf{Prf}[\overline{\phi}]$; every term of kind $\mathbf{Prf}[\overline{\phi}]$ is of this form. Further, $\overline{\Delta}$ and $\overline{\Delta'}$ are *identical* if and only if Δ and Δ' are the same derivation.

Further, the rules of deduction under the appropriate signature mirror the grammar clauses and rules of deduction of the object theory almost exactly. In general, each application of a rule of deduction in the object theory corresponds to the application of a single rule of deduction in the logical framework. This is very far from being the case with most frameworks.

As an example, consider the representation of predicate logic in ELF described in [HHP93]. Suppose we have proved $\forall x \phi$; i.e. we have constructed a term D such that

$$\Gamma \vdash_\Sigma D : \mathbf{Prf}(\forall(\lambda x : \iota . \overline{\phi})) \tag{1.1}$$

and we also know ϕ is a well-formed formula and t is a term:

$$\begin{aligned} \Gamma, x : \iota \quad &\vdash_\Sigma \quad \overline{\phi} : o &(1.2)\\ \Gamma \quad &\vdash_\Sigma \quad \overline{t} : \iota &(1.3) \end{aligned}$$

(ι being the constant representing the syntactic class of terms, and o that of formulas). We wish to apply the universal elimination rule to conclude $[t/x]\phi$; i.e.

$$\Gamma \vdash_\Sigma \mathtt{ALL\text{-}E}(\lambda x : \iota . \overline{\phi})(\overline{t})D : \mathbf{Prf}([\overline{t}/x]\overline{\phi}) \,. \tag{1.4}$$

Doing so in ELF requires going through at least four intermediate judgements. In our weak frameworks, by way of contrast, the derivation of (1.4) from (1.1), (1.2) and (1.3) shall be the application of a single primitive rule of deduction of the framework.

We claim that this answers de Bruijn's second concern above to a large extent. The separation of framework and object theory is achieved by separately considering the strong and weak frames. The weak framework contains nothing but what is essential for defining the representation of the object theory, and the result is an extremely faithful representation of the same. All the various features that are desirable for implementation are present in the strong frame; and, from the fact that the strong frame is a conservative extension of the weak, we know that these features do not interfere with the adequacy of the object theory's representation.

De Bruijn's first argument is also partially answered by the following remarkable fact: the same weak frameworks can serve as the core of many different strong frameworks. Thus, the system of weak frameworks that we shall introduce provide some structure to the otherwise chaotic world of logical frameworks. They provide a means of comparing the strength of two different frameworks, for example, by considering which of our weak frameworks can be embedded in each.

In detail, we consider the following elements to be essential to a framework:

- The representation of the terms and types of the object theory as first-class objects.
- The ability to declare constants with parameters (to represent the term and type constructors of the object theory).
- The ability to declare (parametrised) equations to hold between terms or types of the object theory.

However, we consider the following features normally found in frameworks to be inessential, and we seek to remove them:

- The representation of families of terms or types of the object theory as first-class objects.
- The ability to declare equations to hold between families of terms and types.

These, and other inessentials such as local definitions, subtyping relations, and so forth, can always be added later.

We achieve this simplification by only ever working with *normal forms*; wherever a term not in normal form would normally appear, we write its normal form instead. In most type theories, this would be a disastrous suggestion; even if we can be assured normal forms always exist, they grow in length very rapidly, and the terms would quickly become impractically large. In a logical framework, however, the length of a normal form is proportional to the length of the expression of the object theory which it represents. We may always introduce other forms of expression (such as lambda-abstractions or let-definitions) later as abbreviational mechanisms.

We are thus to present infinitely many *lambda-free* logical frameworks; systems that do not contain lambda-abstraction (nor application, nor product kinds) as a primitive operation. The phrase "lambda-free logical framework" was coined by Luo to describe PAL^+. In the paper [Luo03], he writes:

> A lambda-free logical framework takes parameterisation and definitions as the basic notions to provide schematic mechanisms for specification of type theories and their use in practice.

	No equation declarations	Equation declarations
Parameters of small type only	$\mathbf{SPar}\,(n)^-$	$\mathbf{SPar}\,(n)$
Parameters of small type and large kind	$\mathbf{LPar}\,(n)^-$	$\mathbf{LPar}\,(n)$

Table 1.1: The Families of Weak Frameworks

We are taking this approach further: we take parameterisation alone as basic. Definitions can be added to our weak frameworks, but they are not essential (as opposed to PAL$^+$, where definitions are required to play the role of parametric abstraction; see Section 2.6 in [Luo03]).

These weak frameworks are organised into four infinite families:

$$\mathbf{SPar}\,(n)^-\,, \qquad \mathbf{SPar}\,(n)\,, \qquad \mathbf{LPar}\,(n)^-\,, \qquad \mathbf{LPar}\,(n)$$

where n is either a natural number, or ω.

We have stated that we still include the ability to declare constants with parameters. In the systems $\mathbf{SPar}\,(n)^-$ and $\mathbf{SPar}\,(n)$, these parameters may only represent terms of small type, that is, terms whose kinds are themselves of kind **Type**. In the systems $\mathbf{LPar}\,(n)^-$ and $\mathbf{LPar}\,(n)$, these parameters may also represent terms of large kind; that is, terms whose kind is **Type** itself.

Likewise, we have stated that we still include the ability to declare equations. In fact, only in the systems $\mathbf{SPar}\,(n)$ and $\mathbf{LPar}\,(n)$ is this ability still present; in the systems $\mathbf{SPar}\,(n)^-$ and $\mathbf{LPar}\,(n)^-$, we have removed it.

The ordinal n indicates the *order* of parametrisation permitted. We allow our parameters to take parameters themselves. The *0th-order*, or, as we shall say, *base* constants and variables are those that do not take any parameters. The *1st-order* constants and variables take 0th-order (base) variables as parameters; that is, their parameters represent arbitrary terms, and do not themselves take parameters. The *2nd-order* constants and variables take 1st-order variables as parameters; that is, their parameters themselves take parameters, which are base variables. Thus, a 2nd-order constant's parameters represent arbitrary *families* of terms, indexed by base variables. The 3rd-order constants and variables take 2nd-order variables as parameters, and so forth.

In the systems $\mathbf{SPar}\,(n)^-$, $\mathbf{SPar}\,(n)$, $\mathbf{LPar}\,(n)^-$ and $\mathbf{LPar}\,(n)$ for *finite* n, the integer n indicates the highest level of parameterisation permitted; that is, no constant of order higher than n may appear in the signature. We restrict the variables even further; no variable of higher order than $n-1$ may be used, as it so happens that there is no use for nth-order variables if there are no $n+1$st-order constants. In the systems with $n=\omega$, constants and variables of all orders may appear.

The vast majority of logical frameworks allow parameterisation of all orders. This, we shall see, is a case of overkill: most of our object theories shall be representable in systems with only finite order of parameterisation. In fact, it shall prove very rare that we shall have to go beyond 2nd-order parameterisation. The W-types of Martin-Löf Type Theory require 3rd-order parameterisation, as does the inductive definition of Π-types (with eliminator **funsplit** as opposed to application) if it is used. The only time we shall need to go beyond 3rd order is for Schroeder-Heister's extension of natural deduction, which requires parameterisation of all orders to represent its rules of deduction of differing levels; while we do not consider it in this thesis, the system UTT would also require all orders of parameterisation, as would Dybjer's scheme for inductive types, as either allows the definition of inductive types for inductive schemata of all orders.

In general, the systems $\mathbf{SPar}\,(n)^-$ shall be sufficient to represent deductive systems, as these do not require equations to be declared, and a parameter of large kind in this case would represent an 'arbitrary syntactic class', which does not make sense. The systems $\mathbf{LPar}\,(n)$ are needed to represent type theories, for these do require equations to be declared to represent computation rules, and they do require constants with parameters of large kind to represent those constructors that take types as arguments.

The only example I have found of an object theory that is between these two extremes is higher-order logic, HOL [Chu40, And71, Sch77, Tak75]. HOL can be represented in $\mathbf{LPar}\,(2)^-$ if we wish the types to be represented by terms of kind **Type**. This is quite fitting, as HOL could well be seen as a borderline case between a deductive system and a type theory. (We could also represent HOL in $\mathbf{SPar}\,(2)^-$ as in [HHP93] if we treat the types as just another syntactic class.)

1.4 Outline

In very broad outline, the development in this thesis proceeds as follows:

In Part I, we present our infinite collection of weak frameworks. We show that they extend one another conservatively, and how some of them can be embedded in several existing frameworks: specifically, $\mathbf{LPar}\,(1)^-$ can be embedded in the system PAL and $\mathbf{SPar}\,(\omega)^-$ in AUT-68, both members of the AUTOMATH family; $\mathbf{SPar}\,(\omega)^-$ can also be embedded in the Edinburgh Logical Framework, ELF; and $\mathbf{LPar}\,(\omega)$ can be embedded in PAL$^+$. Thus, the scheme of inclusions is as shown in Figure 1.1. We build, and prove adequate, representations of several object theories in the frameworks low down on the hierarchy: specifically, we build propositional logic in $\mathbf{SPar}\,(1)^-$, predicate logic in $\mathbf{SPar}\,(2)^-$, Schroeder-Heister's extension of natural deduction in $\mathbf{SPar}\,(\omega)^-$, Martin-Löf Type Theory in $\mathbf{LPar}\,(3)$, and the Calculus of Constructions in $\mathbf{LPar}\,(2)$. The adequacy theorems for those frameworks in which we have embedded our weak frames follow immediately as corollaries.

In Part II, we revisit the results of Part I from a "bottom-up" perspective. We observe that each inclusion $F_1 \hookrightarrow F_2$ that we established in Part I can be read as "F_2 is the system F_1 with some extra features added". In the case of the weak frameworks, these features are higher orders of parameterisation, the presence of parameters of large kind, and the ability to declare

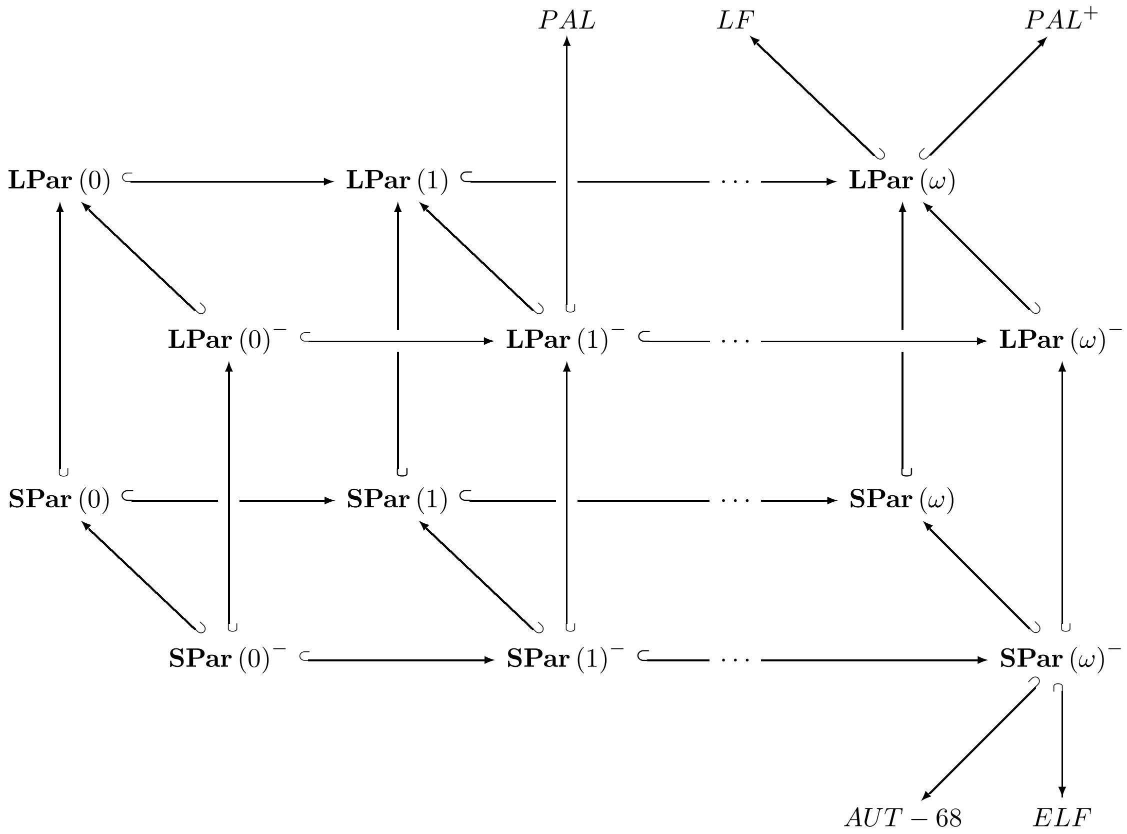

Figure 1.1: The Hierarchy of Logical Frameworks in Part I

equations. In the case that F_2 is an existing framework, these features may be local definitions, or the ability to consider lambda-abstractions as first-class objects.

We investigate whether it would have been possible to define the frameworks in this manner from the beginning, by defining *features* in a framework-independent manner, then building a framework by specifying a set of features to be added to a *basic framework*, **BF** (which is roughly the same as the system $\mathbf{SPar}\,(0)^{-}$). We define features as sets of grammatical constructors and rules of deduction to be added to a framework, in such a way that the result is a conservative extension of the original framework. In Part II, we reconstruct all of the weak frameworks we introduced in Part I in just this manner, as well as two of the existing frameworks we considered: ELF and LF.

In more detail, our plan for this thesis is as follows:

Chapter 2 — Natural Deduction We begin by discussing several of the different ways of presenting natural deduction; three of the systems we shall use as object theories are natural deduction systems, and it is worth treating these matters in a separate chapter rather than interrupting the exposition later. We stated above that none of the usual presentations of natural deduction is exactly captured by the representation of natural deduction in a logical framework. In particular, we would like it to be the case that, for derivations Δ and Δ', $\overline{\Delta}$ and $\overline{\Delta'}$ are convertible iff Δ and Δ' are the same derivation. This is not the case for any obvious definition of 'the same derivation'.

In this chapter, we give a development of natural deduction that is closer to the behaviour of a representation of a natural deduction system in a logical framework. The formulation we give is very similar to the treatment in [HHP93]. Rather than defining derivations as trees of formulas, we define a set of *proof expressions*, with *occurrence markers* to label the places where open hypotheses are used. These occurrence markers can be bound within proof expressions similarly to variables; binding of occurrence markers corresponds to discharge of hypotheses.

We examine in quite some detail the correspondence between our presentation of natural deduction and one traditional account, that in which derivations are trees of formulas with discharge formulas; and we describe in less detail the correspondence with two others, one in which the derivations are trees without discharge formulas, and one in which we simply define inductively the derivability relation without giving a set of concrete objects to be the derivations.

Chapter 3 — The Type Framework TF In this chapter, we describe in detail one of our weak frameworks, the *type framework* TF; this is the strongest of our weak frameworks, to which we gave the name $\mathbf{LPar}\,(\omega)$ above. It allows the declaration of constants with parameters of large kind of any order of parameterisation, and the declaration of equations. We describe its syntax, explaining the intuition behind the various expressions and operations we introduce. We illustrate these concepts by building up, as we proceed, a representation of an object theory: the noble fragment of intuitionistic logic (i.e. the part concerned solely with implication and universal quantification). We proceed to develop the basic metatheory of TF, culminating in the proof that the representation of this object theory is adequate.

Chapter 4 — Object Logics in Weak Frameworks We proceed to build representations of five more object theories within TF: a Hilbert-style presentation of classical propositional logic, a natural deduction presentation of intuitionistic predicate logic, Schroeder-Heister's extension of natural deduction, Martin-Löf Type Theory, and the Calculus of Constructions. In each case, we prove the adequacy theorem for the representation; thanks to the close correspondence between the object theory and the representation in TF, none of these theorems is difficult to prove.

Chapter 5 — Subsystems of TF We introduce the rest of our weak logical frameworks: all of the systems $\mathbf{SPar}\,(n)^-$, $\mathbf{SPar}\,(n)$, $\mathbf{LPar}\,(n)^-$ and $\mathbf{LPar}\,(n)$. We define them as subsystems of TF, and prove that they extend one another conservatively. The proof of this result is not difficult, for the conservativity holds on the level of derivations, not just judgements: if S_1 is a subsystem of S_2, any derivation in S_2 of a judgement expressible in S_1 must be a valid derivation in S_1.

We also introduce a variant S_c of each of these frameworks S, in which the bound variables within abstractions are typed. We prove the correspondence between S and S_c in this chapter. This shall prove highly convenient whenever we embed one of these systems in an existing framework F in Chapter 6; it shall often prove easiest to define a mapping from S_c into F, but to define the inverse mapping from F to S.

Chapter 6 — Existing Logical Frameworks We proceed to embed several of our weak frameworks within existing logical frameworks. Specifically, we show how $\mathbf{SPar}\,(\omega)^-$ can be embedded as a conservative subsystem in both ELF and AUT-68, $\mathbf{LPar}\,(1)^-$ in PAL, and $\mathbf{LPar}\,(\omega)^-$ in PAL$^+$.

Once we have proven these results, we are able to obtain adequacy theorems for all the object theories with which we dealt in Chapter 4 immediately as corollaries. This is perhaps one of the clearest advantages to working with weak logical frameworks — adequacy theorems in weak frameworks are almost trivial to prove, and adequacy theorems for any frame in which the weak framework can be embedded follow as corollaries. The complexity present in usual adequacy theorems is, of course, still present — it manifests now in the proof that the weak framework can be conservatively embedded in the stronger.

Chapter 7 — Features This chapter begins Part II of the thesis, in which we begin building our frameworks from scratch (or, at least, from **BF**) by means of features. A *feature* consists of a set of grammatical constructors and rules of deduction that can be added to a framework. We define the following features in this chapter:

- $\mathbf{SDec}\,(n)$ $(n = 1, 2, \ldots)$ — allows the declaration of nth-order constants with parameters of small kind.
- $\mathbf{LDec}\,(n)$ $(n = 0, 1, 2, \ldots)$ — allows the declaration of nth-order constants with parameters of large kind.
- $\mathbf{S}\lambda^{\beta}\,(n)$ $(n = 1, 2, \ldots)$ — introduces nth-order lambda-abstractions, with domains of small kind, identified up to β-convertibility.

- $\mathbf{S}\lambda^{\beta\eta}(n)$ $(n = 1, 2, \ldots)$ — introduces nth-order lambda-abstractions, with domains of small kind, identified up to $\beta\eta$-convertibility.
- $\mathbf{L}\lambda^{\beta}(n)$ $(n = 1, 2, \ldots)$ — introduces nth-order lambda-abstractions, with domains of large kind, identified up to β-convertibility.
- $\mathbf{L}\lambda^{\beta\eta}(n)$ $(n = 1, 2, \ldots)$ — introduces nth-order lambda-abstractions, with domains of large kind, identified up to $\beta\eta$-convertibility.
- $\mathbf{SEq}(m, n)$ $(m, n = 0, 1, 2, \ldots)$ — allows the declaration of equations between the mth-order objects in small kinds, parameterised by an nth-order list of parameters.
- $\mathbf{LEq}(m, n)$ $(m, n = 0, 1, 2, \ldots)$ — allows the declaration of equations between the mth-order objects in large kinds, parameterised by an nth-order list of parameters.

These features *depend* on one another; that is, we insist that certain features be present in a framework before others can be introduced. For example, $\mathbf{LPar}(n)$ depends on $\mathbf{SPar}(n)$, as well as $\mathbf{LPar}(n-1)$ if $n > 1$. These details are all summarised in Appendix A.

As we introduce each feature, we prove that adding that feature to a framework always results in a conservative extension of the framework. For most of the features, this is almost trivial to prove; for the features $\mathbf{X}\lambda^{R}(n)$, it follows a similar pattern to the conservativity results of Chapter 6. Each conservativity theorem in this chapter is simpler than the ones in Chapter 6, however, as we are now adding the lambda-abstractions one order at a time.

We are able to reconstruct the frameworks with which we dealt in Chapter 1 as follows:

$$
\begin{aligned}
\mathbf{SPar}(n)^{-} &= \mathbf{BF} + \mathbf{SDec}(1) + \mathbf{SDec}(2) + \cdots + \mathbf{SDec}(n) \\
\mathbf{SPar}(\omega) &= \mathbf{BF} + \mathbf{SDec}(1) + \mathbf{SDec}(2) + \cdots \\
\mathbf{SPar}(n) &= \mathbf{BF} + \mathbf{SEq}(0,0) + \mathbf{SDec}(1) + \mathbf{SEq}(0,1) + \cdots + \mathbf{SDec}(n) + \mathbf{SEq}(0,n) \\
\mathbf{SPar}(\omega) &= \mathbf{BF} + \mathbf{SEq}(0,0) + \mathbf{SDec}(1) + \mathbf{SEq}(0,1) + \cdots \\
\mathbf{LPar}(n)^{-} &= \mathbf{BF} + \mathbf{LDec}(0) + \mathbf{SDec}(1) + \mathbf{LDec}(1) + \cdots + \mathbf{SDec}(n) + \mathbf{LDec}(n) \\
\mathbf{LPar}(\omega)^{-} &= \mathbf{BF} + \mathbf{LDec}(0) + \mathbf{SDec}(1) + \mathbf{LDec}(1) + \cdots \\
\mathbf{LPar}(n) &= \mathbf{BF} + \mathbf{LDec}(0) + \mathbf{SEq}(0,0) + \mathbf{LEq}(0,0) \\
&\quad + \mathbf{SDec}(1) + \mathbf{LDec}(1) + \mathbf{SEq}(0,1) + \mathbf{LEq}(0,1) \\
&\quad + \cdots \\
&\quad + \mathbf{SDec}(n) + \mathbf{LDec}(n) + \mathbf{SEq}(0,n) + \mathbf{LEq}(0,n) \\
\mathbf{LPar}(\omega) &= \mathbf{BF} + \mathbf{LDec}(0) + \mathbf{SEq}(0,0) + \mathbf{LEq}(0,0) \\
&\quad + \mathbf{SDec}(1) + \mathbf{LDec}(1) + \mathbf{SEq}(0,1) + \mathbf{LEq}(0,1) \\
&\quad + \cdots
\end{aligned}
$$

Chapter 8 — Existing Logical Frameworks Revisited Finally, we reconstruct one of the existing frameworks with which we dealt in Chapter 6, namely ELF, as well as Martin-Löf's Logical Framework LF. We show that LF is equivalent to the framework obtained by adding all the features **LDec** (n), **LEq** (m,n) and $\mathbf{L}\lambda^{\beta\eta}(n)$ for all m and n, as well as the features on which they depend, to **BF**.

For ELF, we need to define one new feature, **FKnd**, which introduces families of types as first-class objects; ELF is equivalent to the result of adding all the features **SDec** (n), **SEq** (m,n), $\mathbf{S}\lambda^{\beta}(n)$ and **FKnd**, as well as all the features on which they depend, to **BF**.

1.5 Limitations

There are several features that may vary from one framework to another whose consideration lies outside the scope of this thesis. Firstly, as already mentioned, we shall always take **Type** to be a universe à la Russell. This is a fairly minor change, since logical frameworks, considered as typing systems, are fairly simple; it is extremely rare for one to use impredicativity or reflection, for example.

Secondly, we shall only deal with frameworks in which lambda-abstractions are typed. Thus, for example, we do not deal with Martin-Löf's logical framework as it was originally presented, but with the typed version developed by Luo [Luo94]. This is most certainly not a minor change, for the untyped version no longer has the uniqueness of kinds property; $(x)x$, for example, is a member of every kind of the form $(A)A$. We mention here that the proof in Chapter 6 can be straightforwardly modified to a proof that **LPar** (ω) can be embedded as a conservative subsystem in the untyped version of LF; reconstructing this version of LF in terms of features — that is, defining a feature that permits untyped lambda abstraction — is a very difficult task, however, thanks to the absence of this uniqueness of kinds.

Lastly, we deal only with frameworks in which there is a judgement form for equality (convertibility), rather than one which uses an external relation, such as β-convertibility, for this task. This is a deviation from the customary presentation of ELF and the original presentation of the AUTOMATH system. The two versions of each system are equivalent, but this is by no means an obvious fact; indeed, this question has been an important open problem in type theory for some time, and our solution should be of considerable independent interest. We present the proof in Appendix B. We give it in full detail for ELF, and then merely indicate which changes should be made for the two AUTOMATH systems we consider.

1.6 Conclusion

We see the main contributions of this thesis to be as follows.

Firstly, there is the intrinsic interest in the weak frameworks themselves. They are frames that provide closer representations of object theories than is usually the case for a logical framework. While their grammar is more complicated than a traditional framework, their deductive structure is simpler; so, after the initial grammatical lemmas have been proven, their metatheory is quite pleasing and, in many ways, simpler than for most typing systems. In particular, adequacy theorems are much easier to establish than for most frameworks.

Secondly, the method we present should prove useful for the analysis of existing frameworks. It might be worthwhile, in establishing the theoretical properties of a system, to extract a lambda-free core and work within that system; many results will carry over by the conservativity result. Likewise, when comparing two or more frameworks, it may be worthwhile to study the lambda-free subsystem common to both. These weak systems thus bind the world of logical frameworks together into a common hierarchy. It is particularly striking how adequacy theorems for an object theory in many different frameworks follow as corollaries from the adequacy theorem in a common weak subsystem.

Thirdly, there may be implications for the design of frameworks. It remains to be seen whether features can be defined in a truly framework-independent manner, as we take some steps towards in Part II. The utopian ideal would be a situation in which all research in the field of logical frameworks was carried out in terms of framework-independent features, whose properties could be studied independent of the field in which they are present, so that all researchers on whatever system could use one another's results. We shall say more about the possibility of such a general theory of frameworks in the concluding chapter, Chapter 9.

Immediate applications to the implementation of frameworks are not so apparent. It seems unlikely that it would be worthwhile implementing a weak framework directly; the lack of definiton and abbreviation mechanisms would probably make such an implementation impractical to use. Nevertheless, it might be interesting to develop a system with lambda-abstractions, but with a finite bound on the order of parameterisation permitted. However, we see the work in this thesis overall as being of more theoretical than practical interest.

Part I

Weak Logical Frameworks

Chapter 2

Natural Deduction

Before introducing our weak logical frameworks, we shall begin with a discussion of natural deduction, as many of our examples of object theories shall be natural deduction systems.

We would like to investigate how well various frameworks capture natural deduction: how close the representation of a natural deduction system in a framework is to the system itself. Unfortunately, no derivations-as-objects framework very satisfactorily represents the derivations of natural deduction as they are usually defined. We are going to present here a somewhat unusual formulation of natural deduction, but one which matches more closely the behaviour of such systems in logical frameworks than does any of the traditional accounts.

Our presentation is very similar to that given in Section 4.1 of [HHP93], in which a grammar of *proof expressions* is developed, containing free and bound occurrences of *occurrence markers*; we shall borrow this terminology. In the presentation we are to give, we avoid the use of proof contexts by using a different set of occurrence markers for each formula; this may be considered a relatively minor change. Nevertheless, it is worth going into some detail, as the authors of [HHP93] do not investigate the correspondence between proof expressions and the derivations of a traditional presentation of natural deduction.

2.1 Traditional Accounts of Natural Deduction

There are several differing ways in which natural deduction is presented in the literature of formal logic. We shall deal here three of them:

1. Derivations are defined as trees of formulas with a *discharge function* mapping some of the leaves (hypotheses) to nodes below them, each hypothesis being mapped to the node at which it is discharged.

2. We define inductively which trees of formulas are "derivations of ψ depending on assumptions $\phi_1, \ldots, \phi_n$", without any explicit designation of where each temporary hypothesis is discharged.

3. We simply define inductively the relation

$$\Gamma \vdash \phi$$

between sets Γ of formulas and formulas ϕ, without giving any set of concrete objects to be the derivations.

We shall not give a large number of references to texts in which accounts of natural deduction are developed. We shall content ourselves with mentioning Prawitz's classic [Pra65], which gives both accounts 1 and 2, and [Sun83], which contains a comparison of these last two presentations with each other and the sequent calculus.

2.1.1 Propositional Rules

How each rule of deduction is interpreted — in particular, what it means to discharge a hypothesis, and what it means for a variable to be 'fresh' — is subtly different in each of these accounts. Consider, for example, the rule that epitomizes natural deduction — the introduction rule for $\rightarrow$:

$$\frac{\begin{matrix}[\phi] \\ \vdots \\ \psi\end{matrix}}{\phi \rightarrow \psi}\,(\rightarrow I)$$

In account 1, this rule becomes the following clause in the inductive definition of 'derivation':

Suppose D is a derivation with conclusion (root) ψ. Let D' be the tree:

$$\frac{\begin{matrix}\vdots\, D \\ \psi\end{matrix}}{\phi \rightarrow \psi}$$

in which every hypothesis discharged in D is discharged at the corresponding point in D', and, in addition, zero or more occurrences of ϕ as hypotheses are discharged at the root of D'. Then D' is a derivation.

In account 2, the rule is read as follows:

Suppose D is a derivation of ψ depending on $\gamma_1, \ldots, \gamma_m$, and possibly ϕ. Then

$$\frac{\begin{matrix}\vdots\, D \\ \psi\end{matrix}}{\phi \rightarrow \psi}$$

is a derivation of ψ depending on $\gamma_1, \ldots, \gamma_m$ alone.

In account 3, it is read as:

If $\Gamma \cup \{\phi\} \vdash \psi$, then $\Gamma \vdash \phi \rightarrow \psi$.

Each approach has its advantages and disadvantages. The third is simpler technically than the second, which in turn is simpler than the first; however, the first two provide concrete objects (derivations) on which such operations as normalization can be defined, with the first's derivations providing more information than the second's.

A derivations-as-primitive framework can arguably capture the third of these formulations quite well. None of the three is exactly captured when natural deduction is formalized in a derivations-as-objects logical framework, however.

Example: Predicate Logic in ELF Let us consider how well any of the three formulations corresponds to the representation of predicate logic within ELF given in [HHP93], the details of which are reproduced in Appendix C.

The third, the inductive definition of $\vdash$, certainly does not, there is no set of objects to which the proof terms of the framework can correspond. Nor does the second, for the proof terms in the framework *do* record where discharges take place. For example, different proof terms correspond to the derivation

$$\dfrac{\dfrac{[p]}{p \to p}\,(1)}{p \to p \to p}\,(2)$$

depending where the discharge of the sole hypothesis took place. If ϕ was discharged at (1) the corresponding proof term is

$$\texttt{IMP-I}\ p\ (\supset\ \ p\ p)\ \lambda x : \mathit{true}(p).\texttt{IMP-I}\ p\ p\ \lambda y : \mathit{true}(p).x$$

while if it was discharged at (2) the proof term is

$$\texttt{IMP-I}\ p\ (\supset\ \ p\ p)\ \lambda x : \mathit{true}(p).\texttt{IMP-I}\ p\ p\ \lambda y : \mathit{true}(p).y\,.$$

Our first formulation fares the best, but there is still one problem. Undischarged hypotheses of ϕ are represented by free variables of type $\mathit{true}(\epsilon(\phi))$ (where $\epsilon(\phi)$ is the term corresponding to the formula ϕ). Undischarged hypotheses of the same formula may be assigned different, or the same, variable. When a rule, such as the introduction rule for $\to$ or the elimination rule for $\exists$, is used, it is *all and only* those hypotheses that have been assigned a particular variable p that are discharged. Thus, we must decide beforehand whether or not two hypotheses of the same formula are to be discharged together or not.

To make these matters clearer, consider the following derivation of $p \to q$ from $p \to p \to q$:

$$\dfrac{\dfrac{\dfrac{p \to p \to q \quad [p]}{p \to q}\,(\to E) \qquad [p]}{q}\,(\to E)}{p \to q}\,(\to I)$$

The initial part of this derivation:

$$\cfrac{\cfrac{p \to p \to q \quad p}{p \to q}\,(\to E) \qquad p}{q}\,(\to E)$$

can be represented in ELF by either of the proof terms

$$\begin{array}{c} p : o, q : o, x : \mathit{true}(\supset p\ (\supset p\ q)), y : \mathit{true}(p), z : \mathit{true}(p) \\ \vdash \texttt{IMP-E}\ p\ q\ (\texttt{IMP-E}\ p\ (\supset p\ q)\ x\ y)\ z : \mathit{true}(q) \end{array} \tag{2.1}$$

or

$$\begin{array}{c} p : o, q : o, x : \mathit{true}(\supset p\ (\supset p\ q)), y : \mathit{true}(p) \\ \vdash \texttt{IMP-E}\ p\ q\ (\texttt{IMP-E}\ p\ (\supset p\ q)\ x\ y)\ y : \mathit{true}(q)\,. \end{array} \tag{2.2}$$

Only in the latter case, (2.2), can we apply `IMP-I` to the result to obtain a term of type $\mathit{true}(\supset p\ q)$; the reason being that we are discharging two hypotheses of p at once, yet the constant `IMP-I` demands a term of type $\Pi x : \mathit{true}(p).\mathit{true}(q)$ in which only a single variable of type $\mathit{true}(p)$ may be bound. Thus, we need to have labelled both hypotheses with the same variable (y in the case of (2.2)).

This is never a problem in practice, of course, as almost all proof terms are constructed by backwards reasoning; and, even when using forwards reasoning, we can always relabel variables in (2.1) to obtain (2.2). Nevertheless, it is a disparity between the natural deduction system and its representation in the framework, which may well prove inconvenient when we wish to compare the way the same system is represented in two different frameworks.

2.1.2 Quantifier Rules

The situation is just as bad when we come to the rules for quantifiers. The so-called *eigenvariable* in several of these rules often comes with a long list of side-conditions attached, specifying in which formulas it may or may not occur free. We often abbreviate this by saying the eigenvariable must be 'fresh'. For example, consider the worst case, the elimination rule for the existential quantifier:

$$(\exists E)\ \cfrac{\exists x\phi \qquad \begin{array}{c}[[y/x]\phi] \\ \vdots \\ \psi\end{array}}{\psi}\ (y \text{ fresh})$$

Again, this rule is read in a different manner in each of the three accounts:

1. Suppose D_1 is a derivation with conclusion $\exists x\phi$, and D_2 is a derivation with conclusion ψ. Let y be a variables. Let D be the tree

$$\cfrac{\begin{array}{c}\vdots\, D_1 \\ \exists x\phi\end{array} \qquad \begin{array}{c}\vdots\, D_2 \\ \psi\end{array}}{\psi}$$

   in which every hypothesis discharged in D_1 or D_2 is discharged at the corresponding point of D, and in addition zero or more occurrences of $[y/x]\phi$ as hypotheses in D_2 are discharged at the root of D. Then, if y does not occur free in $\exists x\phi$, ψ nor any undischarged hypothesis of D_2, D is a derivation.

2. Suppose D_1 is a derivation of $\exists x\phi$ depending on $\gamma_1, \ldots, \gamma_m$, y is a variable, and D_2 is a derivation of ψ depending on $\delta_1, \ldots, \delta_n$ and possibly $[y/x]\phi$. If y does not occur free in $\exists x\phi$, ψ nor any of $\delta_1, \ldots, \delta_n$, then

$$\frac{\begin{array}{c}\vdots\, D_1 \\ \exists x\phi\end{array} \qquad \begin{array}{c}\vdots\, D_2 \\ \psi\end{array}}{\psi}$$

is a derivation of ψ depending on $\gamma_1, \ldots, \gamma_m, \delta_1, \ldots, \delta_n$.

3. If $\Gamma \vdash \exists x\phi$, y is a variable, $\Gamma \cup \{[y/x]\phi\} \vdash \psi$, and y is not free in $\exists x\phi$, ψ nor any member of Γ, then $\Gamma \vdash \psi$.

These side conditions on y are thus often of quite a complex form technically, and appear somewhat mysterious, although arguably they become more natural with experience. They are reflected in a logical framework by *binding* of the eigenvariables within the proof term.

Some light may be shed on the situation if we consider the use of a free variable in a derivation to be an *assumption* or *hypothesis*: namely, using x as a free variable can be seen as making the hypothesis "There exists an object, which we shall call x." In the rules $(\forall I)$ and $(\exists E)$, these hypotheses are *discharged.*

For example, the $(\exists E)$ rule can be read:

> From $\exists x\phi$, and the fact that, if we assume *there exists* an object y, and that $[y/x]\phi$, we can deduce ψ, conclude ψ.

The side conditions on the eigenvariable y become more natural if we consider that *both* assumptions "There exists an object y" and $[y/x]\phi$ are being discharged at the time the rule $(\exists E)$ is used. In account 1, for example, y must not occur free in $\exists x\phi$ because "y exists" is a hypothesis of D_2 only, not of D_1; the use of y free in $\exists x\phi$, which occurs in D_1, would indicate that the hypothesis "y exists" had been used in D_1. y must not occur free in ψ because the assumption "y exists" is discharged at the use of the $(\exists E)$ rule; the occurrence of y free in its conclusion, ψ, would indicate that the assumption is still open. For the same reason, y must not occur free in the open hypotheses of D_2, as these are still open after the $(\exists E)$ rule; the occurrence of y free in one of them would indicate that the hypothesis "y exists" was still open.

(By the same logic, y should not occur free in the open hypotheses of D_1. However, as it happens we can get away without this restriction; for we can simply "α-convert" D_2, replacing y with some other variable, z, say, throughout, which does not occur free in the open hypotheses of D_1. As y is not free in $\exists x\phi$, $[z/y][y/x]\phi \equiv [z/x]\phi$, so the result is a correct derivation of ψ from the open hypotheses of D_1 and D_2 less $[z/x]\phi$; these are the same as the open hypotheses of D_1 and D_2 less $[y/x]\phi$.)

The treatment of free variables on a par with open hypotheses is effectively what is done by the proof contexts (X, Γ) in [HHP93], which keep track both of which occurrence markers correspond to which formulas (in the set of declarations Γ), and which variables we are permitted to use (in the set of variables X) — we would now say, which variables we are using to represent objects we have assumed to exist. Thus, the proof contexts keep track of which hypotheses of both kinds we have made; hypotheses of either kind are removed from the context as they are discharged. However, these explicit proof contexts are not essential: we can simply look at which variables and occurrence markers occur free in the proof expression, for variables and occurrence markers are *bound* as the corresponding hypotheses are discharged.

This may also help explain an aspect of the predicate calculus often considered anomalous: that it seems to be possible to prove something exists. Consider the derivation

$$\dfrac{\dfrac{\forall x x = x}{x = x}}{\exists x x = x}$$

This is a correct derivation from $\forall x x = x$ (which is true in the empty domain) to $\exists x x = x$ (which is false). Ordinarily, we shrug our shoulders and work around this fact, by (for example) excluding empty interpretations from our semantics. Hodges [Hod01] gives a presentation of tableaux proof in which this anomaly is avoided by imposing restrictions on which terms can be used in the rule $(\forall E)$.

However, from our point of view, there is nothing mysterious happening, as an extra assumption has been made. In the second line, we make the assumption "There exists an object, which we shall call x." It is not at all surprising that $\exists x x = x$ can be deduced from such an assumption. (Hodges' restrictions on $(\forall E)$ can be seen as ensuring that we never make any such assumptions that have not been justified by a previous existential statement.)

Example: Predicate Logic in ELF The quantifier rules present the opposite problem to the one we faced when dealing with the connective rules. There, we found different proof expressions corresponding to the same derivation; now, we find different derivations corresponding to the same proof expression (up to α-convertibility). Consider the following two derivations from $\exists x P x$ and $\forall x(Px \to Q)$ to Q:

$$\dfrac{\exists x P x \qquad \dfrac{[Py] \qquad \dfrac{\forall x(Px \to Q)}{Py \to Q}\,(\forall E)}{Q}\,(\to E)}{Q}\,(\exists E) \qquad\qquad \dfrac{\exists x P x \qquad \dfrac{[Pz] \qquad \dfrac{\forall x(Px \to Q)}{Pz \to Q}\,(\forall E)}{Q}\,(\to E)}{Q}\,(\exists E)$$

The terms corresponding to these derivations are

$$\texttt{SOME-E}\ P\ Q\ p\ \lambda y : \iota.\lambda r : true(Py).\texttt{IMP-E}\ (Py)\ Q\ \texttt{ALL-E}\ (\lambda x : \iota. \supset\ (Px)\ Q)\ y\ q$$

and

$$\texttt{SOME-E}\ P\ Q\ p\ \lambda z : \iota.\lambda r : true(Pz).\texttt{IMP-E}\ (Pz)\ Q\ \texttt{ALL-E}\ (\lambda x : \iota. \supset\ (Px)\ Q)\ z\ q\,,$$

which are α-convertible. Thus, if we wish a faithful correspondence between derivations and terms, we must either abandon our convention that terms are identified up to α-conversion, or impose some equivalence relation on derivations, and declare that we shall represent derivations identified up to this equivalence relation.

It is this latter course that we shall follow. The equivalence relation, $\cong$, must be such that

$$\frac{\begin{array}{c}\vdots\, D\\ \phi\end{array}}{\forall x\phi}\,(\forall I) \qquad\cong\qquad \frac{\begin{array}{c}\vdots\, [y/x]D\\ [y/x]\phi\end{array}}{\forall y\phi}\,(\forall I)$$

and

$$\frac{\begin{array}{c}\\ \vdots\, D_1\\ \exists x\phi\end{array}\qquad\begin{array}{c}[[y/x]\phi]\\ \vdots\, D_2\\ \psi\end{array}}{\psi}\,(\exists E) \qquad\cong\qquad \frac{\begin{array}{c}\\ \vdots\, D_1\\ \exists x\phi\end{array}\qquad\begin{array}{c}[[z/x]\phi]\\ \vdots\, [z/y]D_2\\ \psi\end{array}}{\psi}\,(\exists E)$$

This is not too unnatural an identification to make, especially in light of our reading free variables as existential assumptions. In the latter case, for example, the two arguments are read:

$$\vdots \qquad (D_1)$$

So there exists an object x such that ϕ.

We may therefore assume that y is an object, and that $[y/x]\phi$.

$$\vdots \qquad (D_2)$$

Therefore, ψ.

and

$$\vdots \qquad (D_1)$$

So there exists an object x such that ϕ.

We may therefore assume that z is an object, and that $[z/x]\phi$.

$$\vdots \qquad ([z/y]D_2)$$

Therefore, ψ.

There is a very strong sense in which these two are the same argument.

2.2 Derivations as Proof Expressions

We shall attempt in this chapter to formulate a definition of a natural deduction system that corresponds more closely to the behaviour of such a system in a logical framework. We shall then be able to analyse the adequacy proof for the formulation of a natural deduction system in a given framework into two parts:

1. The traditional formulation of the natural deduction system is equivalent to our formulation.

2. Our formulation is adequately represented in the framework.

We shall provide the proof of 1 here; 2 shall be proven, for various frameworks, in Chapters 3 and 6.

Our formulation shall be an inductive definition of a family of sets: the set of *proof expressions* for ϕ for each formula ϕ. These proof expressions shall be linear grammatical expressions, pieces of syntax, but they shall correspond to the derivations of the traditional accounts; in fact, the traditional derivations shall very closely match the parse trees of our proof expressions.

We introduce a set of *occurrence markers* for each formula ϕ. These labels can occur either free or bound within a proof expression. Free occurrence markers correspond to open hypotheses, and bound labels to discharged hypotheses, in the traditional accounts. More accurately, suppose that, in the first traditional account, we insisted that each hyothesis be labelled, with two hypotheses getting the same label only if they are occurrences of the same formula. Further, suppose we insisted that, when a rule of deduction discharges hypotheses, it is all and only those hypotheses with a certain label p (chosen when the rule is used) that are discharged. Then these labels would correspond to the occurrence markers in our proof expressions. (Thus, and not at all coincidentally, occurrence markers mimic the behaviour of variables in logical frameworks.)

2.2.1 Predicate Logic

We now proceed to build our formulation of the natural deduction system of classical predicate logic. We begin with a very brief summary of the grammar of predicate logic (without equality).

A *first-order language* consists of a set of n-ary *function constants*[1] and a set of n-ary *predicate constants* for every natural number n. We also provide ourselves with an infinite set of *individual variables*.

The set of *terms* is defined thus:

- Every individual variable is a term.

- If f is an n-ary function constant, and t_1, …, t_n are terms, then $f(t_1, \ldots, t_n)$ is a term.

The set of *formulas* is defined thus:

- If R is an n-ary predicate constant, and t_1, …, t_n are terms, then $R(t_1, \ldots, t_n)$ is a formula.

- If ϕ is a formula, then $\neg\phi$ is a formula.

- If ϕ and ψ are formulas, then $\phi \to \psi$, $\phi \wedge \psi$, and $\phi \vee \psi$ are formulas.

[1] We are taking individual constants to be 0-ary function constants.

$$\text{(cont)}\ \frac{\begin{matrix}[\neg\phi] \\ \vdots \\ \psi\end{matrix} \qquad \begin{matrix}[\neg\phi] \\ \vdots \\ \neg\psi\end{matrix}}{\phi}$$

$$(\to I)\ \frac{\begin{matrix}[\phi] \\ \vdots \\ \psi\end{matrix}}{\phi \to \psi} \qquad (\to E)\ \frac{\phi \to \psi \quad \phi}{\psi}$$

$$(\wedge I)\ \frac{\phi \quad \psi}{\phi \wedge \psi} \quad (\wedge El)\ \frac{\phi \wedge \psi}{\phi} \quad (\wedge Er)\ \frac{\phi \wedge \psi}{\psi}$$

$$(\vee Il)\ \frac{\phi}{\phi \vee \psi} \quad (\vee Ir)\ \frac{\psi}{\phi \vee \psi} \quad (\vee E)\ \frac{\phi \vee \psi \qquad \begin{matrix}[\phi] \\ \vdots \\ \chi\end{matrix} \qquad \begin{matrix}[\psi] \\ \vdots \\ \chi\end{matrix}}{\chi}$$

$$(\forall I)\ \frac{\phi}{\forall x\phi}\,(x \text{ fresh}) \quad (\forall E)\ \frac{\forall x\phi}{[t/x]\phi}$$

$$(\exists I)\ \frac{[t/x]\phi}{\exists x\phi} \qquad (\exists E)\ \frac{\exists x\phi \qquad \begin{matrix}[\phi] \\ \vdots \\ \psi\end{matrix}}{\psi}\,(x \text{ fresh})$$

Figure 2.1: Natural Deduction Rules for Classical Predicate Logic

- If ϕ is a formula and x an individual variable, then $\forall x\phi$ and $\exists x\phi$ are formulas.

x is bound within ϕ in the formulas $\forall x\phi$ and $\exists x\phi$; we define free and bound variables, α-conversion, and substitution avoiding variable capture, in the usual manner. We identify formulas up to α-convertibility.

A *first-order theory* over a given language L is a set of sentences (formulas with no free variables), the *axioms*.

Fix a set of *occurrence markers for* ϕ for each formula ϕ. Let us also give a different name Ax_n to each axiom. Then, for each formula ϕ, the set of *proof expressions for* ϕ is given by the rules of deduction in Figure 2.1. Written out in full, the definition is as follows.

- If the axiom labelled Ax_n is the formula ϕ, then Ax_n is a proof expression for ϕ.
- Every occurrence marker p for ϕ is a proof expression for ϕ.
- If p is an occurrence marker for $\neg\phi$, D a proof expression for ψ and D' a proof expression for $\neg\psi$, then
$$\text{(cont)}(\phi, \psi, [p]D, [p]D')$$
is a proof expression for ϕ, where p is bound within both D and D'.

- If p is an occurrence marker for ϕ and D a proof expression for ψ, then

$$(\to I)(\phi, \psi, [p]D)$$

is a proof expression for $\phi \to \psi$, where p is bound within D.

- If D is a proof expression for $\phi \to \psi$ and D' a proof expression for ϕ, then

$$(\to E)(\phi, \psi, D, D')$$

is a proof expression for ψ.

- If D is a proof expression for ϕ and D' a proof expression for ψ, then

$$(\wedge I)(\phi, \psi, D, D')$$

is a proof expression for $\phi \wedge \psi$.

- If D is a proof expression for $\phi \wedge \psi$, then

$$(\wedge El)(\phi, \psi, D)$$

is a proof expression for ϕ.

- If D is a proof expression for $\phi \wedge \psi$, then

$$(\wedge Er)(\phi, \psi, D)$$

is a proof expression for ψ.

- If D is a proof expression for ϕ, then

$$(\vee Il)(\phi, \psi, D)$$

is a proof expression for $\phi \vee \psi$.

- If D is a proof expression for ψ, then

$$(\vee Ir)(\phi, \psi, D)$$

is a proof expression for $\phi \vee \psi$.

- If D is a proof expression for $\phi \vee \psi$, p an occurrence marker for ϕ, q an occurrence marker for ψ, and D_1 and D_2 proof expressions for χ, then

$$(\vee E)(\phi, \psi, \chi, D, [p]D_1, [q]D_2)$$

is a proof expression for χ, in which p is bound within D_1 and q bound within D_2.

- If D is a proof expression for ϕ, then

$$(\forall I)([x]\phi, [x]D)$$

is a proof expression for $\forall x\phi$, in which x is bound within both ϕ and D.

- If D is a proof expression for $\forall x\phi$, then

$$(\forall E)([x]\phi, t, D)$$

is a proof expression for $[t/x]\phi$, in which x is bound within ϕ.

- If D is a proof expression for $[t/x]\phi$, then

$$(\exists I)([x]\phi, t, D)$$

is a proof expression for $\exists x\phi$, in which x is bound within ϕ.

- If D is a proof expression for $\exists x\phi$, p an occurrence marker for ϕ, and D' a proof expression for ψ, then

$$(\exists E)([x]\phi, \psi, D, [x,p]D')$$

is a proof expression for ψ, in which x is bound within ϕ and D', and p is bound within D'.

Just as for terms, we identify proof expressions up to α-conversion, which in this case covers relabelling both of individual variables and occurrence markers.

The equivalence between this formulation and the traditional accounts of natural deduction can be proven as follows. We shall deal here only with the *noble fragment* of predicate logic: that is, the fragment concerned solely with $\forall$ and $\rightarrow$. The other connectives and quantifier are handled similarly.

Account One

We define the set of *derivations*, which are trees of formulas with discharge functions, as in Figure 2.2. As we have already discussed, we can only represent derivations up to an equivalence relation of α-convertibility. The inductive definition of this relation is given in figure 2.3. As we are identifying formulas up to α-convertibility, we can see that, if $\Delta \cong \Delta'$, then Δ and Δ' have the same open hypotheses and conclusion.

There are two further problems when converting a proof expression to a derivation. Firstly, we need to keep track of which open hypotheses correspond to which occurrence markers. Secondly, when we use the rule $(\forall E)$:

$$\frac{\forall x\phi}{[t/x]\phi}$$

the proof expression keeps track of which term t is used, whereas the derivation does not. If $x \notin \mathrm{FV}(\phi)$, then the use of two different terms s, t will lead to two different proof expressions, but the same tree of formulas.

- The one-node tree consisting solely of

$$\phi$$

is a derivation.

- If Δ is a derivation with conclusion ψ, then the tree Δ':

$$\dfrac{\begin{array}{c}\vdots\,\Delta\\ \psi\end{array}}{\phi \to \psi}$$

in which every hypothesis discharged in Δ is discharged at the corresponding point in Δ', and in addition zero or more occurrences of ϕ are discharged at the root, is a derivation.

- If Δ_1 is a derivation with conclusion $\phi \to \psi$, and Δ_2 is a derivation with conclusion ϕ, then the tree Δ':

$$\dfrac{\begin{array}{c}\vdots\,\Delta_1\\ \phi \to \psi\end{array} \qquad \begin{array}{c}\vdots\,\Delta_2\\ \phi\end{array}}{\psi}$$

in which every hypothesis discharged in Δ_1 or Δ_2 is discharged at the corresponding point in Δ', is a derivation.

- If Δ is a derivation with conclusion ϕ, and x does not occur free in any open hypothesis of Δ, then the tree Δ':

$$\dfrac{\begin{array}{c}\vdots\,\Delta\\ \phi\end{array}}{\forall x\phi}$$

in which every hypothesis discharged in Δ is discharged at the corresponding point in Δ', is a derivation.

- If Δ is a derivation with conclusion $\forall x\phi$, and t is any term, then the tree Δ':

$$\dfrac{\begin{array}{c}\vdots\,\Delta\\ \forall x\phi\end{array}}{[t/x]\phi}$$

in which every hypothesis discharged in D is discharged at the corresponding point in D', is a derivation.

Figure 2.2: Definition of Derivations in Natural Deduction

- For any derivation Δ, $\Delta \cong \Delta$.
- If $\Delta \cong \Delta'$, then $\Delta' \cong \Delta$.
- If $\Delta \cong \Delta'$ and $\Delta' \cong \Delta''$, then $\Delta \cong \Delta''$.
- If Δ and Δ' are derivations of ψ, and $\Delta \cong \Delta'$, then

$$\dfrac{\begin{array}{c}\vdots\,\Delta\\ \psi\end{array}}{\phi \to \psi} \quad\cong\quad \dfrac{\begin{array}{c}\vdots\,\Delta'\\ \psi\end{array}}{\phi \to \psi}$$

provided the corresponding occurrences of ϕ are discharged at the root in both cases.

- If Δ_1 and Δ_1' are derivations of $\phi \to \psi$, Δ_2 and Δ_2' are derivations of ϕ, $\Delta_1 \cong \Delta_1'$, and $\Delta_2 \cong \Delta_2'$, then

$$\dfrac{\begin{array}{c}\vdots\,\Delta_1\\ \phi \to \psi\end{array} \qquad \begin{array}{c}\vdots\,\Delta_2\\ \phi\end{array}}{\psi} \quad\cong\quad \dfrac{\begin{array}{c}\vdots\,\Delta_1'\\ \phi \to \psi\end{array} \qquad \begin{array}{c}\vdots\,\Delta_2'\\ \phi\end{array}}{\psi}$$

- If Δ and Δ' are derivations of ϕ, $\Delta \cong \Delta'$, x does not occur free in any open hypothesis of Δ or Δ', and y does not occur free anywhere in Δ', then

$$\dfrac{\begin{array}{c}\vdots\,\Delta\\ \phi\end{array}}{\forall x\phi} \quad\cong\quad \dfrac{\begin{array}{c}\vdots\,[y/x]\Delta'\\ [y/x]\phi\end{array}}{\forall y[y/x]\phi}$$

- If Δ and Δ' are derivations of $\forall x\phi$, $\Delta \cong \Delta'$, and t is any term, then

$$\dfrac{\begin{array}{c}\vdots\,\Delta\\ \forall x\phi\end{array}}{[t/x]\phi} \quad\cong\quad \dfrac{\begin{array}{c}\vdots\,\Delta'\\ \forall x\phi\end{array}}{[t/x]\phi}$$

Figure 2.3: Equivalence of Derivations in Natural Deduction

So we define:

Definition 2.1 (Assignment Function) *For a derivation* Δ*, an* assignment function *is a function assigning:*

- *To each occurrence of the formula* ϕ *as an open hypothesis, an occurrence marker for* ϕ*.*
- *To each point at which the rule* $\forall E$ *is used , a term* t *such that, if the formula immediately above that point is* $\forall x \phi$*, the formula at that point is* $[t/x]\phi$*. (There will be a choice as to what this term is if and only if* x *is not free in* ϕ*.)*

We extend the definition of $\cong$ to pairs $\langle \Delta, f \rangle$ consisting of a derivation with an assignment function. The definition is given in Figure 2.4. Note that, if $\langle \Delta, f \rangle \cong \langle \Delta', g \rangle$, then $\Delta \cong \Delta'$.

We are now ready to establish our correspondence between proof expressions and derivations. We shall define, for each proof expression D for ϕ, a pair $[\![D]\!] \equiv \langle \Delta, f \rangle$ consisting of a derivation Δ with conclusion ϕ, together with an assignment function f for Δ. Our translation $[\![\,]\!]$ will be defined only up to $\cong$; that is, $[\![D]\!]$ is strictly a $\cong$-equivalence class of derivation-assignment function pairs.

The definition is as follows:

- The occurrence marker p for ϕ is mapped to the one-node tree

  $$\phi$$

  in which the sole hypothesis is assigned the occurrence marker p.

- $(\to I)(\phi, \psi, [p]D)$ is mapped to the tree

  $$\dfrac{\begin{array}{c} \vdots\, [\![D]\!] \\ \psi \end{array}}{\phi \to \psi}$$

  in which every hypothesis discharged in $[\![D]\!]$ is discharged at the corresponding point, and in addiction every undischarged occurrence of ϕ as a hypothesis in $[\![D]\!]$ to which the occurrence marker p had been assigned is discharged at the root. Every hypothesis left undischarged is assigned the same occurrence marker as in $[\![D]\!]$, and every use of $(\forall E)$ in $[\![D]\!]$ is assigned the same term as it was in $[\![D]\!]$.

- $(\to E)(\phi, \psi, D_1, D_2)$ is mapped to the tree

  $$\dfrac{\begin{array}{c} \vdots\, [\![D_1]\!] \\ \phi \to \psi \end{array} \qquad \begin{array}{c} \vdots\, [\![D_2]\!] \\ \phi \end{array}}{\psi}$$

  in which every hypothesis discharged in $[\![D_1]\!]$ or $[\![D_2]\!]$ is discharged at the corresponding point, and the assignment function agrees with those of $[\![D_1]\!]$ and $[\![D_2]\!]$.

- For any $\langle \Delta, f\rangle$, $\langle \Delta, f\rangle \cong \langle \Delta, f\rangle$.
- If $\langle \Delta, f\rangle \cong \langle \Delta', g\rangle$, then $\langle \Delta', g\rangle \cong \langle \Delta, f\rangle$.
- If $\langle \Delta, f\rangle \cong \langle \Delta', g\rangle$ and $\langle \Delta', g\rangle \cong \langle \Delta'', h\rangle$, then $\langle \Delta, f\rangle \cong \langle \Delta'', h\rangle$.
- Suppose Δ_0 and Δ_0' are derivations of ψ, and $\langle \Delta_0, f\rangle \cong \langle \Delta_0', f'\rangle$. Let p be an occurrence marker for ϕ. Let

$$\Delta \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_0\\ \psi\end{array}}{\phi \to \psi} \qquad \Delta' \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_0'\\ \psi\end{array}}{\phi \to \psi}$$

  where the hypotheses discharged at the root are all and only those to which were assigned p by f or f', respectively. Let g and g' be the assignment functions on Δ and Δ' that agree with f and f' on those hypotheses that are still open and the uses of $(\forall E)$ in Δ_0 and Δ_0', respectively. Then $\langle \Delta, g\rangle \cong \langle \Delta', g'\rangle$.

- Suppose Δ_1 and Δ_1' are derivations of $\phi \to \psi$ with assignment functions f and f', Δ_2 and Δ_2' are derivations of ϕ with assignment functions g and g', $\langle \Delta_1, f\rangle \cong \langle \Delta_1', f'\rangle$, and $\langle \Delta_2, g\rangle \cong \langle \Delta_2', g'\rangle$. Let

$$\Delta \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_1\\ \phi \to \psi\end{array} \qquad \begin{array}{c}\vdots\ \Delta_2\\ \phi\end{array}}{\psi} \qquad \Delta' \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_1'\\ \phi \to \psi\end{array} \qquad \begin{array}{c}\vdots\ \Delta_2'\\ \phi\end{array}}{\psi}$$

  Let h and h' be the assignment functions on Δ and Δ' that agree with f and f' on Δ_1 and Δ_1', and agree with g and g' on Δ_2 and Δ_2', respectively. Then $\langle \Delta, h\rangle \cong \langle \Delta', h'\rangle$.

- Suppose Δ_0 and Δ_0' are derivations of ϕ with assignment functions f and f', $\langle \Delta_0, f\rangle \cong \langle \Delta_0', f'\rangle$, x does not occur free in any open hypothesis of Δ or Δ', and y does not occur free anywhere in Δ'. Let

$$\Delta \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_0\\ \phi\end{array}}{\forall x\phi} \qquad \Delta' \equiv \dfrac{\begin{array}{c}\vdots\ [y/x]\Delta_0'\\ [y/x]\phi\end{array}}{\forall y[y/x]\phi}$$

  Let g be the assignment function on Δ that agrees with f on Δ_0. Let g' be the function that agrees with f' on the open hypotheses of Δ_0', and assigns $[y/x]t$ to any use of $(\forall E)$ to which f' assigns t. Then $\langle \Delta, g\rangle \cong \langle \Delta', g'\rangle$.

- Suppose Δ_0 and Δ_0' are derivations of $\forall x\phi$ with assignment functions f and f', $\langle \Delta_0, f\rangle \cong \langle \Delta_0', f'\rangle$, and t is any term. Let

$$\Delta \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_0\\ \forall x\phi\end{array}}{[t/x]\phi} \qquad \Delta' \equiv \dfrac{\begin{array}{c}\vdots\ \Delta_0'\\ \forall x\phi\end{array}}{[t/x]\phi}$$

  Let g be the assignment function on Δ that agrees with f on Δ_0, and assigns t to the root. Let g' be the assignment function on Δ' that agrees with f' on Δ_0', and assigns t to the root. Then $\langle \Delta, g\rangle \cong \langle \Delta', g'\rangle$.

Figure 2.4: Equivalence of Derivations with Assignment Functions

- $(\forall I)([x]\phi, [x]D)$ is mapped to the tree

$$\dfrac{\begin{array}{c}\vdots\, [\![D]\!] \\ \phi\end{array}}{\forall x \phi}$$

  in which every hypothesis discharged in $[\![D]\!]$ is discharged at the corresponding point, and the assignment function agrees with that of $[\![D]\!]$. By α-conversion, we may assume that x does not occur free in any formula for which an occurrence marker occurs free in D, hence it does not occur free in any undischarged hypothesis of $[\![D]\!]$.

  (Note that different choices of bound variable here yield different, but $\cong$-equivalent, derivations and assignment functions.)

- $(\forall E)([x]\phi, t, D)$ is mapped to the tree

$$\dfrac{\begin{array}{c}\vdots\, [\![D]\!] \\ \forall x \phi\end{array}}{[t/x]\phi}$$

  in which every hypothesis discharged in $[\![D]\!]$ is discharged at the corresponding point, the final use of $(\forall E)$ is assigned the term t, and the assignment function otherwise agrees with that of D

It is easy to prove that $[\![\,]\!]$ is a bijection between proof expressions up to α-convertibility, and pairs consisting of derivations and assignment functions up to the relation $\cong$ on derivations. It follows that the mapping is a bijection between derivations up to $\cong$, and proof expressions considered up to some appropriate equivalence relation. To be precise, let us define:

Definition 2.2 *For any formula ϕ, we define the relation $\sim$ on the set of proof expressions for ϕ inductively as follows:*

- *For any two occurrence markers p, q for the same formula ψ, $D \sim [p/q]D$.*
- *If $x \notin \mathrm{FV}(\phi)$, and D' is the result of replacing an occurrence of $(\forall E)([x]\phi, s, D_0)$ with $(\forall E)([x]\phi, t, D_0)$ within D, then*
$$D \sim D'$$
- *If $D \sim D'$ then $D' \sim D$.*
- *If $D \sim D'$ and $D' \sim D''$, then $D \sim D''$.*

(There is no need to add a reflexivity clause, as this follows from taking $p \equiv q$ in the first clause.)

It is easy to see that this relation is a congruence over those constructors that do not bind occurrence markers:

Lemma 2.3

1. *If D_1 and D_1' are proof expressions for $\phi \to \psi$, D_2 and D_2' are proof expressions for ϕ, $D_1 \sim D_1'$, and $D_2 \sim D_2'$, then then*

$$(\to E)(\phi, \psi, D_1, D_2) \sim (\to E)(\phi, \psi, D_2, D_2') .$$

2. *If D and D' are proof expressions for ϕ, and $D \sim D'$, then*

$$(\forall I)([x]\phi, [x]D) \sim (\forall I)([x]\phi, [x]D') .$$

3. *If D and D' are proof expressions for ϕ, and $D \sim D'$, then*

$$(\forall E)([x]\phi, t, D) \sim (\forall E)([x]\phi, t, D') .$$

Proof We shall prove 1 last, as it is the most complex.

2 is proven by induction on the relation $D \sim D'$.

If $D' \equiv [q/p]D$, then we have

$$\begin{aligned} (\forall I)([x]\phi, [x]D') &\equiv (\forall I)([x]\phi, [x][q/p]D) \\ &\equiv [q/p](\forall I)([x]\phi, [x]D) \\ &\sim (\forall I)([x]\phi, [x]D) . \end{aligned}$$

If D' is the result of replacing an occurrence of $(\forall E)([y]\psi, s, D_0)$ with $(\forall E)([y]\psi, t, D_0)$ in D, then $(\forall I)([x]\phi, [x]D')$ is the result of replacing an occurrence of $(\forall E)([y]\psi, s, D_0)$ with $(\forall E)([y]\psi, t, D_0)$ in $(\forall I)([x]\phi, [x]D)$. The other two cases are simple.

3 is proven similarly.

For 1, it is sufficient to prove

(i) If $D_1 \sim D_1'$, then $(\to E)(\phi, \psi, D_1, D_2) \sim (\to E)(\phi, \psi, D_1', D_2)$.

(ii) If $D_2 \sim D_2'$, then $(\to E)(\phi, \psi, D_1, D_2) \sim (\to E)(\phi, \psi, D_1, D_2')$.

For then, if $D_1 \sim D_1'$ and $D_2 \sim D_2'$, then

$$\begin{aligned} (\to E)(\phi, \psi, D_1, D_2) &\sim (\to E)(\phi, \psi, D_1', D_2) \\ &\sim (\to E)(\phi, \psi, D_1', D_2') \end{aligned}$$

and the result follows by transitivity.

Each of the two claims is proven similarly; we shall prove only the first. The proof is by induction on the relation $D_1 \sim D_1'$. We shall deal here only with the case $D_1' \equiv [p/q]D_1$; the other cases are straightforward.

If $D_1' \equiv [p/q]D_1$, where p and q are occurrence markers for ψ, choose some occurrence marker r for ψ that does not occur in D_1, D_1' or D_2. Let

$$D^* \equiv (\to E)(\phi, \psi, D_1, [r/q]D_2) .$$

We shall prove that

$$(\to E)(\phi, \psi, D_1, D_2) \equiv [q/r]D^* \tag{2.3}$$

$$(\to E)(\phi, \psi, D_1', D_2) \equiv [q/r][p/q]D^* \tag{2.4}$$

It follows that $(\to E)(\phi, \psi, D_1, D_2) \sim D^* \sim (\to E)(\phi, \psi, D_1', D_2)$ as required.

To prove (2.3):

$$\begin{aligned} [q/r]D^* &\equiv [q/r](\to E)(\phi, \psi, D_1, [r/q]D_2) \\ &\equiv (\to E)(\phi, \psi, D_1, [q/r][r/q]D_2) \qquad (r \text{ not occurring in } D_1) \\ &\equiv (\to E)(\phi, \psi, D_1, D_2) \qquad (r \text{ not occurring in } D_2) \end{aligned}$$

To prove (2.4):

$$\begin{aligned} [q/r][p/q]D^* &\equiv [q/r][p/q](\to E)(\phi, \psi, D_1, [r/q]D_2) \\ &\equiv [q/r](\to E)(\phi, \psi, [p/q]D_1, [r/q]D_2) \\ &\equiv [q/r](\to E)(\phi, \psi, D_1', [r/q]D_2) \\ &\equiv (\to E)(\phi, \psi, D_1', [q/r][r/q]D_2) \\ &\equiv (\to E)(\phi, \psi, D_1', D_2) \end{aligned}$$

as required. □

It is not the case that $\sim$ is a congruence over $(\to I)$; for let p and q be two distinct occurrence markers for ϕ. Then $p \sim q$, but $(\to I)(\phi, \phi, [p]p) \nsim (\to I)(\phi, \phi, [p]q)$ as proof expressions for $\phi \to \phi$. However, we do have:

Lemma 2.4 *Suppose D and D' are proof expressions for ψ, $D \sim D'$, and the occurrence marker p for ϕ occurs in D in the same locations as it does in D'. Then*

$$(\forall I)(\phi, \psi, [p]D) \sim (\forall I)(\phi, \psi, [p]D') .$$

Proof Again, this is proven by induction on the relation $D \sim D'$.

If $D \equiv D'$, the result follows from the fact that $\sim$ is reflexive. If $D' \equiv [q/r]D$, and this is a non-trivial substitution (i.e. $D \not\equiv D'$), then neither q nor r can be the occurrence marker p. Therefore,

$$\begin{aligned} (\forall I)(\phi, \psi, [p]D') &\equiv (\forall I)(\phi, \psi, [p][q/r]D) \\ &\equiv [q/r](\forall I)(\phi, \psi, [p]D) \\ &\sim (\forall I)(\phi, \psi, [p]D) . \end{aligned}$$

The other four cases are simple. □

We can now prove:

Theorem 2.5 *Given two proof expressions D, D' for the same formula ϕ, let $[\![D]\!]$ consist of the derivation Δ with assignment function f, and $[\![D']\!]$ consist of the derivation Δ' with assignment* *function g. Then:*

$$\begin{aligned} \langle \Delta, f\rangle \cong \langle \Delta', g\rangle &\quad\Leftrightarrow\quad D \equiv_\alpha D' \\ \Delta \cong \Delta' &\quad\Leftrightarrow\quad D \sim D' \end{aligned}$$

Proof The easiest way to prove this theorem is to define the inverse to $[\![\,]\!]$. We shall define, for each derivation Δ with an assignment function f, the proof term $\langle\!|\Delta, f|\!\rangle$.

If Δ is the one-line derivation

$$\phi$$

in which the sole hypothesis is assigned occurrence marker p by f, then

$$\langle\!|\Delta, f|\!\rangle \equiv p\,.$$

If Δ is the derivation

$$\dfrac{\begin{matrix} \vdots\,\Delta_1 \\ \psi \end{matrix}}{\phi \to \psi}$$

choose some fresh occurrence marker p for ϕ. Let g be the assignment function for Δ_1 that assigns every hypothesis that was open in Δ the same occurrence marker as f does in Δ, and in addition assigns every hypothesis that is discharged at the root of Δ p. Then

$$\langle\!|\Delta, f|\!\rangle \equiv (\to I)(\phi, \psi, [p]\,\langle\!|\Delta_1, g|\!\rangle)\,.$$

If Δ is the derivation

$$\dfrac{\begin{matrix} \vdots\,\Delta_1 \\ \phi \to \psi \end{matrix} \qquad \begin{matrix} \vdots\,\Delta_2 \\ \phi \end{matrix}}{\psi}$$

then

$$\langle\!|\Delta, f|\!\rangle \equiv (\to E)(\phi, \psi, \langle\!|\Delta_1, f_1|\!\rangle\,, \langle\!|\Delta_2, f_2|\!\rangle)\,,$$

where f_1 and f_2 are the appropriate restrictions of f.

If Δ is the derivation

$$\dfrac{\begin{matrix} \vdots\,\Delta_1 \\ \phi \end{matrix}}{\forall x\phi}$$

then

$$\langle\!|\Delta, f|\!\rangle \equiv (\forall I)([x]\phi, [x]\,\langle\!|\Delta_1, f_1|\!\rangle)\,,$$

where f_1 is the restriction of f to Δ_1.

If Δ is the derivation

$$\dfrac{\begin{matrix} \vdots\,\Delta_1 \\ \forall x\phi \end{matrix}}{[t/x]\phi}$$

with t being the term assigned to the root by f, then

$$\langle\!|\Delta, f|\!\rangle \equiv (\forall E)([x]\phi, t, \langle\!|\Delta_1, f_1|\!\rangle)\,,$$

where f_1 is the restriction of f to Δ_1.

It is simple to prove

- If $\langle\Delta, f\rangle \cong \langle\Delta', g\rangle$, then $\langle\!|\Delta, f|\!\rangle \equiv \langle\!|\Delta', g|\!\rangle$.
- If D is a proof expression for ϕ, then $[\![D]\!]$ is a derivation with conclusion ϕ together with an assignment function, and
$$\langle\!|[\![D]\!]|\!\rangle \equiv D\,.$$
- If Δ is a derivation with conclusion ϕ and f an assignment function f for Δ, then $\langle\!|\Delta, f|\!\rangle$ is a proof expression for ϕ, and
$$[\![\langle\!|\Delta, f|\!\rangle]\!] \cong \langle\Delta, f\rangle\,.$$

The first statement of the theorem follows, for, if $[\![D]\!] \cong [\![D']\!]$, then

$$\begin{aligned} \langle\!|[\![D]\!]|\!\rangle &\equiv \langle\!|[\![D']\!]|\!\rangle \\ \therefore D &\equiv D' \end{aligned}$$

To prove the second statement of the theorem, it is sufficient to show:

1. If D and D' are proof expressions for ϕ, and $D \sim D'$, then the derivations in $[\![D]\!]$ and $[\![D']\!]$ are related by $\cong$; the assignment functions may be different.
2. If $\Delta \cong \Delta'$, f is any assignment function for Δ, and g is any assignment function for Δ, then $\langle\!|\Delta, f|\!\rangle \sim \langle\!|\Delta', g|\!\rangle$. Further, if f and g assign an occurrence marker p to exactly the same hypotheses, then p appears in the same locations in $\langle\!|\Delta, f|\!\rangle$ as in $\langle\!|\Delta', g|\!\rangle$.

1 is proven by an induction on the relation $D \sim D'$. The first case:

> For any proof expression D for ϕ, and any occurrence markers p and q for ψ, the derivations in $[\![[p/q]D]\!]$ and $[\![D]\!]$ are related by $\cong$.

is proven by a simple induction on D. No case is at all difficult. For example, in the case where D is q and $[p/q]D$ is p, the derivation in $[\![[p/q]D]\!]$ and $[\![D]\!]$ is

$$\phi$$

We note that the two assignment functions in $[\![D]\!]$ and $[\![[p/q]D]\!]$ differ only in that, where the first assigns q to a hypothesis ϕ, the second assigns p.

The second case:

> For any proof expression D for ϕ, let D' be the result of replacing an occurrence of $(\forall E)([x]\phi, s, D_0)$ with $(\forall E)([x]\phi, t, D_0)$, where $x \notin \mathrm{FV}(\phi)$. Then $[\![D]\!]$ and $[\![D']\!]$ consist of the same derivation, possibly with different assignment functions.

is proven again by a simple induction on D. Again, no case is difficult. For example, if D is $(\forall E)([x]\phi, s, D_0)$ and D' is $(\forall E)([x]\phi, t, D_0)$, the derivation in $[\![D]\!]$ and $[\![D']\!]$ is

$$\dfrac{\begin{array}{c}\vdots\, [\![D_0]\!] \\ \forall x \phi\end{array}}{\phi}$$

We note that the two assignment functions in $[\![D]\!]$ and $[\![D']\!]$ differ only in that the first assigns the term s to some point to which the second assigns t.

The other two cases follow from the fact that $\cong$ is an equivalence relation.

2 is proven by induction on the size of the derivation Δ and Δ'; as $\Delta \cong \Delta'$, these must have the same size. All cases are straightforward; we shall deal here only with the $(\rightarrow I)$ and $(\forall E)$ cases.

Suppose Δ and Δ' each consist of the derivation

$$\Delta \equiv \dfrac{\begin{array}{c}\vdots\, \Delta_0 \\ \psi\end{array}}{\phi \rightarrow \psi} \qquad \Delta' \equiv \dfrac{\begin{array}{c}\vdots\, \Delta'_0 \\ \psi\end{array}}{\phi \rightarrow \psi}$$

with assignment functions f and g respectively, where $\Delta \cong \Delta_0$. Let p be some fresh occurrence marker for ϕ. Let f_0 be the assignment function for Δ_0 that assigns p to every hypothesis that was discharged at the root of Δ, and otherwise agrees with f. Let g_0 be the assignment function for Δ'_0 that assigns p to every hypothesis that was discharged at the root of Δ', and otherwise agrees with g.

By the induction hypothesis,

$$\langle\!|\Delta_0, f_0|\!\rangle \sim \langle\!|\Delta'_0, g_0|\!\rangle$$

and p occurs free in the same locations in $\langle\!|\Delta_0, f_0|\!\rangle$ and $\langle\!|\Delta'_0, g_0|\!\rangle$. Therefore, by Lemma 2.4,

$$(\rightarrow I)(\phi, \psi, [p]\, \langle\!|\Delta_0, f_0|\!\rangle) \sim (\rightarrow I)(\phi, \psi, [p]\, \langle\!|\Delta'_0, g_0|\!\rangle)\,,$$

i.e.

$$\langle\!|\Delta, f|\!\rangle \sim \langle\!|\Delta', g|\!\rangle\ .$$

Further, if f and g agree on their assignments of q, so do f_0 and g_0; hence q occurs in the same locations in $\langle\!|\Delta_0, f_0|\!\rangle$ and $\langle\!|\Delta'_0, g_0|\!\rangle$; hence q occurs in the same locations in $\langle\!|\Delta, f|\!\rangle$ and $\langle\!|\Delta', g|\!\rangle$.

Suppose now

$$\Delta \equiv \dfrac{\begin{array}{c}\vdots\, \Delta_0 \\ \forall x\phi\end{array}}{[t/x]\phi} \qquad \Delta' \equiv \dfrac{\begin{array}{c}\vdots\, \Delta'_0 \\ \forall x\phi\end{array}}{[t/x]\phi}$$

with assignment functions f and g respectively, where $\Delta_0 \cong \Delta'_0$. Let f_0 and g_0 be the restrictions of f and g to Δ_0 and Δ'_0, respectively. By the induction hypothesis,

$$\langle\!|\Delta_0, f_0|\!\rangle \sim \langle\!|\Delta'_0, g_0|\!\rangle\ .$$

Now, if $x \in \mathrm{FV}(\phi)$, f and g must both assign t to the root of the derivation. Therefore,

$$\begin{array}{rcll} \langle\!| \Delta, f |\!\rangle & \equiv & (\forall E)([x]\phi, t, \langle\!| \Delta_0, f_0 |\!\rangle) & \\ & \sim & (\forall E)([x]\phi, t, \langle\!| \Delta'_0, g_0 |\!\rangle) & \text{(Lemma 2.3)} \\ & \equiv & \langle\!| \Delta', g |\!\rangle \ . & \end{array}$$

If $x \notin \mathrm{FV}(\phi)$, let f assign s to the root of the derivation, and g assign t. Then

$$\begin{array}{rcll} \langle\!| \Delta, f |\!\rangle & \equiv & (\forall E)([x]\phi, s, \langle\!| \Delta_0, f_0 |\!\rangle) & \\ & \sim & (\forall E)([x]\phi, s, \langle\!| \Delta'_0, g_0 |\!\rangle) & \text{(Lemma 2.3)} \\ & \sim & (\forall E)([x]\phi, t, \langle\!| \Delta'_0, g_0 |\!\rangle) & \\ & \equiv & \langle\!| \Delta', g |\!\rangle \ . & \end{array}$$

In either case, if f and g agree on the assignment of q, so do f_0 and g_0; hence, q occurs in the same locations in $\langle\!| \Delta_0, f_0 |\!\rangle$ and $\langle\!| \Delta'_0, g_0 |\!\rangle$; hence q occurs in the same locations in $\langle\!| \Delta, f |\!\rangle$ and $\langle\!| \Delta', g |\!\rangle$. $\square$

Corollary 2.6 *Let $[\![D]\!]'$ be the derivation $[\![D]\!]$ with the assignment function forgotten. Then $[\![\,]\!]$ is a bijection between the set of proof expressions up to $\sim$, and the set of derivations up to $\cong$.*

This, then, is the relationship between our presentation of natural deduction in terms of proof expressions, and the first traditional account of natural deduction, that we have been seeking to establish.

Account Two

For our second traditional account, a similar correspondence holds. This time, the appropriate equivalence relation allows us to relabel occurrence markers under a binding as well as those occurring free. Formally, define the relation $\simeq$ on proof expressions thus:

- For any occurrence markers p, q for the same formula ϕ, $p \simeq q$.
- If $D \simeq D'$, then $(\to I)(\phi, \psi, [p]D) \simeq (\to I)(\phi, \psi, [p]D')$.
- If $D_1 \simeq D'_1$ and $D_2 \simeq D'_2$, then $(\to E)(\phi, \psi, D_1, D_2) \simeq (\to E)(\phi, \psi, D'_1, D'_2)$.
- If $D \simeq D'$ then $(\forall I)([x]\phi, [x]D) \simeq (\forall I)([x]\phi, [x]D')$.
- If $D \simeq D'$ then $(\forall E)([x]\phi, t, D) \simeq (\forall E)([x]\phi, t, D')$.
- If $D \simeq D'$ and $x \notin \mathrm{FV}(\phi)$, then $(\forall E)([x]\phi, s, D) \simeq (\forall E)([x]\phi, t, D')$.

Theorem 2.7 *There is a bijection between derivations of ϕ depending on $\psi_1, \ldots, \psi_n$ identified up to $\cong$, and the proof expressions for ϕ identified up to the relation $\simeq$ in which every free occurrence marker that occurs is one of $\phi_1, \ldots, \phi_n$.*

This can be proven either directly, by a similar method to above, or by establishing a correspondence between the derivations in Account One and Account Two. We omit the details of the proof.

Account Three

If we inductively define a derivability relation as in the third account, then we have

Theorem 2.8 $\Gamma \vdash \phi$ *iff there is a proof expression D for ϕ such that every free occurrence marker in D is for some formula $\psi \in \Gamma$.*

We could prove this theorem by establishing a correspondence with one of the previous two accounts; or, this result can be established directly. The left-to-right direction is proven by induction on the relation $\Gamma \vdash \phi$; the right-to-left, by induction on the proof expression p. Neither gives any great difficulty.

2.2.2 Schroeder-Heister's Extension

In [SH84], Schroeder-Heister presents an extension of natural deduction, in which rules as well as formulas can be assumed and discharged. Specifically, we define rules of various *levels* as follows:

The rules of level 0 are the axioms. The rules of level 1 are the rules familiar from Hilbert systems, of the form

$$\frac{\phi_1 \quad \cdots \quad \phi_n}{\psi}$$

These may be thought of as rules that have rules of level 0 as premises. The rules of level 2 are the usual form of rules in natural deduction:

$$\frac{\begin{matrix}[\Gamma_1] \\ \vdots \\ \phi_1\end{matrix} \qquad \cdots \qquad \begin{matrix}[\Gamma_n] \\ \vdots \\ \phi_n\end{matrix}}{\psi}$$

where each Γ_i is a finite set of formulae to be discharged. These may be thought of as rules that have rules of level 1 as their premises; specifically, the rule given above has the n level 1 rules

$$\frac{\Gamma_1}{\phi_1} \qquad \cdots \qquad \frac{\Gamma_n}{\phi_n}$$

as premises.

More generally, we define a rule of level $n+1$ to be a rule whose premises are all rules of level n. Thus, a rule of level $n+2$ has the form

$$\frac{\begin{matrix}[R_{11}] \cdots [R_{1k_1}] \\ \vdots \\ \phi_1\end{matrix} \qquad \cdots \qquad \begin{matrix}[R_{n1}] \cdots [R_{nk_n}] \\ \vdots \\ \phi_n\end{matrix}}{\psi}$$

where each R_{ij} is a rule of level n. This rule should be read as follows:

> If it is the case that:
>
> - if we (temporarily) add the rules $R_{11}, \ldots, R_{1k_1}$ to the system, we are able to deduce ϕ_1;
> - $\cdots$
> - if we add the rules $R_{n1}, \ldots, R_{nk_n}$ to the system, we are able to deduce ϕ_n
>
> then ψ is a theorem.

Thus, for example, the hypotheses of natural deduction are seen as temporarily assumed axioms. The $(\rightarrow I)$ introduction rule

$$\frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \rightarrow \psi}$$

is read as:

> If, when we add the axiom ϕ to our system, we are able to prove ψ, then $\phi \rightarrow \psi$ is a theorem.

The formal definition of a rule is:

Definition 2.9 *A* rule *is a finite tree of formulas. Its* level *is the length of its longest branch. A* theory *is a set of rules.*

The definition of derivations as trees, corresponding to Accounts One and Two above, is given in [SH84]; it is quite technically involved, so we do not reproduce it here. However, the inductive definition of the accompanying derivability relation, corresponding to Account Three above, takes on quite a pleasing form.

We define the relation

$$T \vdash \phi\,,$$

where T is a theory and ϕ a formula, inductively as follows:

> Suppose the rule
>
> $$(R) \qquad \frac{\begin{array}{c}[R_{11}] \cdots [R_{1k_1}]\\ \vdots\\ \phi_1\end{array} \qquad \cdots \qquad \begin{array}{c}[R_{n1}] \cdots [R_{nk_n}]\\ \vdots\\ \phi_n\end{array}}{\psi}$$
>
> is in T. If
>
> $$\begin{array}{rcl} T \cup \{R_{11}, \ldots, R_{1k_1}\} & \vdash & \phi_1 \\ & \vdots & \\ T \cup \{R_{n1}, \ldots, R_{nk_n}\} & \vdash & \phi_n \end{array}$$

then

$$T \vdash \psi$$

We shall see that this scheme fits very neatly into the logical frameworks that we shall build; the concept of 'level' matches very closely the notion of 'order' of parametrization that we shall introduce in the next chapter. For now, we shall show how proof expressions can be given to this system.

Fix a set of *occurrence markers* for each rule R.[2] The set of *proof expressions* for a formula R are now defined as follows.

Suppose p is an occurrence marker for the rule

$$(R) \quad \frac{\begin{array}{c}[R_{11}] \cdots [R_{1k_1}] \\ \vdots \\ \phi_1\end{array} \qquad \cdots \qquad \begin{array}{c}[R_{n1}] \cdots [R_{nk_n}] \\ \vdots \\ \phi_n\end{array}}{\psi}$$

and, for $i = 1, \ldots, n$ and $j = 1, \ldots, k_i$, q_{ij} is an occurrence marker for R_{ij} and D_i a proof expression for ϕ_i. Then

$$p[[q_{11}, \ldots, q_{1k_1}]D_1, \ldots, [q_{n1}, \ldots, q_{nk_n}]D_n]$$

is a proof expression for ψ. Each q_{ij} is bound within D_i in this term.

Note that this definition generalizes the one for predicate logic given above, if we take $(\to I)$ to be an occurrence marker for the rule

$$\frac{\begin{array}{c}[\phi] \\ \vdots \\ \psi\end{array}}{\phi \to \psi}$$

and similarly for the other rules.

It is quite simple to prove

Theorem 2.10 *$S \vdash \phi$ iff there is a proof expression D for ϕ such that every occurrence marker p that occurs free in D is an occurrence marker for some member of S.*

Proof An induction on proof expressions in one direction, and on D in the other. Neither case presents any difficulty. $\square$

Despite the apparent naturalness of this extension to natural deduction, it is striking that one hardly ever needs a rule of deduction of level > 2. We can give rules for the connectives and quantifiers of first- and second-order logic, as well as such esoteric constructions as the transitive closure of a relation, or the well-founded part of a relation, using only second-level

[2] If one wishes, one could also introduce constants for each rule R, to distinguish between those rules we take as part of the theory, and those rules we have temporarily assumed as hypotheses.

rules. We can give alternative presentations of some connectives using third-level rules; for example, an alternative elimination rule for $\to$ is the third-level rule

$$\dfrac{\phi \to \psi \qquad \begin{matrix} \left[\dfrac{\phi}{\psi}\right] \\ \vdots \\ \chi \end{matrix}}{\chi}$$

As another example, we know that we can characterise classical logic by the addition to intuitionistic logic of axioms such as $\phi \vee \neg\phi$ or $((\phi \to \psi) \to \phi) \to \phi$, or rules for proof by contradiction or proof by cases on ϕ and $\neg\phi$. If one wishes a characterisation that does not depend on either of the connectives $\to$ or $\neg$, classical logic can be defined as the result of adding the third-level rule

$$\dfrac{\begin{matrix} \left[\dfrac{\phi}{\psi}\right] \\ \vdots \\ \phi \end{matrix}}{\phi}$$

to intuitionistic logic.

Uses for rules of level > 3 seem to be extremely difficult to find; it seems that none is known at present.

Chapter 3

The Type Framework TF

We now proudly present our first example of a weak framework, the *type framework* TF. As was discussed in the introduction, the framework TF includes nothing but what is essential to the representation of an object theory. In particular, it contains neither lambda-abstraction nor local definition; its basic concepts are parameterisation, and the instantiation of parameters, alone. As its name suggests, TF was originally intended for the representation of type theories; however, it shall prove itself to be equally adept at the representation of deductive systems.

TF first appeared in [Acz]. Our development here goes quite some distance beyond that presentation: most notably, we introduce the set of arities to organise the grammar, and we make explicit the definition of instantiation.

3.1 Arities

We begin by introducing a set of *arities*. These arities shall prove very useful in organising the syntax of TF; time and again we shall find ourselves defining families of expressions, or operations on expressions, by recursion over the set of arities. These arities play a very similar role to the arities in Martin-Löf's theory of arities (see [NPS90]), which in turn behave similarly to the types of the simply-typed lambda calculus.

The definition of the set of arities is deceptively simple:

> An arity is a finite sequence of arities.

We may write this definition in the form of a BNF-style grammar thus:

$$\text{Arity } \alpha ::= (\alpha, \ldots, \alpha)$$

The base case for this definition is the arity consisting of the empty sequence, (), which we write as $\mathbf{0}$. The next arities that can be formed are

$$\overbrace{(\mathbf{0}, \ldots, \mathbf{0})}^{n}$$

for positive n; we shall write this arity as $\mathbf{n}$.

The intuition behind arities may be explained by defining, informally, what it is to be an *α-ary object* in a logical framework for each arity α. The definition is by recursion on α as follows:

- The base (i.e. $\mathbf{0}$-ary or ()-ary) objects are the entities (terms and types) of the object logic.
- An $(\alpha_1, \alpha_2, \ldots, \alpha_n)$-ary object is a function which takes n arguments — an α_1-ary object, an α_2-ary object, ..., and an α_n-ary object — and returns an entity of the object theory.

  Alternatively, an $(\alpha_1, \ldots, \alpha_n)$-ary object is a family of entities of the object theory parametrised by n parameters: an α_1-ary object, ..., and an α_n-ary object.

Thus, for example:

- A $\mathbf{2}$-ary, or $(\mathbf{0}, \mathbf{0})$-ary, object is a binary operation on the object theory entities. An example would be a binary term constructor that does not bind any variables in its arguments, such as the connectives $\wedge$, $\vee$ and $\rightarrow$ in propositional logic.
- A $(\mathbf{0}, \mathbf{1})$-ary, or $(\mathbf{0}, (\mathbf{0}))$-ary, object is a function which takes an object theory entity and a unary function on the object theory entities, and returns an object theory entity. An example would be a binary type constructor that binds a single variable in its second argument; the type constructor Π, for example, takes a type A and the A-indexed family of types given by $B[x]$, where x is a variable of type A, and returns the type $\Pi x : A.B[x]$.

Only the base objects (that is, the object theory entities) are represented by terms in TF. Nevertheless, it shall be highly convenient to be able to speak of the objects of higher arity when discussing the system. In particular, TF has α-ary variables and α-ary constants of any arity α, representing arbitrary and particular α-ary objects, respectively; however, only the values of these objects when applied to appropriate arguments shall be represented by terms in TF. Formally, when $\alpha \neq \mathbf{0}$, an α-ary variable or constant is not a term on its own; only when applied to the appropriate number of terms as arguments does it form a term.

While discussion of α-ary objects is often a useful way to explain the intuition behind the various aspects of TF, it is by no means the only meaning explanation that can be given to the system. One might prefer, for example, to see the arity of a constant or variable simply as an indication of to how many arguments that symbol must be applied, and how many variables are bound in each. For example, by our first meaning explanation, a $(\mathbf{0}, \mathbf{1})$-ary constant stands for a binary function, one of whose arguments must itself be a unary function. We may prefer simply to state that a $(\mathbf{0}, \mathbf{1})$-ary constant must be applied to two terms, and binds a single base variable in the second of these terms.

We shall write $\alpha\hat{}\beta$ for the *concatenation* of the two arities α and β; that is,

$$(\alpha_1, \ldots, \alpha_m)\hat{}(\beta_1, \ldots, \beta_n) \equiv (\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n)\,.$$

This operation shall be most useful to us when we come to relate TF to a framework that uses lambda-abstraction. For example, if x is an α-ary variable, and M a term that represents a

β-ary object, then

$$\lambda x.M$$

is a term that represents an $(\alpha)\hat{}\beta$-ary object.

Orders We ascribe an *order* to each arity as follows:

- The only 0th-order arity is $\mathbf{0}$.
- If the largest order among the arities $\alpha_1, \ldots, \alpha_n$ is k, then

  $$(\alpha_1, \ldots, \alpha_n)$$

  is a $k + 1$st-order arity.

The first-order arities are thus the arities $\mathbf{n}$ for positive n. The second-order arities are those of the form

$$(\mathbf{n_1}, \ldots, \mathbf{n_k})$$

where at least one of the n_is is positive.

We shall repeatedly have occasion to define classes of α-ary items for every arity α; for example, α-ary variables, constants, abstractions, objects, kinds, and so forth. When we do so, we shall refer to $\mathbf{0}$-ary items as *base* items, and use "nth-order item" to refer to an item of arity α for some nth-order arity α.

In particular, it follows from the informal definition of "α-ary object" above that:

- The 0th-order objects, or base objects, are the entities of the object theory.
- The first-order objects are functions that take some number of object theory entities, and return an object theory entity.
- The second-order objects are function that take object theory entities and at least one first-order object, and return an object theory entity.

and so forth. Or, if one prefers:

- The base constants are entities of the object theory on their own.
- The first-order constants take some number of object theory entities and produce an object theory entity without binding any variables.
- The second-order constants take some number of object theory entities, bind at least one base variable, and return an object theory entity.
- The third-order constants take some number of object theory entities, bind at least one first-order variable and possibly some number of base variables, and returns an object theory entity.

and so forth.

Examples of first-order and second-order objects are easy to come by; they are the constructors that bind no variables, or bind only base variables, respectively. An example of a third-order object is the eliminator **wrec** for W-types in Martin-Löf Type Theory; it's typing rule, as presented in [NPS90], is:

$$\frac{\begin{array}{c} a \in W(A,B) \\ C(v) \text{ set} \qquad [v \in W(A,B)] \\ b(y,z,u) \in C(\mathbf{sup}(y,z)) \\ [y \in A, z(x) \in W(A,B)\ [x \in B(y)], u(x) \in C(z(x))\ [x \in B(y)]] \end{array}}{\mathbf{wrec}(a,b) \in C(a)}$$

Thus, **wrec**'s second argument b is a function, two of whose arguments, z and u, are themselves first-order objects.. Thus, b is a second-order object, and so **wrec** is a third-order object.

We have to look quite hard to find examples of objects higher than third-order being used. One example is UTT [Luo94], where, if one of the inductive schemata $\overline{\Theta}$ is nth-order, the corresponding constructor $\iota_i[\overline{\Theta}]$ is nth-order, and the eliminator $E[\overline{\Theta}]$ is $n+1$st-order.

Example To illustrate the various ideas we introduce in this chapter, as we progress we shall build up within TF a representation of the *noble fragment* of intuitionistic predicate logic; that is, the part of predicate logic dealing only with implication and universal quantification. We shall use the novel presentation of natural deduction in terms of proof expressions given in the previous chapter. This theory is the result of removing, from the system of predicate logic presented there, all clauses from the definition of formula apart from those dealing with predicate constants, $\rightarrow$ and $\forall$, and all rules of deduction except $(\rightarrow I)$, $(\rightarrow E)$, $(\forall I)$ and $(\forall E)$. For the remainder of this chapter, 'predicate logic' shall mean the noble fragment of intuitionistic predicate logic.

To build this system within TF, we anticipate that we shall need to declare the following objects:

- An **n**-ary object to represent each n-ary function constant.
- An **n**-ary object to represent each n-ary predicate constant.
- A **2**-ary object to represent the formula constructor $\rightarrow$.
- A $(\mathbf{1})$-ary object to represent the formula constructor $\forall$.
- A **1**-ary object to represent the function that, given a formula ϕ, returns the type of all proofs of ϕ.
- For each axiom Ax_n, a **0**-ary object to represent the one-line proof that consists solely of Ax_n.
- A $(\mathbf{0},\mathbf{0},\mathbf{1})$-ary object to represent the rule of deduction $(\rightarrow I)$.
- A **4**-ary object to represent the rule of deduction $(\rightarrow E)$.

- A $(\mathbf{1}, \mathbf{1})$-ary object to represent the rule of deduction $(\forall I)$.
- A $(\mathbf{1}, \mathbf{0}, \mathbf{0})$-ary object to represent the rule of deduction $(\forall E)$.

Note, in particular, the second-order objects. The formula $\forall x \phi[x]$ shall be represented by the $(\mathbf{1})$-ary object $\forall$ applied to the family of formulas specified by the predicate $\phi[x]$; that is, the function which takes a term t and returns a formula $\phi[t]$. This trick — representing an expression E with distinguished free variable x by the function that maps a term t to $[t/x]E$ — is a standard technique for representing binding operators in a logical framework; when building predicate logic in ELF, for example (see [HHP93]), the representation $[\![\forall x \phi]\!]$ of $\forall x \phi$ is

$$\forall(\lambda x : \iota. [\![\phi]\!]) .$$

The fact that such a representation is possible is, of course, the first faint glimmering of the Curry-Howard Isomorphism. However, if one does not like this idea of representing expressions with free variables by functions, we refer the reader to the comments we made above: reading constants and variables as functions is not the only way to read TF. The fact that Π shall be represented by a $(\mathbf{0}, \mathbf{1})$-ary constant need indicate nothing more than that it takes two arguments and binds a single base variable in its second.

Similarly, for example, the proof expression $(\rightarrow I)(\phi, \psi, [p]D)$ is represented by the $(\mathbf{0}, \mathbf{0}, \mathbf{1})$-ary object $(\rightarrow I)$ applied to the formulas ϕ and ψ and the function that, given a proof Q of ϕ, returns the proof expression $[Q/p]D$. Likewise, the proof expression $(\forall I)([x]\phi, [x]D)$ is represented by $(\forall I)$ applied to the predicate ϕ (that is, the function that maps t to $[t/x]\phi$), and the function that takes a term t and returns the proof $[t/x]D$. Or, if one prefers, we will represent $(\rightarrow I)$ by a $(\mathbf{0}, \mathbf{0}, \mathbf{1})$-ary constant because it takes three arguments and binds a base variable (the occurrence marker) in its third, and $(\forall I)$ by a $(\mathbf{1}, \mathbf{1})$-ary constant because it takes two arguments and binds a base individual variable in each.

3.2 Grammar of TF

We introduce infinite sets $\mathcal{V}^\alpha$ and $\mathcal{C}^\alpha$ of *α-ary variables* and *α-ary constants* for every arity α.[1] We shall use the letters x^α, y^α for arbitrary α-ary variables, and a^α, b^α, c^α for arbitrary α-ary constants; z^α shall usually denote a symbol that is either an α-ary variable or an α-ary constant. We shall take the opportunity to drop the arity-indicating superscripts when possible.

The set of *terms* of TF is then defined recursively thus:

- If z^α is an α-ary constant or variable, where

$$\begin{aligned} \alpha &\equiv (\alpha_1, \ldots, \alpha_n) \\ &\equiv ((\alpha_{11}, \ldots, \alpha_{1r_1}), \ldots, (\alpha_{n1}, \ldots, \alpha_{nr_n})) \end{aligned}$$

[1] We assume that no constant or variable has more than one arity, nor is any constant a variable. Henceforth, such disjointness conditions shall always be taken as read.

and, for $i = 1, \ldots, n$, M_i is a term, and

$$x_{i1}, \ldots, x_{ir_i}$$

is a sequence of distinct variables, x_{ij} being of arity α_{ij}, then

$$z^{\alpha}[[x_{11}, \ldots, x_{1r_1}]M_1, \ldots, [x_{n1}, \ldots, x_{nr_n}]M_n] \tag{3.1}$$

is a term.

In the term (3.1) above, each variable x_{ij} is bound within the term M_i. We identify terms up to α-convertibility as usual.

The base case of this definition is, of course, the case $\alpha \equiv \mathbf{0}$:

If $z^{\mathbf{0}}$ is a base constant or variable, then

$$z^{\mathbf{0}}[]$$

is a term.

We shall normally omit the empty list of arguments, and just write $z^{\mathbf{0}}$ for the term $z^{\mathbf{0}}[]$.

For first-order variables and constants, the definition becomes:

If $z^{\mathbf{n}}$ is an $\mathbf{n}$-ary constant or variable, and $M_1, \ldots, M_n$ are terms, then

$$z^{\mathbf{n}}[[]M_1, \ldots, []M_n]$$

are terms.

Again, we shall omit the empty lists of bound variables, and write this term as $z^{\mathbf{n}}[M_1, \ldots, M_n]$.

For second-order variables and constants, the definition becomes:

If z is an $(\mathbf{n_1}, \ldots, \mathbf{n_k})$-ary variable or constant, and, for $i = 1, \ldots, k$, M_i is a term and

$$\langle x_{i1}, \ldots, x_{in_i} \rangle$$

a list of n_i distinct base variables, then

$$z[[x_{11}, \ldots, x_{1n_1}]M_1, \ldots, [x_{k1}, \ldots, x_{kn_k}]M_k]$$

is a term. Each x_{ij} is bound in the term M_i that follows it.

The subexpressions $[x_{i1}, \ldots, x_{ir_i}]M_i$ are not first-class expressions in TF; they shall never appear except as the argument to some constant or variable. Nevertheless, it shall be convenient to have some way of referring to these pieces of raw syntax. We therefore make the following definitions, for any arity $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$:

- An *α-ary variable sequence* $\vec{x}^{\alpha}$ is a sequence

$$\langle x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \rangle$$

of n variables, all distinct, the ith of which, $x_i^{\alpha_i}$ is of arity α_i.

- An *α-ary abstraction* M^α is an expression of the form

$$[\vec{x}^\alpha]M$$

where $\vec{x}^\alpha$ is an α-ary variable sequence, and M a term. We take each member of $\vec{x}^\alpha$ to be bound within M in this expression.

- An *α-ary abstraction sequence* $\vec{M}^\alpha$ is a sequence

$$M^{\alpha_1}, \ldots, M^{\alpha_n}$$

where each M^{α_i} is an α_i-ary abstraction.

The definition of the set of terms may therefore be rephrased thus:

If z^α is an α-ary variable or constant, and $\vec{M}^\alpha$ is an α-ary abstraction sequence, then

$$z^\alpha[\vec{M}^\alpha]$$

is a term.

There is one more notational abbreviation that we shall make use of: that of η-contraction. We shall sometimes write a constant or variable when we should, strictly, write its η-long form. Formally, this means we are identifying each α-ary variable or constant with a particular α-ary abstraction. For example, we may write the **2**-ary constant $c^{\mathbf{2}}$ in place of the abstraction

$$[x^{\mathbf{0}}, y^{\mathbf{0}}]c^{\mathbf{2}}[x^{\mathbf{0}}, y^{\mathbf{0}}]$$

or the (**1**)-ary constant $d^{(\mathbf{1})}$ in place of the abstraction

$$[x^{\mathbf{1}}]d^{(\mathbf{1})}[[y^{\mathbf{0}}]x^{\mathbf{1}}[y^{\mathbf{0}}]] \, .$$

We emphasise again that this is an abbreviation we employ only for the purposes of discussing TF; officially, all terms of TF are η-long.

The formal definition is as follows.

Definition 3.1 (η-long Form) *The* η-long form $\overline{z}$ *of an α-ary variable or constant z is an α-ary abstraction defined by recursion on α as follows.*

- *A base variable or constant is its own η-long form.*

- *If z is an $(\alpha_1, \ldots, \alpha_n)$-ary variable or constant, its η-long form is*

$$\overline{z} \equiv [y_1, \ldots, y_n]z[\overline{y_1}, \ldots, \overline{y_n}]$$

where y_i is an α_i-ary variable. (By α-conversion, it does not matter which variables we choose here.)

This shall be the first of many definitions by recursion over the arities.

Example In our predicate logic example, we shall need to declare the following constants, corresponding to the objects that we earlier decided we need:

- For each n-ary function constant f, an $\mathbf{n}$-ary constant $\overline{f}$.
- For each n-ary predicate constant R, an $\mathbf{n}$-ary constant $\overline{R}$.
- A $\mathbf{2}$-ary constant $\rightarrow$.
- A $(\mathbf{1})$-ary constant $\forall$.
- A $\mathbf{1}$-ary constant $\mathbf{Prf}$.
- For each axiom Ax_n, a $\mathbf{0}$-ary constant Ax_n.
- A $(\mathbf{0}, \mathbf{0}, \mathbf{1})$-ary constant $(\rightarrow I)$.
- A $\mathbf{4}$-ary constant $(\rightarrow E)$.
- A $(\mathbf{1}, \mathbf{1})$-ary constant $(\forall I)$.
- A $(\mathbf{1}, \mathbf{0}, \mathbf{0})$-ary constant $(\forall E)$.

We shall use the $\mathbf{0}$-ary variables of TF to serve as both individual variables and occurrence markers.

The terms formed by $\rightarrow$ are of the form

$$\rightarrow [M, N] .$$

If M represents the formula ϕ, and N the formula ψ, then this is the term we shall use to represent $\phi \rightarrow \psi$.

The terms formed by $\forall$ are of the form

$$\forall[[x]M] .$$

If we use x as an individual variable, and the term M represents the formula ϕ, then this is the term we shall use to represent $\forall x \phi$.

The terms formed by $(\rightarrow I)$ are of the form

$$(\rightarrow I)[M, N, [p]P] .$$

If we use p as a occurrence marker, the term M represents the formula ϕ, the term N represents ψ, and the term P represents the proof expression D containing the occurrence marker p for ϕ, then the term $(\rightarrow I)[M, N, [p]P]$ shall represent the proof expression

$$(\rightarrow I)(\phi, \psi, [p]D) .$$

More formally, let $\mathcal{T}$ be the set of terms, $\mathcal{F}$ the set of formulas, and $\mathcal{P}$ the set of proof expressions of predicate logic, and let $\mathcal{E}$ be the set of terms of TF whose only constants are those listed above. We shall define functions

$$\begin{aligned}
[\![\,]\!]_{\mathcal{T}} &: \mathcal{T} \to \mathcal{E} \\
[\![\,]\!]_{\mathcal{F}} &: \mathcal{F} \to \mathcal{E} \\
[\![\,]\!]_{\mathcal{P}} &: \mathcal{P} \to \mathcal{E}
\end{aligned}$$

that take each term, formula or proof expression to its (intended) representative in TF.

We shall assume that the set of base TF variables $\mathcal{V}^{\mathbf{0}}$ is the union of the set of individual variables and the set of proof variables from predicate logic.

Terms

$$\begin{aligned}
[\![x]\!] &\equiv x \\
[\![f(t_1, \ldots, t_n)]\!] &\equiv \overline{f}[[\![t_1]\!], \ldots, [\![t_n]\!]]
\end{aligned}$$

Formulas

$$\begin{aligned}
[\![R(t_1, \ldots, t_n)]\!] &\equiv \overline{R}[[\![t_1]\!], \ldots, [\![t_n]\!]] \\
[\![\phi \to \psi]\!] &\equiv \to[[\![\phi]\!], [\![\psi]\!]] \\
[\![\forall x \phi]\!] &\equiv \forall[[x]\, [\![\phi]\!]]
\end{aligned}$$

Proof Expressions

$$\begin{aligned}
[\![\mathrm{Ax}_n]\!] &\equiv \mathrm{Ax}_n \\
[\![p]\!] &\equiv p \\
[\![(\to I)(\phi, \psi, [p]D)]\!] &\equiv (\to I)[[\![\phi]\!], [\![\psi]\!], [p]\, [\![D]\!]] \\
[\![(\to E)(\phi, \psi, D_1, D_2)]\!] &\equiv (\to E)[[\![\phi]\!], [\![\psi]\!], [\![D_1]\!], [\![D_2]\!]] \\
[\![(\forall I)([x]\phi, [x]D)]\!] &\equiv (\forall I)[[x]\, [\![\phi]\!], [x]\, [\![D]\!]] \\
[\![(\forall E)([x]\phi, t, D)]\!] &\equiv (\forall E)[[x]\, [\![\phi]\!], [\![t]\!], [\![D]\!]]
\end{aligned}$$

Once we have introduced the rules of deduction of TF, we shall be able to prove the adequacy of this representation of predicate logic within TF, and justify our claim that the correspondence is closer than is usual between an object logic and its representation in a logical framework.

3.2.1 Instantiation

We now introduce the very important operation of *instantiation* on the terms of TF. This operation plays the role that substitution plays in most frameworks.

We are to define, for every term M, α-ary abstraction sequence $\vec{N}^\alpha$, and α-ary variable sequence $\vec{x}^\alpha$, the term

$$\{\vec{N}^\alpha / \vec{x}^\alpha\} M \,,$$

the result of *instantiating* $\vec{x}^\alpha$ with $\vec{N}^\alpha$ throughout M.

It may help, when reading the definition, to bear in mind that $\{\vec{N}^\alpha/\vec{x}^\alpha\}M$ is the normal form of the result of substituting the 'lambda abstractions' $\vec{N}$ for $\vec{x}^\alpha$ in M.

The definition is by double recursion, first on α, then on the term M.

$$
\begin{array}{rcll}
\{\vec{N}^\alpha/\vec{x}^\alpha\}z^\beta[\vec{P}^\beta] & \equiv & z^\beta[\{\vec{N}^\alpha/\vec{x}^\alpha\}\vec{P}^\beta] & \quad (\text{if } z^\beta \notin \{\vec{x}^\alpha\}) \\
\{\vec{N}^\alpha/\vec{x}^\alpha\}x_i^{\alpha_i}[\vec{P}^{\alpha_i}] & \equiv & \{\{\vec{N}^\alpha/\vec{x}^\alpha\}\vec{P}^{\alpha_i}/\vec{y}^{\alpha_i}\}N_i &
\end{array}
$$

where $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, and $\vec{N}^\alpha \equiv \langle [\vec{y}^{\alpha_1}]N_1, \ldots, [\vec{y}^{\alpha_n}]N_n \rangle$.

Note that, in the second clause in this definition, each term $\{\vec{N}^\alpha/\vec{x}^\alpha\}P_j$ has previously been defined, since P_j is a subterm of M; the term as a whole is defined, therefore, as α_i is a subarity of α. One of our principal reasons for using arities is to make this recursion succeed.

Examples In the case of base variables, instantiation reduces to the familiar operation of substitution. For the definition in this case is

$$
\begin{array}{rcl}
\{N/x^{\mathbf{0}}\}z^\beta[\vec{P}^\beta] & \equiv & z^\beta[\{N/x^{\mathbf{0}}\}\vec{P}^\beta] \\
\{N/x^{\mathbf{0}}\}x^{\mathbf{0}} & \equiv & N
\end{array}
$$

which is just the definition of substitution.

The behaviour of first-order instantiation is as follows. When a first-order variable $x^{\mathbf{n}}$ occurs, it is in the term

$$y^{\mathbf{n}}[M_1, \ldots, M_n]$$

where $M_1, \ldots, M_n$ are terms. Let us write X^* for $\{[x_1, \ldots, x_n]N/y^{\mathbf{n}}\}X$. Then the result of instantiating $y^{\mathbf{n}}$ with $[x_1, \ldots, x_n]N$ in the term above is

$$
\begin{array}{rcl}
(y^{\mathbf{n}}[M_1, \ldots, M_n])^* & \equiv & \{M_1^*/x_1, \ldots, M_n^*/x_n\}N \\
 & \equiv & [M_1^*/x_1, \ldots, M_n^*/x_n]N
\end{array}
$$

This is the operation required when, for example, instantiating a predicate variable with a given predicate in second-order logic. For example, we can deduce from

$$\forall X X t_1 \cdots t_n$$

that

$$[t_1/x_1, \ldots, t_n/x_n]\phi$$

for any formula ϕ; if we define instantiation on formulas as we did for the terms of TF, this is the formula

$$\{[x_1, \ldots, x_n]\phi/X\}X[t_1, \ldots, t_n]\,.$$

Likewise, it is the operation needed to handle the W-types of Martin-Löf Type Theory. The computation rule for W-types has the form

$$\mathbf{wrec}(\mathbf{sup}(d, e), b) = b(d, e, (x)\mathbf{wrec}(e(x), b))\,.$$

If we substitute $(y, z, u)B$ for b in this equation, we wish the right-hand side to be the result of substituting:

- d for y
- $e(M)$ for $z(M)$
- $\mathbf{wrec}(e(M), b) \equiv \mathbf{wrec}(e(M), (y, z, u)B)$ for $u(M)$

wherever these subterms occur in B. This is precisely the term given by the instantiation

$$\{d/y,\ [x]e(x)/z,\ [x]\mathbf{wrec}(e(x), (y, z, u)B)/u\}B\,.$$

When we build Martin-Löf Type Theory within TF (in the next chapter), we shall declare a third-order, $(\mathbf{0}, (\mathbf{0}, \mathbf{1}, \mathbf{1}))$-ary constant **wrec**, a second-order, $(\mathbf{0}, \mathbf{1})$-ary constant **sup**, and an equality rule that shall ensure

$$\mathbf{wrec}[\mathbf{sup}[D, [x^{\mathbf{0}}]E], [y^{\mathbf{0}}, z^{\mathbf{1}}, u^{\mathbf{1}}]B] = \{D/y^{\mathbf{0}},\ [x^{\mathbf{0}}]E[x^{\mathbf{0}}]/z^{\mathbf{1}},\ [x^{\mathbf{0}}]\mathbf{wrec}[E, [y^{\mathbf{0}}, z^{\mathbf{1}}, u^{\mathbf{1}}]B]/u^{\mathbf{1}}\}B\,.$$

We shall not need instantiation for variables of higher than first order in this thesis. They would be needed if, for example, we were to build UTT within TF.

Lemmas about Instantiation For substitution, there is a result that states that, if x is not free in M, then $[N/x]M \equiv M$. The corresponding result for instantiation is:

Lemma 3.2 *If none of $\vec{x}$ is free in the term N, then*

$$\{\vec{M}/\vec{x}\}N \equiv N\,.$$

Proof A simple induction on the term N. N cannot be an application of any of $\vec{x}$, so it must be

$$y[\vec{P}]$$

for some $y \notin \{\vec{x}\}$. Therefore,

$$\begin{aligned} \{\vec{M}/\vec{x}\}y[\vec{P}] &\equiv y[\{\vec{M}/\vec{x}\}\vec{P}] \\ &\equiv y[\vec{P}] \qquad \text{(i.h.)} \end{aligned}$$

□

The analogue for instantiation of the famous substitution lemma is the following:

Lemma 3.3 (Instantiation Lemma) *If the variable sequences $\vec{x}$ and $\vec{y}$ are disjoint, and no member of $\vec{y}$ is free in any of $\vec{M}$, then*

$$\{\vec{M}/\vec{x}\}\{\vec{N}/\vec{y}\}P \equiv \{\{\vec{M}/\vec{x}\}\vec{N}/\vec{y}\}\{\vec{M}/\vec{x}\}P$$

Proof The proof is by double induction, primarily on the *sum* of the lengths of the arities α and β, secondarily on the term P. Let the ith member of $\vec{M}$ be $[\vec{s_i}]M_i$, and the ith member of $\vec{N}$ be $[\vec{t_i}]N_i$. By α-conversion, we may assume that all the sequences $\vec{s_i}$, $\vec{t_i}$, $\vec{x}$, $\vec{y}$ are pairwise disjoint. For the purposes of this proof, we shall write θX for $\{\vec{M}/\vec{x}\}X$; thus, the result we must prove is $\theta\{\vec{N}/\vec{y}\}P \equiv \{\theta\vec{N}/\vec{y}\}\theta P$.

The cases where P is an application of none of $\vec{x}$ nor $\vec{y}$ are simple. The case where P is an application of y_i is dealt with as follows.

$$\begin{array}{rcll}
\theta\{\vec{N}/\vec{y}\}y_i[\vec{P}] & \equiv & \theta\{\{\vec{N}/\vec{y}\}\vec{P}/\vec{t_i}\}N_i & \\
& \equiv & \{\theta\{\vec{N}/\vec{y}\}\vec{P}/\vec{t_i}\}\theta N_i & \text{(i.h. on } \beta) \\
& \equiv & \{\{\theta\vec{N}/\vec{y}\}\theta\vec{P}/\vec{t_i}\}\theta N_i & \\
& & \quad \text{(i.h. on } P) & \\
& \equiv & \{\theta\vec{N}/\vec{y}\}y_i[\theta\vec{P}] & \\
& \equiv & \{\theta\vec{N}/\vec{y}\}\theta y_i[\vec{P}] &
\end{array}$$

For the final case, where $P \equiv x_i[\vec{P}]$, we have:

$$\begin{array}{rcll}
\theta\{\vec{N}/\vec{y}\}x_i[\vec{P}] & \equiv & \theta(x_i[\{\vec{N}/\vec{y}\}\vec{P}]) & \\
& \equiv & \{\theta\{\vec{N}/\vec{y}\}\vec{P}/\vec{s_i}\}M_i & \\
& \equiv & \{\{\theta\vec{N}/\vec{y}\}\theta\vec{P}/\vec{s_i}\}M_i & \text{(i.h. on } P)
\end{array}$$

while

$$\begin{array}{lll}
\{\theta\vec{N}/\vec{y}\}\theta(x_i[\vec{P}]) & & \\
\quad \equiv & \{\theta\vec{N}/\vec{y}\}\{\theta\vec{P}/\vec{s_i}\}M_i & \\
\quad \equiv & \{\{\theta\vec{N}/\vec{y}\}\theta\vec{P}/\vec{s_i}\}\{\theta\vec{N}/\vec{y}\}M_i & \text{(i.h. on } \alpha) \\
\quad \equiv & \{\{\theta\vec{N}/\vec{y}\}\theta\vec{P}/\vec{s_i}\}M_i & \text{(Lemma 3.2)}
\end{array}$$

These two terms are identical, as required. □

We are also able to prove the analogue of the result that $[x/x]M \equiv M$:

Lemma 3.4 *Let $\vec{x}$ be an α-ary variable sequence, and let $\overrightarrow{\overline{x}}$ be the sequence of η-long abstractions corresponding to $\vec{x}$. For any term M,*

$$[\overrightarrow{\overline{x}}/\vec{x}]M \equiv M$$

Proof Throughout this proof, we shall write $\overline{x}$ for the η-long abstraction corresponding to the variable x.

Let $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, and $\vec{x} \equiv (x_1^{\alpha_1}, \ldots, x_n^{\alpha_n})$. Then

$$\overline{x_i} \equiv [\vec{y_i}]x_i[\overrightarrow{\overline{y_i}}]$$

where $\vec{y_i}$ is any α_i-ary variable sequence.

The proof is by double induction on the arity α and the term M. The case where M is not an application of a member of $\vec{x}$ itself is simple. The case where

$$M \equiv x_i[\vec{P}]$$

is handled as follows:

$$\begin{array}{rcl} \{\overrightarrow{\overline{x}}/\vec{x}\}x_i[\vec{P}] & \equiv & \{\{\overrightarrow{\overline{x}}/\vec{x}\}\vec{P}/\vec{y_i}\}x_i[\overrightarrow{\overline{y_i}}] \\ & \equiv & \{\vec{P}/\vec{y_i}\}x_i[\overrightarrow{\overline{y_i}}] \qquad \text{(i.h. on } M) \end{array}$$

Now, suppose $\alpha_i \equiv (\alpha_{i1}, \ldots, \alpha_{ir_i})$. For $j = 1, \ldots, r_i$, let

$$\overline{y_{ij}} \equiv [\vec{s_{ij}}]y_{ij}[\overrightarrow{\overline{s_{ij}}}]$$

and the jth member of $\vec{P}$ be

$$[\vec{s_{ij}}]P_j$$

Then

$$\begin{array}{rcl} \{\vec{P}/\vec{y_i}\}y_{ij}[\overrightarrow{\overline{s_{ij}}}] & \equiv & \{\overrightarrow{\overline{s_{ij}}}/\vec{s_{ij}}\}P_j \\ & \equiv & P_j \qquad \text{(i.h. on } \alpha) \end{array}$$

Therefore,

$$\begin{array}{rcl} \{\overrightarrow{\overline{x}}/\vec{x}\}x_i[\vec{P}] & \equiv & \{\vec{P}/\vec{y_i}\}x_i[\overrightarrow{\overline{y_i}}] \\ & \equiv & x_i[\vec{P}] \end{array}$$

as required. $\square$

3.2.2 Kinds

A *kind*, or *base kind*, in TF is either **Type**, or a term. The types A of the object theory are represented in TF by the terms $[\![A]\!]$ of kind **Type**; the objects of type A are then represented by the terms of kind $[\![A]\!]$.

3.2.3 Signatures and Contexts

We define the set of *α-ary contexts* in TF as follows, by recursion on α. If $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, an α-ary context Δ^α is a sequence of the form

$$x_1^{\alpha_1} : (\Delta^{\alpha_1})K_1, \ldots, x_n^{\alpha_n} : (\Delta^{\alpha_n})K_n \tag{3.2}$$

where each $x_i^{\alpha_i}$ is an α_i-ary variable, all distinct, Δ^{α_i} is an α_i-ary context, and K_i is a base kind. Its *domain*, $\operatorname{dom}\Delta^\alpha$, is defined to be

$$\langle x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \rangle \, ;$$

note that this is an α-ary variable sequence. We shall often omit the operator dom; for example, we shall write $\{\vec{N}/\Delta\}M$ for $\{\vec{N}/\operatorname{dom}\Delta\}M$.

We shall sometimes refer to the expressions of the form $(\Delta)K$ as *α-ary product kinds*, and write them as K^α. Just as with abstractions, product kinds are not first-class entities of TF; only the base kinds are. For this reason, we shall avoid the phrase 'α-ary kind'; 'kind' on its own shall always mean base kind.

The elements of the context, which have the form $x^\alpha : (\Delta^\alpha)K$, shall be referred to as *α-ary variable declarations.*

In the product kind

$$(x_1 : K_1, \ldots, x_n : K_n)K$$

we take x_i to be bound within K_{i+1}, ..., K_n, and K. In particular, in (3.2) above, every member of $\operatorname{dom}\Delta^{\alpha_i}$ is bound within every term that follows it in $(\Delta^{\alpha_i})K_i$; however, the variables $x_1^{\alpha_1}$, ..., $x_n^{\alpha_n}$ themselves are not bound in this context.

When writing product kinds, we shall omit any variable that does not occur later in the kind. For example, we write

$$(A)B \text{ in place of } (x : A)B$$

if x does not occur free in B. We also concatenate repeated kinds; we write, for example,

$$(x, y : A)B \text{ in place of } (x : A, y : A)B\,.$$

We may sometimes use semicolons in place of commas within contexts when it aids readability.

A *signature* in TF consists of a sequence of

- *constant declarations*

  $$c^\alpha : (\Delta^\alpha)K$$

  where c^α is an α-ary constant, Δ^α an α-ary context, and K a kind; and

- *equation declarations*

  $$(\Delta^\alpha)(M = N : K)$$

  where Δ^α is an α-ary context, M and N terms, and K a kind.

such that no two constant declarations are of the same constant.

Example: Predicate Logic We are now able to give the signature that shall represent the noble fragment of intuitionistic predicate logic in TF. We give ourselves two base constants — 'term' and 'form' — of kind **Type** to represent the type of all terms, and the type of all formulas. We shall use variables of type 'term' as individual variables; and, if the term P represents the formula ϕ, we shall use variables of type $\mathbf{Prf}[P]$ to represent occurrence markers for ϕ.

With these conventions, for any finite set of axioms T over a finite language L, we write down the following constant declarations.

Let Σ_L denote the signature consisting of the following declarations:

$$\begin{array}{rcl} \mathrm{term} & : & \mathbf{Type} \\ \mathrm{form} & : & \mathbf{Type} \end{array}$$

For each n-ary function constant f:

$$\overline{f} : \overbrace{(\mathrm{term}, \ldots, \mathrm{term})}^{n} \mathrm{term}$$

For each n-ary predicate constant R:

$$\begin{array}{rcl} \overline{R} & : & \overbrace{(\mathrm{term}, \ldots, \mathrm{term})}^{n} \mathrm{form} \\ \rightarrow & : & (\mathrm{form}, \mathrm{form})\mathrm{form} \\ \forall & : & ((\mathrm{term})\mathrm{form})\mathrm{form} \\ \mathbf{Prf} & : & (\mathrm{form})\mathbf{Type} \\ (\rightarrow I) & : & (p, q : \mathrm{form};\ (\mathbf{Prf}[p])\,\mathbf{Prf}[q])\,\mathbf{Prf}[\rightarrow [p, q]] \\ (\rightarrow E) & : & (p, q : \mathrm{form};\ \mathbf{Prf}[\rightarrow [p, q]];\ \mathbf{Prf}[p])\,\mathbf{Prf}[q] \\ (\forall I) & : & (p : (\mathrm{term})\mathrm{form};\ (x : \mathrm{term})\mathbf{Prf}[p[x]])\,\mathbf{Prf}[\forall[p]] \\ (\forall E) & : & (p : (\mathrm{term})\mathrm{form};\ t : \mathrm{term};\ \mathbf{Prf}[\forall[p]])\,\mathbf{Prf}[p[t]] \end{array}$$

Let Σ_T consist of Σ_L followed by the following declaration for each axiom. If axiom Ax_n is the formula ϕ, we have

$$\mathrm{Ax}_n : \mathbf{Prf}[[\![\phi]\!]]$$

Observe the close correspondence between this signature and the recursive definitions of terms, formulas and proof expressions that we gave in Chapter 2.

Note that we can only represent finite languages and theories in this manner; infinite languages or theories, under the above scheme, would require an infinite signature, which is precluded in TF. There are ways around this restriction; for example, we could only ever work with a finite part of this infinite signature, moving (via the Weakening lemma to come) to a larger finite part when necessary.

It is also possible to represent axiom *schemata* directly. For example, the axiom schema for induction in Peano Arithmetic:

For every formula $\phi[x]$, the formula

$$\phi[0] \to \forall x(\phi[x] \to \phi[s(x)]) \to \forall x\phi[x] \tag{3.3}$$

is an axiom.

can be represented by the declaration:

$$\mathrm{Ax_{Ind}} : (p : (\mathrm{term})\mathrm{form})\,\mathbf{Prf}[\to[p[0],\ \to[\forall[[x]\to[p[x],\ p[s[x]]]],\ \forall[p]]]]$$

Now, given any formula $\phi[x]$, the term

$$\mathrm{Ax_{Ind}}[[x]\,[\![\phi]\!]]$$

represents the one-line proof of the formula (3.3) above.

3.3 Judgement Forms

There are four judgement forms in TF:

- Σ sig (Σ is a valid signature.)
- $\Gamma \vdash_\Sigma$ valid (Γ is a valid context under the valid signature Σ.)
- $\Gamma \vdash_\Sigma M : K$ (Under the valid signature Σ and valid context Γ, M is a term of kind K.)
- $\Gamma \vdash_\Sigma M = N : K$ (Under the valid signature Σ and valid context Γ, M and N are convertible terms of kind K.)

We now define several new judgement forms in terms of these primitive ones.

Firstly, while there is no need for a primitive judgement form "$\Gamma \vdash_\Sigma K$ kind" in TF, it shall nevertheless be convenient to use such judgements in our discussion. Therefore, we define

$$\Gamma \Vdash_\Sigma \mathbf{Type}\ \mathrm{kind}$$

to be the judgement

$$\Gamma \vdash_\Sigma \mathrm{valid}\,,$$

and, for M a term, we define

$$\Gamma \Vdash_\Sigma M\ \mathrm{kind}$$

to be the judgement

$$\Gamma \vdash_\Sigma M : \mathbf{Type}\,.$$

We likewise define judgement forms for equality of kinds. We define

$$\Gamma \Vdash_\Sigma \mathbf{Type} = \mathbf{Type}$$

to be

$$\Gamma \vdash_\Sigma \text{valid}\,,$$

and we define

$$\Gamma \Vdash_\Sigma M = N$$

to be

$$\Gamma \vdash_\Sigma M = N : \textbf{Type}\,.$$

We leave

$$\Gamma \Vdash_\Sigma M = \textbf{Type} \text{ and } \Gamma \Vdash_\Sigma \textbf{Type} = M$$

undefined.

We shall need a judgement form to represent the conditions under which an α-ary variable or constant z may be applied to the α-ary abstraction sequence $\vec{M}$. If z has kind

$$(\Delta)M$$

z may be applied to $\vec{M}$ iff the abstractions $\vec{M}$ have the kinds given by Δ; we say $\vec{M}$ *satisfies* Δ. More specifically, if the ith member of Δ is $(\Delta_i)K_i$, and the ith member of $\vec{M}$ is $[\vec{x}_i]M_i$, then $\vec{M}$ satisfies Δ iff, when $\vec{x}_i$ have the kinds as given by Δ_i, M_i has kind K_i. The formal definition is as follows.

Let $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$ be an arity. Let

$$\begin{aligned} \vec{M} &\equiv [\vec{x}_1]M_1, \ldots, [\vec{x}_n]M_n \\ \vec{N} &\equiv [\vec{x}_1]N_1, \ldots, [\vec{x}_n]N_n \end{aligned}$$

be α-ary abstraction sequences, and

$$\Delta \equiv y_1 : (\Delta_1)K_1, \ldots, y_n : (\Delta_n)K_n$$

be an α-ary product kind. By α-conversion, we may assume that $\vec{x}_i \equiv \mathrm{dom}\,\Delta_i$. Then we define

$$\Gamma \Vdash_\Sigma \vec{M} :: \Delta$$

("$\vec{M}$ *satisfies* Δ under Σ and Γ") to be the set of judgements

$$\begin{array}{rcl} \Gamma, \Delta_1 & \vdash_\Sigma & M_1 : K_1 \\ \Gamma, \{[\vec{x}_1]M_1/y_1\}\Delta_2 & \vdash_\Sigma & M_2 : \{[\vec{x}_1]M_1/y_1\}K_2 \\ & \vdots & \\ \Gamma, \{[\vec{x}_1]M_1/y_1, \ldots, [\vec{x}_{n-1}]M_{n-1}/y_{n-1}\}\Delta_n & \vdash_\Sigma & M_n : \{[\vec{x}_1]M_1/y_1, \ldots, [\vec{x}_{n-1}]M_{n-1}/y_{n-1}\}K_n \end{array}$$

If $n = 0$, i.e. $\alpha \equiv \mathbf{0}$, we take

$$\Gamma \Vdash_\Sigma \vec{M}^\alpha :: \Delta^\alpha$$

or

$$\Gamma \Vdash_\Sigma \langle\rangle :: \langle\rangle$$

to be the single judgement

$$\Gamma \vdash_\Sigma \text{valid}\,.$$

We define

$$\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$$

to be the set of judgements

$$\begin{array}{rcl}
\Gamma, \Delta_1 & \vdash_\Sigma & M_1 = N_1 : K_1 \\
\Gamma, \{[\vec{x}_1]M_1/y_1\}\Delta_2 & \vdash_\Sigma & M_2 = N_2 : \{[\vec{x}_1]M_1/y_1\}K_2 \\
& \vdots & \\
\Gamma, \{[\vec{x}_1]M_1/y_1, \ldots, [\vec{x}_{n-1}]M_{n-1}/y_{n-1}\}\Delta_n & \vdash_\Sigma & M_n = N_n \\
& & : \{[\vec{x}_1]M_1/y_1, \ldots, [\vec{x}_{n-1}]M_{n-1}/y_{n-1}\}K_n
\end{array}$$

Again, if $n = 0$, we take

$$\Gamma \Vdash_\Sigma \langle\rangle = \langle\rangle :: \langle\rangle$$

to be the single judgement

$$\Gamma \vdash_\Sigma \text{valid}\,.$$

Finally, we define a judgement form for the convertibility of two contexts. This form shall not be used in giving the rules of deduction of TF itself, but shall be useful in considering its metatheory. Let

$$\begin{array}{rcl}
\Delta & \equiv & x_1 : (\Delta_1)K_1, \ldots, x_n : (\Delta_n)K_n \\
\Theta & \equiv & x_1 : (\Theta_1)K'_1, \ldots, x_n : (\Theta_n)K'_n
\end{array}$$

be two α-ary contexts with the same domain (we may also assume that, for each i, Δ_i and Θ_i have the same domain). We define the set of judgements

$$\Gamma \Vdash_\Sigma \Delta = \Theta$$

as follows. The definition is by recursion on α. As usual, if $\alpha \equiv \mathbf{0}$, then $\Gamma \vdash_\Sigma \langle\rangle = \langle\rangle$ is the singleton

$$\{\Gamma \vdash_\Sigma \text{valid}\}\,.$$

Otherwise, we define

$$\begin{array}{rcl}(\Gamma \Vdash_\Sigma \Delta = \Theta) & = & \Gamma \Vdash_\Sigma \Delta_1 = \Theta_1 \\ & & \cup \{\Gamma, \Delta_1 \Vdash_\Sigma K_1 = K_1'\} \\ & & \cup\ \Gamma, x_1 : (\Delta_1)K_1 \Vdash_\Sigma \Delta_2 = \Theta_2 \\ & & \cup \{\Gamma, x_1 : (\Delta_1)K_1, \Delta_2 \Vdash_\Sigma K_2 = K_2'\} \\ & & \cup \cdots \\ & & \cup\ \Gamma, x_1 : (\Delta_1)K_1, \ldots, x_{n-1} : (\Delta_{n-1})K_{n-1} \Vdash_\Sigma \Delta_n = \Theta_n \\ & & \cup \{\Gamma, x_1 : (\Delta_1)K_1, \ldots, x_{n-1} : (\Delta_{n-1})K_{n-1}, \Delta_n \Vdash_\Sigma K_n = K_n'\}\end{array}$$

Note that, therefore, $\Gamma \Vdash_\Sigma \Delta^\alpha = \Theta^\alpha$ is only defined if, for each i, K_i and K_i' are either both **Type** or both terms.

We are now finally able to give the rules of deduction of TF. They are given in Figure 3.1.

3.4 Basic Metatheory

We can now proceed to prove the metatheoretic properties of this system. Unless otherwise stated, each lemma is proven by a simple induction on derivations.

We must temporararily work with a system in which the rule (var_eq) has a different form; we shall eventually prove the two systems equivalent. To give the new form of the rule, we must define a new judgement form.

Given an α-ary context Δ and α-ary abstraction sequences $\vec{M}$ and $\vec{N}$, the judgement form

$$\Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta$$

is defined as follows. Let

$$\begin{array}{rcl}\Delta & \equiv & x_1 : (\Delta_1)K_1, \ldots, x_n : (\Delta_n)K_n \\ \vec{M} & \equiv & [\Delta_1]M_1, \ldots, [\Delta_n]M_n \\ \vec{N} & \equiv & [\Delta_1]N_1, \ldots, [\Delta_n]N_n\end{array}$$

Then $\Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta$ is defined to be the following set of judgements:

$$\begin{array}{rcl}\Gamma, \Delta_1 & \vdash_\Sigma & N_1 : K_1 \\ \Gamma, \{[\Delta_1]M_1/x_1\}\Delta_2 & \vdash_\Sigma & N_2 : \{[\Delta_1]M_1/x_1\}K_2 \\ & \vdots & \\ \Gamma, \{[\Delta_1]M_1/x_1, \ldots, [\Delta_{n-1}]M_{n-1}/x_{n-1}\}\Delta_n & \vdash_\Sigma & N_n : \{[\Delta_1]M_1/x_1, \ldots, [\Delta_{n-1}]M_{n-1}/x_{n-1}\}K_n\end{array}$$

(This is the same as the judgement form $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$, except the terms N_i appear in the subject positions. Alternatively, it is the same as the term $\Gamma \Vdash_\Sigma \vec{N} :: \Delta$, except the elements of $\vec{M}$ are instantiated in the contexts and kinds.)

Signature Formation

$$\text{(emp\_sig)}\ \frac{}{\langle\rangle\ \text{sig}} \qquad \text{(sig)}\ \frac{\Delta \Vdash_\Sigma K\ \text{kind}}{\Sigma, c : (\Delta)K\ \text{sig}}\ (c \notin \mathrm{dom}\,\Sigma)$$

$$\text{(eq\_sig)}\ \frac{\Delta \vdash_\Sigma M : K \quad \Delta \vdash_\Sigma N : K}{\Sigma, (\Delta)(M = N : K)\ \text{sig}}$$

Context Formation

$$\text{(emp\_ctxt)} \qquad \frac{\Sigma\ \text{sig}}{\vdash_\Sigma\ \text{valid}}$$

$$\text{(ctxt)} \quad \frac{\Gamma, \Delta \Vdash_\Sigma K\ \text{kind}}{\Gamma, x : (\Delta)K \vdash_\Sigma \text{valid}} \quad (x \notin \mathrm{dom}\,\Gamma)$$

Application

$$\text{(const)} \qquad \frac{\Gamma \Vdash_\Sigma \vec{M} :: \Delta}{\Gamma \vdash_\Sigma c[\vec{M}] : \{\vec{M}/\Delta\}K} \qquad (c : (\Delta)K \in \Sigma)$$

$$\text{(const\_eq)} \quad \frac{\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta}{\Gamma \vdash_\Sigma c[\vec{M}] = c[\vec{N}] : \{\vec{M}/\Delta\}K} \quad (c : (\Delta)K \in \Sigma)$$

$$\text{(var)} \qquad \frac{\Gamma \Vdash_\Sigma \vec{M} :: \Delta}{\Gamma \vdash_\Sigma x[\vec{M}] : \{\vec{M}/\Delta\}K} \qquad (x : (\Delta)K \in \Gamma)$$

$$\text{(var\_eq)} \quad \frac{\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta}{\Gamma \vdash_\Sigma x[\vec{M}] = x[\vec{N}] : \{\vec{M}/\Delta\}K} \quad (x : (\Delta)K \in \Gamma)$$

Equality

$$\text{(eq\_use)}\ \frac{\Gamma \Vdash_\Sigma \vec{M} :: \Delta}{\Gamma \vdash_\Sigma \{\vec{M}/\Delta\}N = \{\vec{M}/\Delta\}P : \{\vec{M}/\Delta\}K}\ ((\Delta)(N = P : K) \in \Sigma)$$

$$\text{(ref)}\ \frac{\Gamma \vdash_\Sigma M : K}{\Gamma \vdash_\Sigma M = M : K} \qquad \text{(sym)}\ \frac{\Gamma \vdash_\Sigma M = N : K}{\Gamma \vdash_\Sigma N = M : K}$$

$$\text{(trans)} \quad \frac{\Gamma \vdash_\Sigma M = N : K \quad \Gamma \vdash_\Sigma N = P : K}{\Gamma \vdash_\Sigma M = P : K}$$

$$\text{(conv)} \quad \frac{\Gamma \vdash_\Sigma M : N \quad \Gamma \vdash_\Sigma N = P : \mathbf{Type}}{\Gamma \vdash_\Sigma M : P}$$

$$\text{(conv\_eq)} \quad \frac{\Gamma \vdash_\Sigma M = M' : N \quad \Gamma \vdash_\Sigma N = P : \mathbf{Type}}{\Gamma \vdash_\Sigma M = M' : P}$$

Figure 3.1: Rules of Deduction of TF

For the moment, we deal with the system in which the rule (var_eq) has the following form:

$$\text{(var\_eq)}\ \frac{\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta \quad \Gamma \Vdash_\Sigma \vec{M} :: \Delta \quad \Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta}{\Gamma \vdash_\Sigma x[\vec{M}] = x[\vec{N}] : \{\vec{M}/\Delta\}A}\ (x : (\Delta)A \in \Gamma)$$

Once we have proven Equation Validity, it shall be seen that this system is equivalent to the one presented in Figure 3.1.

Lemma 3.5 (Signature Validity) *Any derivation of* Σ_1, Σ_2 sig *or* $\Gamma \vdash_{\Sigma_1,\Sigma_2} J$ *has a subderivation of* Σ_1 sig*.*

Corollary 3.6

1. *Any derivation of* $\Sigma_1, c : (\Delta)\mathbf{Type}, \Sigma_2$ sig *or* $\Gamma \vdash_{\Sigma_1,c:(\Delta)\mathbf{Type},\Sigma_2} J$ *has a subderivation of* $\Delta \vdash_{\Sigma_1}$ valid*.*
2. *Any derivation of* $\Sigma_1, c : (\Delta)M, \Sigma_2$ sig *or* $\Gamma \vdash_{\Sigma_1,c:(\Delta)M,\Sigma_2} J$ *has a subderivation of* $\Delta \vdash_{\Sigma_1} M : \mathbf{Type}$*.*
3. *Any derivation of* $\Sigma_1, (\Delta)(M = N : K), \Sigma_2$ sig *or* $\Gamma \vdash_{\Sigma_1,(\Delta)(M=N:K),\Sigma_2} J$ *has subderivations of* $\Delta \vdash_{\Sigma_1} M : K$ *and* $\Delta \vdash_{\Sigma_1} N : K$*.*

Lemma 3.7 (Context Validity) *Any derivation of* $\Gamma, \Delta \vdash_\Sigma J$ *has a subderivation of* $\Gamma \vdash_\Sigma$ valid*.*

Corollary 3.8

1. *Any derivation of* $\Gamma_1, x : (\Delta)\mathbf{Type}, \Gamma_2 \vdash_\Sigma J$ *has a subderivation of* $\Gamma_1, \Delta \vdash_\Sigma$ valid*.*
2. *Any derivation of* $\Gamma_1, x : (\Delta)M, \Gamma_2 \vdash_\Sigma J$ *has a subderivation of* $\Gamma_1, \Delta \vdash_\Sigma M : \mathbf{Type}$*.*

Definition 3.9 (Consistent) *We shall call a context*

$$x_1 : (\Delta_1)K_1, \ldots, x_n : (\Delta_n)K_n$$

consistent *iff every free variable in* $(\Delta_i)K_i$ *is one of* $\{x_1, \ldots, x_{i-1}\}$*. (Recall that we consider the members of* $\mathrm{dom}\,\Delta_i$ *to be bound within* $(\Delta_i)K_i$*.)*

A signature Σ *is* consistent *iff it has no free variables, and the only constants occuring in any declaration* δ *of* Σ *are declared in* Σ *before* δ*.*

A pair (Σ, Γ) *consisting of a signature* Σ *and context* Γ *is* consistent *iff* Σ *and* Γ *are consistent, and every constant in* Γ *is declared in* Σ*. We shall also express this by saying that* Γ *is consistent* with respect to Σ*.*

Lemma 3.10 (Contants and Free Variables)

1. *If* Σ sig*, then* Σ *is consistent.*
2. *If* $\Gamma \vdash_\Sigma J$*, then* Γ *is consistent with respect to* Σ*, every free variable in* J *is declared in* Γ*, and every constant in* J *is declared in* Σ*.*

Corollary 3.11 *Let* $\Delta \equiv \langle x_1 : (\Delta_1)K_1, \ldots, x_n : (\Delta_n)K_n\rangle$ *be a non-empty context, and* $\vec{M} \equiv \langle[\Delta_1]M_1, \ldots, [\Delta_n]M_n\rangle$, $\vec{N} \equiv \langle[\Delta_1]N_1, \ldots, [\Delta_n]N_n\rangle$ *abstraction sequences of corresponding arity. If* Δ *is consistent, then*

$$\Gamma \Vdash_\Sigma \vec{M} :: \Delta$$

is the set of judgements

$$\Gamma, \{\vec{M}/\Delta\}\Delta_i \vdash_\Sigma M_i : \{\vec{M}/\Delta\}K_i$$

for $i = 1, \ldots, n$; *and*

$$\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$$

is the set of judgements

$$\Gamma, \{\vec{M}/\Delta\}\Delta_i \vdash_\Sigma M_i = N_i : \{\vec{M}/\Delta\}K_i$$

for $i = 1, \ldots, n$.

Proof By consistency, the only members of $\mathrm{dom}\,\Delta$ free in Δ_i or K_i are $x_1, \ldots, x_{i-1}$. Therefore, by Lemma 3.2,

$$\{[\Delta_1]M_1/x_1, \ldots, [\Delta_{i-1}]M_{i-1}/x_{i-1}\}\Delta_i \equiv \{\vec{M}/\Delta\}\Delta_i$$

and similarly for K_i. $\Box$

Lemma 3.12 (Weakening)

1. *If* $\Gamma \vdash_{\Sigma_1} J$, $\Sigma_1 \subseteq \Sigma_2$, *and* Σ_2 sig, *then* $\Gamma \vdash_{\Sigma_2} J$.
2. *If* $\Gamma \vdash_\Sigma J$, $\Gamma \subseteq \Delta$, *and* $\Delta \vdash_\Sigma$ valid, *then* $\Delta \vdash_\Sigma J$.

Lemma 3.13 1. ***Instantiation***

If $\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma J$ *and* $\Gamma_1 \Vdash_\Sigma \vec{N} :: \Delta$, *then* $\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\vec{N}/\Delta\}J$.

2. ***Functionality, Preliminary Form***

If $\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma M : K$; $\Gamma_1 \Vdash_\Sigma \vec{N} = \vec{P} :: \Delta$; $\Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{N} :: \Delta$ *and* $\Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{P} :: (\vec{N})\Delta$; *then*

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\vec{N}/\Delta\}M = \{\vec{P}/\Delta\}M : \{\vec{N}/\Delta\}K\ .$$

Proof Let α be the common arity of Δ, $\vec{N}$ and $\vec{P}$. We shall assume, as a global induction hypothesis, that both parts hold for all subarities of α, and prove first Instantiation then Functionality for α. Each part is proven by a local induction on the derivation of the first listed premise.

1. **Instantiation** We deal here with the case of the (var_eq) rule where the variable concerned is declared in Δ:

$$\frac{\Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{M_1} = \vec{M_2} :: \Theta \quad \Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{M_1} :: \Theta \quad \Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{M_2} :: (\vec{M_1})\Theta}{\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma x[\vec{M_1}] = x[\vec{M_2}] : \{\vec{M_1}/\Theta\}K}\ (x : (\Theta)K \in \Delta)$$

Let the corresponding element of $\vec{N}$ be $[\Theta]N$.

We must show

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\{\vec{N}/\Delta\}\vec{M}_1/\Theta\}N = \{\{\vec{N}/\Delta\}\vec{M}_2/\Theta\}N : \{\{\vec{N}/\Delta\}\vec{M}_1/\Theta\}\{\vec{N}/\Delta\}K \ .$$

By the local induction hypothesis,

$$\begin{aligned}
\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 &\Vdash_\Sigma \{\vec{N}/\Delta\}\vec{M}_1 = \{\vec{N}/\Delta\}\vec{M}_2 :: \{\vec{N}/\Delta\}\Theta \\
\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 &\Vdash_\Sigma \{\vec{N}/\Delta\}\vec{M}_1 :: \{\vec{N}/\Delta\}\Theta \\
\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 &\Vdash_\Sigma \{\vec{N}/\Delta\}\vec{M}_2 :: (\{\vec{N}/\Delta\}M_1)\{\vec{N}/\Delta\}\Theta
\end{aligned}$$

Applying the Functionality part of the global induction hypothesis to $\{\vec{N}/\Delta\}\Theta$, which is of a lower order than Δ, it is sufficient to prove

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2, \{\vec{N}/\Delta\}\Theta \vdash_\Sigma N : \{\vec{N}/\Delta\}K \ .$$

We are given

$$\Gamma_1, \{\vec{N}/\Delta\}\Theta \vdash_\Sigma N : \{\vec{N}/\Delta\}K \ .$$

The result follows by Weakeninig.

2. **Functionality, Preliminary Form** We deal here with the case of the (var) rule where the variable concerned is declared in Δ:

$$\frac{\Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{M} :: \Theta}{\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma x[\vec{M}] : \{\vec{M}/\Theta\}K} \ (x : (\Theta)K \in \Delta)$$

Let the corresponding elements of $\vec{N}$, $\vec{P}$ be $[\Theta]N$ and $[\Theta]P$. We must show

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}N = \{\{\vec{P}/\Delta\}\vec{M}/\Theta\}P : \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}\{\vec{N}/\Delta\}K$$

We shall show:

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}N = \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}P : \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}\{\vec{N}/\Delta\}K \tag{3.4}$$

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}P = \{\{\vec{P}/\Delta\}\vec{M}/\Theta\}P : \{\{\vec{N}/\Delta\}\vec{M}/\Theta\}\{\vec{N}/\Delta\}K \tag{3.5}$$

The local induction hypothesis gives

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \Vdash_\Sigma \{\vec{N}/\Delta\}\vec{M} = \{\vec{P}/\Delta\}\vec{M} :: \{\vec{N}/\Delta\}\Theta$$

We are given

$$\Gamma_1, \{\vec{N}/\Delta\}\Theta \vdash_\Sigma N = P : \{\vec{N}/\Delta\}K$$

Applying Instantiation to the premises $\Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{M} :: \Theta$,

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \Vdash_\Sigma \{\vec{N}/\Delta\}\vec{M} :: \{\vec{N}/\Delta\}\Theta \ .$$

It follows by Weakening and the Instantiation part of the global induction hypothesis that (3.4) holds.

We are also given

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma P : \{\vec{N}/\Delta\}K \ .$$

It follows again by Weakening and the Instantiation part of the global induction hypothesis that (3.5) holds.

□

Corollary 3.14 *If* $\Gamma, \Delta, \Theta \vdash_\Sigma \text{valid}$*,* $\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$*,* $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$*, and* $\Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta$*, then*

$$\Gamma, \Vdash_\Sigma \{\vec{M}/\Delta\}\Theta = \{\vec{N}/\Delta\}\Theta$$

Proof By induction on the arity of Θ. □

Theorem 3.15 *1.* ***Big Conversion, Preliminary Form*** *If* $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$*,* $\Gamma \Vdash_\Sigma \Delta = \Theta$*, and* $\Gamma, \Theta \vdash_\Sigma \text{valid}$*, then*

$$\Gamma \Vdash_\Sigma \vec{M} :: \Theta \ .$$

2. ***Context Replacement, Preliminary Form***

If $\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma J$*,* $\Gamma_1 \Vdash_\Sigma \Delta = \Theta$*, and* $\Gamma_1, \Theta, \Gamma_2 \vdash_\Sigma J$*, then*

$$\Gamma_1, \Theta, \Gamma_2 \vdash_\Sigma J \ .$$

Proof The two parts are proven simultaneously by induction on the common arity of Δ, Θ and $\vec{M}$.

1. **Big Conversion, Preliminary Form** If $\Delta \equiv \Theta \equiv \langle\rangle$, there is nothing to prove.

   Let

$$\begin{array}{rcl} \Delta & \equiv & \Delta_0, x : (\Phi)K \\ \Theta & \equiv & \Theta_0, x : (\Psi)K' \\ \vec{M} & \equiv & \vec{M}_0, [\Theta]M \end{array}$$

   We are given

$$\begin{array}{c} \Gamma \Vdash_\Sigma \vec{M}_0 :: \Delta_0, \qquad \Gamma, \{\vec{M}_0/\Delta_0\}\Phi \vdash_\Sigma M : \{\vec{M}_0/\Delta_0\}K \\ \Gamma \Vdash_\Sigma \Delta_0 = \Theta_0, \qquad \Gamma, \Delta_0 \Vdash_\Sigma \Phi = \Psi, \qquad \Gamma, \Delta_0, \Phi \Vdash_\Sigma K = K', \\ \Gamma, \Theta_0, \Psi \text{valid} \end{array}$$

   and must show

$$\Gamma \Vdash_\Sigma \vec{M}_0 :: \Theta_0, \qquad \Gamma, \{\vec{M}_0/\Delta_0\}\Psi \vdash_\Sigma M : \{\vec{M}_0/\Delta_0\}K' \ .$$

$\Gamma \Vdash_\Sigma \vec{M}_0 :: \Theta_0$ holds by i.h.

By Instantiation,

$$\begin{array}{rcl}\Gamma & \Vdash_\Sigma & \{\vec{M}_0/\Delta_0\}\Phi = \{\vec{M}_0/\Delta_0\}\Psi \\ \Gamma, \{\vec{M}_0/\Delta_0\}\Phi & \Vdash_\Sigma & \{\vec{M}_0/\Delta_0\}K = \{\vec{M}_0/\Delta_0\}K' \\ \Gamma, \{\vec{M}_0/\Delta_0\}\Psi & \vdash_\Sigma & \text{valid}\end{array}$$

and so

$$\Gamma, \{\vec{M}_0/\Delta_0\}\Phi \vdash_\Sigma M : \{\vec{M}_0/\Delta_0\}K' \qquad (\text{conv}) \ .$$

By the Context Replacement part of the i.h. we have

$$\Gamma, \{\vec{M}_0/\Delta_0\}\Psi \vdash_\Sigma M : \{\vec{M}_0/\Delta_0\}K'$$

as required.

2. **Context Replacement, Preliminary Form**

This part is proven by a local induction on the derivation of $\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma J$. We deal here with the case of the (var) rule where the variable involved is declared in Δ.

$$\frac{\Gamma_1, \Delta, \Gamma_2 \Vdash_\Sigma \vec{M} :: \Phi}{\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma x[\vec{M}] : \{\vec{M}/\Phi\}K} \, (x : (\Phi)K \in \Delta)$$

Let

$$\begin{array}{rcl}\Delta & \equiv & \Delta_1, x : (\Phi)K, \Delta_2 \\ \Theta & \equiv & \Theta_1, x : (\Psi)K', \Theta_2\end{array}$$

By the local induction hypothesis,

$$\Gamma_1, \Theta, \Gamma_2 \Vdash_\Sigma \vec{M} :: \Phi$$

We are given

$$\begin{array}{rr}\Gamma_1 \Vdash_\Sigma \Delta_1 = \Theta_1 & \Gamma_1, \Theta_1 \vdash_\Sigma \text{valid} \\ \Gamma_1, \Delta_1 \Vdash_\Sigma \Phi = \Psi & \Gamma_1, \Theta_1, \Psi \vdash_\Sigma \text{valid} \\ \Gamma_1, \Delta_1, \Phi \Vdash_\Sigma K = K' & \Gamma_1, \Theta_1, \Psi \Vdash_\Sigma K' \text{ kind} \\ \Gamma_1, \Delta_1, x : (\Phi)K \Vdash_\Sigma \Delta_2 = \Theta_2 & \Gamma_1, \Theta_1, x : (\Psi)K', \Theta_2 \vdash_\Sigma \text{valid}\end{array}$$

Applying the Context Replacement part of the global induction hypothesis,

$$\begin{array}{rcl}\Gamma_1, \Theta_1 & \Vdash_\Sigma & \Phi = \Psi \\ \Gamma_1, \Theta_1, \Psi & \Vdash_\Sigma & K = K'\end{array}$$

Hence, by Weakening,

$$\begin{array}{rcl}\Gamma_1, \Theta_1, \Gamma_2 & \Vdash_\Sigma & \Phi = \Psi \\ \Gamma_1, \Theta_1, \Gamma_2, \Psi & \Vdash_\Sigma & K = K'\end{array}$$

From the first of these judgements, we have

$$\begin{array}{rcll}\Gamma_1, \Theta_1, \Gamma_2 & \Vdash_\Sigma & \vec{M} :: \Psi & \text{(global i.h. 1)} \\ \therefore \Gamma_1, \Theta_1, \Gamma_2 & \vdash_\Sigma & x[\vec{M}] : \{\vec{M}/\Phi\}K' & \text{(var)}\end{array}$$

From the second, we have

$$\Gamma_1, \Theta_1, \Gamma_2 \Vdash_\Sigma \{\vec{M}/\Phi\}K = \{\vec{M}/\Phi\}K' \qquad \text{(Instantiation)}$$

The result follows by (conv).

□

We are now close to proving the following result:

Theorem 3.16 (Equation Validity) *If* $\Gamma \vdash_\Sigma M = N : K$*, then* $\Gamma \vdash_\Sigma M : K$ *and* $\Gamma \vdash_\Sigma N : K$.

Once this has been proven, we will immediately be able to show that the additional premises in the (var_eq) rule are redundant, as well as removing several hypotheses in each of the results marked "Preliminary Form" above.

To prove Equation Validity, we need the following lemma, which shows when $\Gamma \Vdash_\Sigma \vec{N} :: \Delta$ can be deduced from $\Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta$:

Lemma 3.17 *If* $\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$*,* $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$*,* $\Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta$*, and* $\Gamma, \Delta \vdash_\Sigma \text{valid}$*, then*

$$\Gamma \Vdash_\Sigma \vec{N} :: \Delta \ .$$

Proof The proof is by induction on the common arity of $\vec{M}$, $\vec{N}$ and Δ.

If this arity is $\mathbf{0}$, there is nothing to prove.

Let

$$\begin{array}{rcl}\Delta & \equiv & \Delta_0, x : (\Theta)K \\ \vec{M} & \equiv & \vec{M_0}, [\Theta]M \\ \vec{N} & \equiv & \vec{N_0}, [\Theta]N\end{array}$$

We are given

$$\begin{array}{ll} \Gamma \Vdash_\Sigma \vec{M_0} = \vec{N_0} :: \Delta_0 & \Gamma, \{\vec{M_0}/\Delta_0\}\Theta \vdash_\Sigma M = N : \{\vec{M_0}/\Delta_0\}K \\ \Gamma \Vdash_\Sigma \vec{M_0} :: \Delta_0 & \Gamma, \{\vec{M_0}/\Delta_0\}\Theta \vdash_\Sigma M : \{\vec{M_0}/\Delta_0\}K \\ \Gamma \Vdash_\Sigma \vec{N_0} :: (\vec{M_0})\Delta_0 & \Gamma, \{\vec{M_0}/\Delta_0\}\Theta \vdash_\Sigma N : \{\vec{M_0}/\Delta_0\}K \\ \Gamma, \Delta_0, \Theta \Vdash_\sigma K \text{ kind} & \end{array}$$

We must show

$$\Gamma, \{\vec{N_0}/\Delta_0\}\Theta \vdash_\Sigma N : \{\vec{N_0}/\Delta_0\}K \ .$$

By Functionality,

$$\begin{array}{rcl} \Gamma & \Vdash_\Sigma & \{\vec{M_0}/\Delta_0\}\Theta = \{\vec{N_0}/\Delta_0\}\Theta \\ \Gamma, \{\vec{M_0}/\Delta_0\}\Theta_0 & \Vdash_\Sigma & \{\vec{M_0}/\Delta_0\}K = \{\vec{N_0}/\Delta_0\}K \end{array}$$

By Instantiation,

$$\begin{array}{rcll} \Gamma, \{\vec{N_0}/\Delta_0\}\Theta & \vdash_\Sigma & \text{valid} & \\ \Gamma, \{\vec{M_0}/\Delta_0\}\Theta & \vdash_\Sigma & N : \{\vec{N_0}/\Delta_0\}K & \text{(conv)} \\ \therefore \Gamma, \{\vec{M_0}/\Delta_0\}\Theta & \vdash_\Sigma & N : \{\vec{N_0}/\Delta_0\}K & \text{(Context Replacement)} \end{array}$$

□

Proof of Equation Validity

The proof is by induction on the derivation of $\Gamma \vdash_\Sigma M = N : K$. The only non-trivial case is

$$(\text{var\_eq}) \ \frac{\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta \quad \Gamma \Vdash_\Sigma \vec{M} :: \Delta \quad \Gamma \Vdash_\Sigma \vec{N} :: (\vec{M})\Delta}{\Gamma \vdash_\Sigma x[\vec{M}] = x[\vec{N}] : \{\vec{M}/\Delta\}K} \ (x : (\Delta)K \in \Gamma)$$

The rule (var) immediately gives us

$$\Gamma \vdash_\Sigma x[\vec{M}] : \{\vec{M}/\Delta\}K \ .$$

It remains to show

$$\Gamma \vdash_\Sigma x[\vec{N}] : \{\vec{N}/\Delta\}K \ .$$

If $K \equiv \mathbf{Type}$, then $\{\vec{M}/\Delta\}K \equiv \{\vec{N}/\Delta\}K \equiv \mathbf{Type}$. Otherwise,

$$\begin{array}{llll} \Gamma, \Delta & \vdash_\Sigma & K : \mathbf{Type} & \text{(Context Validity)} \\ \therefore \Gamma & \vdash_\Sigma 7\{\vec{M}/\Delta\}K = \{\vec{N}/\Delta\}K : \mathbf{Type} & & \text{(Functionality)} \end{array}$$

In either case, it is sufficient to prove

$$\Gamma \vdash_\Sigma x[\vec{N}] : \{\vec{N}/\Delta\}K \ .$$

By the lemma,

$$\Gamma \Vdash_\Sigma \vec{N} : \Delta \ .$$

The result follows by (var). $\square$

Equation Validity shows that the two additional premises in the (var_eq) rule are redundant. Henceforth, we shall deal with the original system as presented in Figure 3.1.

Equation Validity also allows us to remove some of the hypotheses in Functionality, Big Conversion and Context Replacement:

Corollary 3.18 (Functionality) *If* $\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma M : K$ *and* $\Gamma_1 \Vdash_\Sigma \vec{N} = \vec{P} :: \Delta$*, then*

$$\Gamma_1, \{\vec{N}/\Delta\}\Gamma_2 \vdash_\Sigma \{\vec{N}/\Delta\}M = \{\vec{P}/\Delta\}M : \{\vec{N}/\Delta\}K \ .$$

Corollary 3.19 (Big Conversion)

1. *If* $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$ *and* $\Gamma \Vdash_\Sigma \Delta = \Theta$*, then* $\Gamma \Vdash_\Sigma \vec{M} :: \Theta$.
2. *If* $\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$ *and* $\Gamma \Vdash_\Sigma \Delta = \Theta$*, then* $\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Theta$.

Corollary 3.20 (Context Replacement) *If* $\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma J$ *and* $\Gamma_1 \Vdash_\Sigma \Delta = \Theta$*, then*

$$\Gamma_1, \Theta, \Gamma_2 \vdash_\Sigma J \ .$$

Theorem 3.21 (Signature Replacement)

1. *If* $\Sigma_1, c : (\Delta)K, \Sigma_2$ sig, $\Vdash_{\Sigma_1} \Delta = \Delta'$*, and* $\Delta \Vdash_{\Sigma_1} K = K'$*, then* $\Sigma_1, c : (\Delta')K', \Sigma_2$ sig.
2. *If* $\Sigma_1, (\Delta)(M = N : K), \Sigma_2$ sig, $\Vdash_\Sigma \Delta = \Delta'$*, and* $\Delta \Vdash_{\Sigma_1} K = K'$*, then* $\Sigma_1, (\Delta')(M = N : K'), \Sigma_2$ sig.
3. *If* $\Gamma \vdash_{\Sigma_1, c:(\Delta)K, \Sigma_2} J$*,* $\Vdash_{\Sigma_1} \Delta = \Delta'$*, and* $\Delta \Vdash_{\Sigma_1} K = K'$*, then* $\Gamma \vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} J$.
4. *If* $\Gamma \vdash_{\Sigma_1, (\Delta)(M=N:K), \Sigma_2} J$*,* $\Vdash_{\Sigma_1} \Delta = \Delta'$*, and* $\Delta \Vdash_{\Sigma_1} K = K'$*, then* $\Gamma \vdash_{\Sigma_1, (\Delta')(M=N:K'), \Sigma_2} J$.

Proof The four parts are proven simultaneously, by induction on the derivation of the first premise. We deal with the case (const) here. The cases (const_eq) and (eq_use) are similar, and the other cases are trivial.

$$(\text{const}) \ \frac{\Gamma \Vdash_{\Sigma_1, c:(\Delta)K, \Sigma_2} \vec{M} :: \Delta}{\Gamma \vdash_{\Sigma_1, c:(\Delta)K, \Sigma_2} x[\vec{M}] : \{\vec{M}/\Delta\}K}$$

By the induction hypothesis,

$$\Gamma \Vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} \vec{M} :: \Delta \ .$$

Weakening applied to the second premise of the theorem gives

$$\Gamma \Vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} \Delta = \Delta'$$

so Big Conversion gives

$$\Gamma \Vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} \vec{M} :: \Delta' \ .$$

(const) therefore gives

$$\Gamma \vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} c[\vec{M}] : \{\vec{M}/\Delta\}K' \ .$$

Weakening applied to the third premise of the theorem gives

$$\Gamma, \Delta \Vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} K = K'$$

and so we have

$$\begin{array}{rll} \Gamma & \Vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} & \{\vec{M}/\Delta\}K = \{\vec{M}/\Delta\}K' \qquad \text{(Instantiation)} \\ \therefore \Gamma & \vdash_{\Sigma_1, c:(\Delta')K', \Sigma_2} & c[\vec{M}] : \{\vec{M}/\Delta\}K' \qquad \text{(conv)} \end{array}$$

as required. □

Theorem 3.22 (Type Validity) *If* $\Gamma \vdash_\Sigma M : K$ *or* $\Gamma \vdash_\Sigma M = N : K$*, then* $\Gamma \Vdash_\Sigma K$ kind.

Proof Induction on the derivation of the premise. All cases are straightforward; we give only one here: the case of the rule (var) with the declaration $x : (\Delta)N$, where N is a term.

$$\text{(var)}\ \frac{\Gamma \Vdash_\Sigma \vec{M} :: \Delta}{\Gamma \vdash_\Sigma x[\vec{M}] : \{\vec{M}/\Delta\}N}\ (x : (\Delta)N \in \Gamma)$$

Let $\Gamma \equiv \Gamma_1, x : (\Delta)N, \Gamma_2$. By Context Validity,

$$\Gamma_1, \Delta \vdash_\Sigma N : \mathbf{Type}$$

and so, by Weakening,

$$\Gamma, \Delta \vdash_\Sigma N : \mathbf{Type}\,.$$

Hence, by Cut,

$$\Gamma \vdash_\Sigma \{\vec{M}/\Delta\}N : \mathbf{Type}\,,$$

as required. □

The only non-trivial equations that hold in TF are those specified by the equations in the context, as the following result shows:

Lemma 3.23 *If* Σ *does not involve any equations and* $\Gamma \vdash_\Sigma M = N : K$*, then* $M \equiv N$*.*

Proof The proof is by induction on the derivation of $\Gamma \vdash_\Sigma M = N : K$; all the cases are trivial. □

Thanks to this result, we may prove Strengthening very simply for those contexts that do not involve equations:

Theorem 3.24 (Strengthening)

1. *If* $\Gamma \vdash_{\Sigma_1, c:(\Delta)K, \Sigma_2} J$*, c does not occur in Σ_2, Γ or J, and no equations occur in Σ_1 or Σ_2, then* $\Gamma \vdash_{\Sigma_1,\Sigma_2} J$*.*

2. *If* $\Gamma_1, x : (\Delta)K, \Gamma_2 \vdash_\Sigma J$*, x does not occur free in Γ_2 or J, and no equations occur in Σ, then* $\Gamma_1, \Gamma_2 \vdash_\Sigma J$*.*

Proof A simple induction on the derivation of the premise. All cases are straightforward. In the cases (trans), (conv) and (conv_eq), the previous lemma gives us that $N \equiv P$; hence c or x does not occur free in N, and the induction hypothesis can be applied to the premises. □

Strengthening does *not* hold for signatures that involve equations. As a counterexample, let Σ be the signature

$$A : \mathbf{Type},\ B : \mathbf{Type},\ f : (A)B,\ b : B,\ b' : B,\ (x : A)(b = f[x] : B),\ (x : A)(f[x] = b' : B)$$

(In this signature, b and b' are equal provided that A is non-empty.) Then we have

$$\begin{array}{rcl} x : A & \vdash_\Sigma & b = b' : B \\ \langle\rangle & \nvdash_\Sigma & b = b' : B \end{array}$$

An even simpler counterexample — but one which seems closer to 'cheating' — is the signature

$$\Sigma' \equiv A : \mathbf{Type},\ B : \mathbf{Type},\ b : B,\ b' : B,\ (x : A)(b = b' : B)$$

Again, we have

$$\begin{array}{rcl} x : A & \vdash_{\Sigma'} & b = b' : B \\ \langle\rangle & \nvdash_{\Sigma'} & b = b' : B \end{array}$$

Lemma 3.25 (Generation)

1. *If* $\Gamma \vdash_\Sigma x[\vec{M}] : K$*, then there is a declaration*

$$x : (\Delta)K'$$

in Γ*, and we have*

$$\begin{array}{lll} \Gamma & \Vdash_\Sigma & \vec{M} :: \Delta \\ \Gamma & \Vdash_\Sigma & \{\vec{M}/\Delta\}K' = K \end{array}$$

2. *If* $\Gamma \vdash_\Sigma c[\vec{M}] : K$*, then there is a declaration*

$$c : (\Delta)K'$$

in Σ*, and we have*

$$\begin{array}{lll} \Gamma & \Vdash_\Sigma & \vec{M} :: \Delta \\ \Gamma & \Vdash_\Sigma & \{\vec{M}/\Delta\}K' = K \end{array}$$

Proof A straightforward induction on the derivation of the premise. $\square$

Corollary 3.26 (Uniqueness of Types) *If* $\Gamma \vdash_\Sigma M : K$ *and* $\Gamma \vdash_\Sigma M : K'$*, then* $\Gamma \Vdash_\Sigma K = K'$*.*

Proof If M has the form $x[\vec{N}]$, then, by Generation, there is a declaration $x : (\Delta)K^* \in \Gamma$ such that

$$\Gamma \vdash_\Sigma K = \{\vec{N}/\Delta\}K^* = K' .$$

Similarly if M has the form $c[\vec{N}]$. $\square$

3.5 Adequacy of Predicate Logic Encoding

With these results to hand, we are able to prove the representation of predicate logic that we have been building adequate, in the following sense. We first need to show how substitution behaves under the translations that we defined previously, in Section 3.2.

Lemma 3.27 *1. For any term, formula or proof expression* X*, any term* t*, and any individual variable* x*,*

$$\{[\![t]\!]/x\}\,[\![X]\!] \equiv [\![[t/x]X]\!] .$$

2. *For any proof expressions* D *and* D'*, and any occurrence marker* p*,*

$$\{[\![D]\!]/p\}\,[\![D']\!] \equiv [\![[D/p]D']\!] .$$

Proof 1 is proven by induction on X, 2 by induction on D'. No case is at all difficult, thanks to the fact that we observed previously, that base instantiation is substitution. □

We can now prove:

Theorem 3.28 *For any finite predicate logic language L, and any finite theory T over L,*

1. Σ_L sig.
2. *Let $\{x_1, \ldots, x_n\}$ be a finite set of individual variables. $[\![\,]\!]_{\mathcal{T}}$ is a bijection between the terms of L with free variables among $\{x_1, \ldots, x_n\}$, and the TF terms M such that*
$$x_1 : \text{term}, \ldots, x_n : \text{term} \vdash_{\Sigma_L} M : \text{term}$$
3. *Let $\{x_1, \ldots, x_n\}$ be a finite set of individual variables. $[\![\,]\!]_{\mathcal{F}}$ is a bijection between the formulas of L with free variables among $\{x_1, \ldots, x_n\}$, and the TF terms M such that*
$$x_1 : \text{term}, \ldots, x_n : \text{term} \vdash_{\Sigma_L} M : \text{form}$$
4. Σ_T sig.
5. *Let $\{x_1, \ldots, x_n\}$ be a finite set of individual variables, and $\{p_1, \ldots, p_m\}$ a finite set of proof variables. $[\![\,]\!]_{\mathcal{P}}$ is a bijection between those proof expressions D of the formula ψ such that the occurrence markers that occur free are among $\{p_1, \ldots, p_m\}$, p_i being a occurrence marker for ϕ_i, and such that the free individual variables in D, ϕ_1, …, ϕ_m are all among $\{x_1, \ldots, x_n\}$; and those TF terms M such that*
$$x_1 : \text{term}, \ldots, x_n : \text{term}, p_1 : \mathbf{Prf}[[\![\phi_1]\!]], \ldots, p_m : \mathbf{Prf}[[\![\phi_m]\!]] \vdash_{\Sigma_T} M : \mathbf{Prf}[[\![\psi]\!]]$$

Proof To show that Σ_L sig is a straightforward exercise in the application of the rules of deduction of TF.

Parts 2 and 3 are formally proven by two inductions. For example, for part 2, we prove by induction on t that $\vec{x} : \text{term} \vdash_{\Sigma_L} [\![t]\!] : \text{term}$, and then by induction on derivations that, if $\vec{x} : \text{term} \vdash_{\Sigma_L} M : \text{term}$, then $M \equiv [\![t]\!]$ for some term t.

However, each part is almost simple enough to be proven by inspection. Firstly, it is easy to show that every context of the form $x_1 : \text{term}, \ldots, x_n : \text{term}$ is valid under Σ_L. We then observe that the rules of deduction that result in judgements of the form
$$\vec{x} : \text{term} \vdash_{\Sigma_L} M : \text{term}$$
are the following:
$$(\text{var})\ \frac{\vec{x} : \text{term} \vdash_{\Sigma_L} \text{valid}}{\vec{x} : \text{term} \vdash_{\Sigma_L} x_i : \text{term}}$$
for each n-ary function constant f:
$$(\text{const})\ \frac{\vec{x} : \text{term} \vdash_{\Sigma_L} M_1 : \text{term} \quad \cdots \quad \vec{x} : \text{term} \vdash_{\Sigma_L} M_n : \text{term}}{\vec{x} : \text{term} \vdash_{\Sigma_L} \overline{f}[M_1, \ldots, M_n] : \text{term}}$$

and

$$\text{(conv)}\ \frac{\vec{x} : \text{term} \vdash_{\Sigma_L} M : N \quad \vec{x} : \text{term} \vdash_{\Sigma_L} N = \text{term} : \textbf{Type}}{\vec{x} : \text{term} \vdash_{\Sigma_L} M : \text{term}}$$

This last rule can only be used when $N \equiv \text{term}$, thanks to Lemma 3.23; so it is only the first two that can produce new judgements of the required form. These two rules correspond exactly to the inductive definition of terms in predicate logic.

Likewise, the rules that produce judgements of the form $\vec{x} : \text{term} \vdash_{\Sigma_L} M : \text{form}$ are: for each n-ary predicate constant R,

$$\text{(const)}\ \frac{\vec{x} : \text{term} \vdash_{\Sigma_L} M_1 : \text{term} \quad \cdots \quad \vec{x} : \text{term} \vdash_{\Sigma_L} M_n : \text{term}}{\vec{x} : \text{term} \vdash_{\Sigma_L} \overline{R}[M_1, \ldots, M_n] : \text{form}}$$

$$\text{(const)}\ \frac{\vec{x} : \text{term} \vdash_{\Sigma_L} M : \text{form} \quad \vec{x} : \text{term} \vdash_{\Sigma_L} N : \text{form}}{\vec{x} : \text{term} \vdash_{\Sigma_L} \rightarrow [M, N] : \text{form}}$$

$$\text{(const)}\ \frac{\vec{x} : \text{term}, y : \text{term} \vdash_{\Sigma_L} M : \text{form}}{\vec{x} : \text{term} \vdash_{\Sigma_L} \forall[[y]M] : \text{form}}$$

and

$$\text{(conv)}\ \frac{\vec{x} : \text{term} \vdash_{\Sigma_L} M : N \quad \vec{x} : \text{term} \vdash_{\Sigma_L} N = \text{form} : \textbf{Type}}{\vec{x} : \text{term} \vdash_{\Sigma_L} M : \text{form}}$$

Again, the last rule cannot be used non-trivially (i.e. with a conclusion different to the first premise), and the first three mirror the recursive definition of the set of formulas.

Once we have proven part 3, it is easy to show that Σ_T sig. The proof of part 5 then follows the pattern of parts 2 and 3. In particular, we observe that the rules of deduction of TF under Σ_T match the rules of deduction of the object logic much more closely than is usual for a logical framework. The rules that result in judgements of the form $\vec{x} : \text{term}, p_1 : \textbf{Prf}[M_1], \ldots, p_n : \textbf{Prf}[M_n] \vdash_{\Sigma_T} N : \textbf{Prf}[P]$ are as follows. Let $\Gamma \equiv \vec{x} : \text{term}, p_1 : \textbf{Prf}[M_1], \ldots, p_n : \textbf{Prf}[M_n]$.

$$\text{(var)}\ \frac{\Gamma \vdash_{\Sigma_T} \text{valid}}{\Gamma \vdash_{\Sigma_T} p_i : \textbf{Prf}[M_i]}$$

For each axiom ϕ with label Ax_n:

$$\text{(const)}\ \frac{\Gamma \vdash_{\Sigma_T} \text{valid}}{\Gamma \vdash_{\Sigma_T} \text{Ax}_n : \textbf{Prf}[\llbracket \phi \rrbracket]}$$

$$\text{(const)}\ \frac{\Gamma \vdash_{\Sigma_T} P : \text{form} \quad \Gamma \vdash_{\Sigma_T} Q : \text{form} \quad \Gamma, p : \textbf{Prf}[P] \vdash_{\Sigma_T} N : \textbf{Prf}[Q]}{\Gamma \vdash_{\Sigma_T} (\rightarrow I)[P, Q, [p]N] : \textbf{Prf}[\rightarrow [P, Q]]}$$

$$\text{(const)}\ \frac{\Gamma \vdash_{\Sigma_T} P : \text{form} \quad \Gamma \vdash_{\Sigma_T} Q : \text{form} \quad \Gamma \vdash_{\Sigma_T} M : \textbf{Prf}[\rightarrow [P, Q]] \quad \Gamma \vdash_{\Sigma_T} N : \textbf{Prf}[P]}{\Gamma \vdash_{\Sigma_T} (\rightarrow E)[P, Q, M, N] : \textbf{Prf}[Q]}$$

$$\text{(const)}\ \frac{\Gamma, x : \text{term} \vdash_{\Sigma_T} P : \text{form} \quad \Gamma, x : \text{term} \vdash_{\Sigma_T} N : \textbf{Prf}[P]}{\Gamma \vdash_{\Sigma_T} (\forall I)[[x]P, [x]N] : \textbf{Prf}[\forall[[x]P]]}$$

$$\text{(const)}\ \frac{\Gamma, x : \text{term} \vdash_{\Sigma_T} P : \text{form} \quad \Gamma \vdash_{\Sigma_T} M : \text{term} \quad \Gamma \vdash_{\Sigma_T} N : \textbf{Prf}[\forall[[x]P]]}{\Gamma \vdash_{\Sigma_T} (\forall E)[[x]P, M, N] : \textbf{Prf}[\{M/x\}P]}$$

$$\text{(conv)}\ \frac{\Gamma \vdash_{\Sigma_T} M : N \quad \Gamma \vdash_{\Sigma_T} N = \mathbf{Prf}[P] : \mathbf{Type}}{\Gamma \vdash_{\Sigma_T} M : \mathbf{Prf}[P]}$$

Again, the final rule is redundant, and the other rules (using our lemma about substitution) mirror the rules of deduction of predicate logic exactly. $\square$

Notice that there is a very close correspondence between the object logic and its representation in TF. This is seen in the fact that each term of the object theory is represented by a *unique* term in the framework, and by the correspondence between the definition of the theory and the rules of deduction of the framework under the signature noted in the proof.

Chapter 4

Object Logics in Weak Frameworks

We have already seen how the noble fragment of intuitionistic predicate logic can be built within TF. We now present five more example object logics, with their representations in TF, and prove these representations correct and adequate.

4.1 Propositional Logic

We build here a Hilbert-style axiomatisation of classical propositional logic, as based on the connectives $\neg$ and $\rightarrow$. It should be noted that this representation involves no variables other than base variables, and no constants higher than first-order. This is because the axiom schemata do not involve any consideration of free variables within formulas nor substitution of terms for variables, and the sole rule of deduction (modus ponens) does not involve any discharge of hypotheses nor use of eigenvariables.

4.1.1 Grammar

Fix a countably infinite set of *propositional variables.* The set of *formulas* is defined inductively as follows:

- Every propositional variable is a formula.
- If ϕ is a formula, so is $\neg\phi$.
- If ϕ and ψ are formulas, so is $\phi \rightarrow \psi$.

As usual, we shall adopt the convention that $\rightarrow$ associates to the right.

4.1.2 Rules of Deduction

The set of *provable* formulas is given by the following axioms and rule of deduction:

$$(L1)\,\frac{}{\phi \to \psi \to \phi} \qquad (L2)\,\frac{}{(\phi \to \psi \to \chi) \to (\phi \to \psi) \to \phi \to \chi} \qquad (L3)\,\frac{}{(\neg\phi \to \neg\psi) \to \psi \to \phi}$$

$$(MP)\,\frac{\phi \quad \phi \to \psi}{\psi}$$

As we did for natural deduction, we can rewrite these rules as an inductive definition of the set of *proof expressions* for each formula ϕ as follows. Assume we are given an infinite set of *occurrence markers* for each formula.

- Every occurrence marker for ϕ is a proof expression for ϕ.
- $(L1)(\phi, \psi)$ is a proof expression for $\phi \to \psi \to \phi$.
- $(L2)(\phi, \psi, \chi)$ is a proof expression for $(\phi \to \psi \to \chi) \to (\phi \to \psi) \to \phi \to \chi$.
- $(L3)(\phi, \psi)$ is a proof expression for $(\neg\phi \to \neg\psi) \to \psi \to \phi$.
- If P is a proof expression for $\phi \to \psi$, and Q is a proof expression for ϕ, then $MP(\phi, \psi, P, Q)$ is a proof expression for ψ.

4.1.3 Representation in TF

Let Σ_{prop} be the following TF signature:

$$\begin{array}{rcl}
\text{form} & : & \mathbf{Type} \\
\neg & : & (\text{form})\text{form} \\
\to & : & (\text{form}, \text{form})\text{form} \\
\mathbf{Prf} & : & (\text{form})\mathbf{Type} \\
L1 & : & (p, q : \text{form})\mathbf{Prf}[\to[p, \to[q, p]]] \\
L2 & : & (p, q, r : \text{form})\mathbf{Prf}[\to[\to[p, \to[q, r]], \to[\to[p, q], \to[p, r]]]] \\
L3 & : & (p, q : \text{form})\mathbf{Prf}[\to[\to[\neg[p], \neg[q]], \to[q, p]]] \\
MP & : & (p, q : \text{form};\ \mathbf{Prf}[p];\ \mathbf{Prf}[\to[p, q]])\ \mathbf{Prf}[q]
\end{array}$$

We define the mapping

$$\phi \mapsto [\![\phi]\!]$$

that takes each formula to a TF term under this signature as follows. We assume that the propositional variables and proof variables we have been using are all base variables of TF.

$$\begin{array}{rcl}
[\![p]\!] & \equiv & p \\
[\![\neg\phi]\!] & \equiv & \neg[[\![\phi]\!]] \\
[\![\phi \to \psi]\!] & \equiv & \to[[\![\phi]\!], [\![\psi]\!]]
\end{array}$$

We also define the mapping

$$D \mapsto [\![D]\!]$$

taking proof expressions to TF terms as follows.

$$\begin{array}{rcl} [\![p]\!] & \equiv & p \\ [\![(L1)(\phi,\psi)]\!] & \equiv & L1[[\![\phi]\!],[\![\psi]\!]] \\ [\![(L2)(\phi,\psi,\chi)]\!] & \equiv & L2[[\![\phi]\!],[\![\psi]\!],[\![\chi]\!]] \\ [\![(L3)(\phi,\psi)]\!] & \equiv & L3[[\![\phi]\!],[\![\psi]\!]] \end{array}$$

If P is a proof expression for $\phi \to \psi$, and Q a proof expression for ψ, then

$$[\![MP(\phi,\psi,P,Q)]\!] \equiv MP[[\![\phi]\!],[\![\psi]\!],[\![P]\!],[\![Q]\!]]$$

Theorem 4.1

1. Σ_{prop} sig.
2. *For any finite set of propositional variables* $\{p_1,\ldots,p_n\}$*,* $[\![\,]\!]$ *is a bijection between the formulas* ϕ *whose variables are all among* $p_1, \ldots, p_n$*, and the TF terms* M *such that*

$$p_1 : \mathrm{form}, \ldots, p_n : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} M : \mathrm{form}$$

3. *Let* $\{p_1,\ldots,p_n\}$ *be a finite set of propositional variables, and* $\{q_1,\ldots,q_m\}$ *a finite set of occurrence markers,* q_i *a occurrence marker for* ϕ_i*, say, where the variables of* ϕ_i *are all among* $p_1, \ldots, p_n$*. Let* ψ *be another formula all of whose variables are among* $p_1, \ldots, p_n$*. Then* $[\![\,]\!]$ *is a bijection between the proof expressions for* ψ *whose only occurrence markers are* $q_1, \ldots, q_m$*; and the TF terms* M *such that*

$$p_1 : \mathrm{form}, \ldots, p_n : \mathrm{form}, q_1 : \mathbf{Prf}[[\![\phi_1]\!]], \ldots, q_m : \mathbf{Prf}[[\![\phi_m]\!]] \vdash_{\Sigma_{\mathrm{prop}}} M : \mathbf{Prf}[[\![\psi]\!]]$$

Proof Again, this can almost be said to follow from an inspection of TF's rules of deduction.

That Σ_{prop} sig is derivable is perfectly straightforward to show.

Part 2 follows from the fact that the following are the only rules of deduction of TF ending in judgements of the form $\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} M : \mathrm{form}$:

$$(\mathrm{var})\,\frac{\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} \mathrm{valid}}{\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} p_i : \mathrm{form}}$$

$$(\mathrm{const})\,\frac{\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} P : \mathrm{form}}{\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} \neg[P] : \mathrm{form}}$$

$$(\mathrm{const})\,\frac{\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} P : \mathrm{form} \quad \vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} Q : \mathrm{form}}{\vec{p} : \mathrm{form} \vdash_{\Sigma_{\mathrm{prop}}} \to [P,Q] : \mathrm{form}}$$

$$(\text{conv})\ \frac{\vec{p} : \text{form} \vdash_{\Sigma_{\text{prop}}} P : M \quad \vec{p} : \text{form} \vdash_{\Sigma_{\text{prop}}} M = \text{form} : \textbf{Type}}{\vec{p} : \text{form} \vdash_{\Sigma_{\text{prop}}} P : \text{form}}$$

and, whenever (conv) is used, $M \equiv \text{form}$ by Lemma 3.23.

Part 3 follows from the fact that the following are the only rules of deduction of TF ending in judgements of the form $\Gamma \vdash_{\Sigma_{\text{prop}}} M : \textbf{Prf}[N]$, where Γ has the required form:

$$(\text{var})\ \frac{\Gamma \vdash_{\Sigma_{\text{prop}}} \text{valid}}{\Gamma \vdash_{\Sigma_{\text{prop}}} q_i : \textbf{Prf}[\llbracket\phi_i\rrbracket]}$$

$$(\text{const})\ \frac{\Gamma \vdash_{\Sigma_{\text{prop}}} P : \text{form} \quad \Gamma \vdash_{\Sigma_{\text{prop}}} Q : \text{form}}{\Gamma \vdash_{\Sigma_{\text{prop}}} L1[P,Q] : \textbf{Prf}[\to [P, \to [Q,P]]]}$$

$$(\text{const})\ \frac{\Gamma \vdash_{\Sigma_{\text{prop}}} P : \text{form} \quad \Gamma \vdash_{\Sigma_{\text{prop}}} Q : \text{form} \quad \Gamma \vdash_{\Sigma_{\text{prop}}} R : \text{form}}{\Gamma \vdash_{\Sigma_{\text{prop}}} L2[P,Q,R] : \textbf{Prf}[\to [\to [P, \to [Q,R]], \to [\to [P,Q], \to [P,R]]]]}$$

$$(\text{const})\ \frac{\Gamma \vdash_{\Sigma_{\text{prop}}} P : \text{form} \quad \Gamma \vdash_{\Sigma_{\text{prop}}} Q : \text{form}}{\Gamma \vdash_{\Sigma_{\text{prop}}} L3[P,Q] : \textbf{Prf}[\to [\to [\neg[P], \neg[Q]], \to [Q,P]]]}$$

$$(\text{const})\ \frac{\begin{array}{c}\Gamma \vdash_{\Sigma_{\text{prop}}} P : \text{form} \quad \Gamma \vdash_{\Sigma_{\text{prop}}} Q : \text{form} \\ \Gamma \vdash_{\Sigma_{\text{prop}}} M : \textbf{Prf}[\to [P,Q]] \quad \Gamma \vdash_{\Sigma_{\text{prop}}} N : \textbf{Prf}[P]\end{array}}{\Gamma \vdash_{\Sigma_{\text{prop}}} MP[P,Q,M,N] : \textbf{Prf}[Q]}$$

$$(\text{conv})\ \frac{\Gamma \vdash_{\Sigma_{\text{prop}}} M : N \quad \Gamma \vdash_{\Sigma_{\text{prop}}} N = \textbf{Prf}[P] : \textbf{Type}}{\Gamma \vdash_{\Sigma_{\text{prop}}} M : \textbf{Prf}[P]}$$

Using part 2, every context of the form Γ is valid. Further, when $\Gamma \vdash_{\Sigma_{\text{prop}}} P : \text{form}$, similar reasoning to that of part 2 shows that no q_i occurs in P, and so $P \equiv \llbracket\phi\rrbracket$ for some formula ϕ whose propositional variables are among $p_1, \ldots, p_n$. Therefore, in the case of the use of $L1$, $P \equiv \llbracket\phi\rrbracket$ and $Q \equiv \llbracket\psi\rrbracket$ for some ϕ, ψ; hence the conclusion is of the form

$$\Gamma \vdash_{\Sigma_{\text{prop}}} L1[P,Q] : \llbracket\phi \to \psi \to \phi\rrbracket\ .$$

Similar reasoning applies in the case of L2, L3 and MP. With the usual observation that Lemma 3.23 ensures that (conv) can only be used trivially, the proof is straightforward. □

4.2 Predicate Logic

We briefly describe the representation of the whole of predicate logic, not just the noble fragment, in TF. There will be no need to go into much detail, as not much is different from the previous chapter. The system with which we deal was described in Section 2.2.1.

Definition 4.2 *For a given finite language L of predicate logic, define Σ_L to be the following*

TF signature:

$$
\begin{array}{rcl}
\text{term} & : & \mathbf{Type}, \\
\overline{f} & : & \overbrace{(\text{term}, \ldots, \text{term})}^{n}\,\text{term} \quad \textit{for every } n\textit{-ary function symbol } f, \\
\text{form} & : & \mathbf{Type}, \\
\overline{P} & : & \overbrace{(\text{term}, \ldots, \text{term})}^{n}\,\text{form} \quad \textit{for every } n\textit{-ary predicate symbel } P, \\
\neg & : & (\text{form})\text{form}, \\
\rightarrow & : & (\text{form}, \text{form})\text{form}, \\
\wedge & : & (\text{form}, \text{form})\text{form}, \\
\vee & : & (\text{form}, \text{form})\text{form}, \\
\forall & : & ((\text{term})\text{form})\text{form}, \\
\exists & : & ((\text{term})\text{form})\text{form}, \\
\mathbf{Prf} & : & (\text{form})\mathbf{Type}, \\
\mathit{cont} & : & (p, q : \text{form},\ (\mathbf{Prf}[\neg[p]])\,\mathbf{Prf}[\neg[q]],\ (\mathbf{Prf}[\neg[p]])\,\mathbf{Prf}[q])\,\mathbf{Prf}[p], \\
(\rightarrow I) & : & (p, q : \text{form},\ (\mathbf{Prf}[p])\,\mathbf{Prf}[q])\,\mathbf{Prf}[\rightarrow[p, q]], \\
(\rightarrow E) & : & (p, q : \text{form},\ \mathbf{Prf}[\rightarrow[p, q]],\ \mathbf{Prf}[p])\,\mathbf{Prf}[q], \\
(\wedge I) & : & (p, q : \text{form},\ \mathbf{Prf}[p],\ \mathbf{Prf}[q])\,\mathbf{Prf}[\wedge[p, q]], \\
(\wedge El) & : & (p, q : \text{form},\ \mathbf{Prf}[\wedge[p, q]])\,\mathbf{Prf}[p], \\
(\wedge Er) & : & (p, q : \text{form},\ \mathbf{Prf}[\wedge[p, q]])\,\mathbf{Prf}[q], \\
(\vee Il) & : & (p, q : \text{form},\ \mathbf{Prf}[p])\,\mathbf{Prf}[\vee[p, q]], \\
(\vee Ir) & : & (p, q : \text{form},\ \mathbf{Prf}[q])\,\mathbf{Prf}[\vee[p, q]], \\
(\vee E) & : & (p, q, r : \text{form},\ \mathbf{Prf}[\vee[p, q]],\ (\mathbf{Prf}[p])\,\mathbf{Prf}[r],\ (\mathbf{Prf}[q])\,\mathbf{Prf}[r])\,\mathbf{Prf}[r], \\
(\forall I) & : & (p : (\text{term})\text{form},\ (x : \text{term})\,\mathbf{Prf}[p[x]])\,\mathbf{Prf}[\forall[p]], \\
(\forall E) & : & (p : (\text{term})\text{form},\ t : \text{term},\ \mathbf{Prf}[\forall[p]])\,\mathbf{Prf}[p[t]], \\
(\exists I) & : & (p : (\text{term})\text{form},\ t : \text{term},\ \mathbf{Prf}[p[t]])\,\mathbf{Prf}[\exists[p]], \\
(\exists E) & : & (p : (\text{term})\text{form},\ q : \text{form},\ \mathbf{Prf}[\exists[p]],\ (x : \text{term},\ \mathbf{Prf}[p[x]])\,\mathbf{Prf}[q])\,\mathbf{Prf}[q]
\end{array}
$$

Note the close correspondence between this signature and the definition of the predicate calculus system we gave in Section 2.2.1.

Lemma 4.3 Σ_L sig *is derivable in TF.*

Proof Straightforward. □

Definition 4.4 *Again, we assume the individual variables of predicate logic are* **0***-ary variables of TF.*

Define the TF term $[\![t]\!]$ *for each predicate logic term* t *as follows:*

$$\begin{array}{rcl} [\![x]\!] & \equiv & x \\ [\![f(t_1, \ldots, t_n)]\!] & \equiv & \overline{f}[[\![t_1]\!], \ldots, [\![t_n]\!]] \end{array}$$

Define the TF term $[\![\phi]\!]$ *for each predicate logic proposition* ϕ *as follows:*

$$\begin{array}{rcl} [\![Pt_1 \cdots t_n]\!] & \equiv & \overline{P}[[\![t_1]\!], \ldots, [\![t_n]\!]] \\ [\![\neg\phi]\!] & \equiv & \neg[[\![\phi]\!]] \\ [\![\phi \to \psi]\!] & \equiv & \to [[\![\phi]\!], [\![\psi]\!]] \\ [\![\forall x\phi]\!] & \equiv & \forall[[x]\, [\![\phi]\!]] \end{array}$$

We assume that the hypotheses labels are TF **0***-ary variables. The translation on proof expressions is as follows:*

$$\begin{array}{rcl} [\![\mathrm{Ax}_n]\!] & \equiv & \mathrm{Ax}_n \\ [\![p]\!] & \equiv & p \\ [\![(\mathit{cont})(\phi, \psi, [p]D, [p]D')]\!] & \equiv & \mathit{cont}[[\![\phi]\!], [\![\psi]\!], [p]\, [\![D]\!], [p]\, [\![D']\!]] \\ [\![(\to I)(\phi, \psi, [p]D)]\!] & \equiv & (\to I)[[\![\phi]\!], [\![\psi]\!], [p]\, [\![D]\!]] \\ [\![(\to E)(\phi, \psi, D, D')]\!] & \equiv & (\to E)[[\![\phi]\!], [\![\psi]\!], [\![D]\!], [\![D']\!]] \\ [\![(\wedge I)(\phi, \psi, D, D')]\!] & \equiv & (\wedge I)[[\![\phi]\!], [\![\psi]\!], [\![D]\!], [\![D']\!]] \\ [\![(\wedge El)(\phi, \psi, D)]\!] & \equiv & (\wedge El)[[\![\phi]\!], [\![\psi]\!], [\![D]\!]] \\ [\![(\wedge Er)(\phi, \psi, D)]\!] & \equiv & (\wedge Er)[[\![\phi]\!], [\![\psi]\!], [\![D]\!]] \\ [\![(\vee Il)(\phi, \psi, D)]\!] & \equiv & (\vee Il)[[\![\phi]\!], [\![\psi]\!], [\![D]\!]] \\ [\![(\vee Ir)(\phi, \psi, D)]\!] & \equiv & (\vee Ir)[[\![\phi]\!], [\![\psi]\!], [\![D]\!]] \\ [\![(\vee E)(\phi, \psi, \chi, D, [p]D_1, [q]D_2)]\!] & \equiv & (\vee E)[[\![\phi]\!], [\![\psi]\!], [\![\chi]\!], [\![D]\!], [p]\, [\![D_1]\!], [q]\, [\![D_2]\!]] \\ [\![(\forall I)([x]\phi, [x]D)]\!] & \equiv & (\forall I)[[x]\, [\![\phi]\!], [x]\, [\![D]\!]] \\ [\![(\forall E)([x]\phi, t, D)]\!] & \equiv & (\forall E)[[x]\, [\![\phi]\!], [\![t]\!], [\![D]\!]] \\ [\![(\exists I)([x]\phi, t, D)]\!] & \equiv & (\exists I)[[x]\, [\![\phi]\!], [\![t]\!], [\![D]\!]] \\ [\![(\exists E)([x]\phi, \psi, D, [x,p]D')]\!] & \equiv & (\exists E)[[x]\, [\![\phi]\!], [\![\psi]\!], [\![D]\!], [x,p]\, [\![D']\!]] \end{array}$$

Theorem 4.5 *1. The translation* $[\![\ ]\!]$ *is a bijection between:*

- *The terms whose variables are among* $x_1, \ldots, x_n$*, and*
- *The TF terms* M *such that*

$$x_1 : \mathrm{term}, \ldots, x_n : \mathrm{term} \vdash_{\Sigma_L} M : \mathrm{term}$$

2. The translation $[\![\ ]\!]$ *is a bijection between:*

- *The propositions whose free variables are among* $x_1, \ldots, x_n$*; and*

- *The TF terms M such that*

$$x_1 : \text{term}, \ldots, x_n : \text{term} \vdash_{\Sigma_L} M : \text{form}$$

3. *The translation $[\![\;]\!]$ is a bijection between:*

- *The proof expressions D of the proposition ψ whose free occurrence markers are among $p_1, \ldots, p_m$, p_i being a occurrence marker for ϕ_i, such that the free individual variables in D, $\phi_1, \ldots, \phi_m$ and ψ are all among $x_1, \ldots, x_n$; and*
- *The TF terms M such that*

$$\begin{array}{c} x_1 : \text{term}, \ldots, x_n : \text{term}, \\ p_1 : \mathbf{Prf}[[\![\phi_1]\!]], \ldots, p_m : \mathbf{Prf}[[\![\phi_m]\!]] \quad \vdash_{\Sigma_L} \quad M : \mathbf{Prf}[[\![\psi]\!]] \end{array}$$

Proof The proof follows our usual pattern; we shall not give the details. $\square$

4.3 Extended Natural Deduction

We are able to build Schröder-Heister's extension of natural deduction in TF with hardly any effort at all.

Let Σ be the grammatical part of the signature for predicate logic given in the previous section, from term up to $\exists$. The only addition to the signature required is then

$$\mathbf{Prf} : (\text{form})\mathbf{Type}$$

Let $\Sigma' \equiv \Sigma, \mathbf{Prf} : (\text{form})\mathbf{Type}$.

We assign *arities* to each rule R of extended natural deduction as follows. Recall that a rule of level n can be considered to be a rule that has rules of level $\leq n-1$ as premises. We assign arities by recursion on the level of the rule; the rule

$$\frac{R_1 \quad \cdots \quad R_n}{\psi} \tag{4.1}$$

is given arity

$$(\alpha_1, \ldots, \alpha_n)$$

where α_i is the arity assigned to R_i.

As in our previous examples, we assume that the occurrence markers are variables of TF; specifically, we shall assume that, if R has had arity α assigned to it, then every occurrence marker for R is an α-ary variable of TF.

We now define an α-ary product kind $[\![R]\!]$ associated with each rule R whose arity is α. The definition is as follows: the rule (4.1) above is assigned the product kind

$$(p_1 : [\![R_1]\!], \ldots, p_n : [\![R_n]\!])\,[\![\psi]\!]$$

where each p_i is a occurrence marker for R_i (and, hence, an α_i-ary variable).

The translation on proof expressions can now be defined as follows. We define a TF term $[\![D]\!]$ for each proof expression D thus:

$$[\![p[[q_{11},\ldots,q_{1k_1}]D_1,\ldots,[q_{n1},\ldots,q_{nk_n}]D_n]]\!] \equiv p[[q_{11},\ldots,q_{1k_1}]\,[\![D_1]\!]\,,\ldots,[q_{n1},\ldots,q_{nk_n}]\,[\![D_n]\!]]$$

Theorem 4.6 *Let $x_1, \ldots, x_m$ be a set of variables, and $R_1, \ldots, R_n$ be rules and ψ a formula, the only free variables in which are among $x_1, \ldots, x_m$. Let $p_1, \ldots, p_n$ be occurrence markers, p_i a occurrence marker for R_i. $[\![\,]\!]$ is a bijection between the proof expressions for ψ whose free occurrence markers are among $p_1, \ldots, p_n$, and the terms M of TF such that*

$$x_1 : \text{term}, \ldots, x_m : \text{term}, p_1 : [\![R_1]\!]\,, \ldots, p_n : [\![R_n]\!] \vdash_{\Sigma'} M : \mathbf{Prf}[[\![\psi]\!]]$$

Proof The proof follows our usual pattern. Let

$$\Gamma \equiv x_1 : \text{term}, \ldots, x_m : \text{term}, p_1 : [\![R_1]\!]\,, \ldots, p_n : [\![R_n]\!]$$

and let R_i be the rule

$$\dfrac{\begin{matrix}[S_{11}]\ \cdots\ [S_{1k_1}] \\ \vdots \\ \phi_1\end{matrix} \qquad \cdots \qquad \begin{matrix}[S_{m1}]\ \cdots\ [S_{mk_m}] \\ \vdots \\ \phi_m\end{matrix}}{\chi}$$

Then the instance of (var) corresponding to $p_i : [\![R_i]\!]$ is

$$\dfrac{\begin{array}{rcl}\Gamma, q_{11} : [\![S_{11}]\!]\,, \ldots, q_{1k_1} : [\![S_{1k_1}]\!] & \vdash_{\Sigma'} & M_1 : \mathbf{Prf}[[\![\phi_1]\!]] \\ & \vdots & \\ \Gamma, q_{m1} : [\![S_{m1}]\!]\,, \ldots, q_{mk_m} : [\![S_{mk_m}]\!] & \vdash_{\Sigma'} & M_m : \mathbf{Prf}[[\![\phi_m]\!]]\end{array}}{\Gamma \vdash_{\Sigma'} p_i[[q_{11},\ldots,q_{1k_1}]M_1,\ldots,[q_{m1},\ldots,q_{mk_m}]M_m] : \mathbf{Prf}[[\![\chi]\!]]}$$

which matches the inductive definition of the set of proof expressions. The only other rule that can form judgements of the form $\Gamma \vdash_{\Sigma'} M : \mathbf{Prf}[N]$ is (conv), which cannot be used non-trivially. Thus, the usual trivial induction suffices to establish the result. □

4.4 Martin-Löf Type Theory

We have now given four examples of systems of logic being built within TF. However, TF was originally intended for the representation of type theories, and it is time we turned to some examples of this form. We shall show how Martin-Löf Type Theory, as presented in [NPS90], can be built within TF.

It is worthwhile first establishing our terminology. Two of the systems presented in [NPS90] are called the 'theory of sets' and the 'theory of types' (the other two being the Theory of Arities and the subset theory). The meaning of the phrase 'type theory' has become rather more general in the years since, and the phrase 'logical framework' has come to describe the function of a system such as the theory of types. We shall follow the modern terminology. For us:

- The system called the theory of sets in [NPS90] shall be called by us Martin-Löf Type Theory. We shall write A type for the judgement form there written A set.
- The system called the theory of types in [NPS90] shall be called by us Martin-Löf's Logical Framework. We shall write **Type** for the constant there written **Set**, and A kind for the judgement form there written A type.

Rather than dealing with every type constructor introduced in [NPS90], we shall here deal with the fragment of Martin-Löf Type Theory that includes only Π-types, intensional equality types, $\mathbb{N}$, Σ-types, the first universe U, and W-types.

We must also assume that all our λ-abstractions, applications, and every other variety of term carry full typing information with them. For example, the rules for forming λ-abstractions and applications have the form

$$\frac{A \text{ type} \quad B \text{ type } [x : A] \quad b : B\ [x : A]}{\lambda(A, [x]B, [x]b) : \Pi x : A.B} \qquad \frac{b : \Pi x : A.B \quad a : A}{\mathbf{apply}(A, [x]B, b, a) : [a/x]B}$$

A similar assumption must be made when building Martin-Löf Type Theory within Martin-Löf's Logical Framework; see Chapter 20 of [NPS90].

Let Σ_{ML} be the following signature:

$$\begin{array}{rcl}
\Pi & : & (a : \mathbf{Type}, (a)\mathbf{Type})\mathbf{Type}, \\
\lambda & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, (x : a)b[x])\Pi[a, b], \\
\mathbf{apply} & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, \Pi[a, b], x : a)B[x], \\
\multicolumn{3}{l}{(a : \mathbf{Type}, b : (a)\mathbf{Type}, f : (x : a)b[x], x : a)} \\
(\mathbf{apply}[a, b, \lambda[a, b, f], x] & = & f[x] : b[x]),
\end{array}$$

$$\begin{array}{rcl}
\mathbf{Id} & : & (a : \mathbf{Type}; a; a)\mathbf{Type}, \\
\mathbf{id} & : & (a : \mathbf{Type}; x : a)\mathbf{Id}[a, x, x], \\
\mathbf{idpeel} & : & (a : \mathbf{Type}; x, y : a; e : \mathbf{Id}[a, x, y]; \\
& & c : (x, y : a; \mathbf{Id}[a, x, y])\mathbf{Type}; \\
& & d : (x : a)c[x, x, \mathbf{id}[a, x]]) \\
& & \quad c[x, y, e], \\
\multicolumn{3}{l}{(a : \mathbf{Type}; c : (x, y : a; \mathbf{Id}[a, x, y])\mathbf{Type}; d : (x : a)c[x, x, \mathbf{id}[a, x]]; x : a)} \\
(\mathbf{idpeel}[a, x, x, \mathbf{id}[a, x], c, d] & = & d[x] : c[x, x, \mathbf{id}[a, x]]),
\end{array}$$

$$
\begin{array}{rcl}
\mathbb{N} & : & \mathbf{Type}, \\
0 & : & \mathbb{N}, \\
\mathbf{succ} & : & (\mathbb{N})\mathbb{N}, \\
\mathbf{natrec} & : & (c : (\mathbb{N})\mathbf{Type}; a : \mathbb{N}; c[0]; (x : \mathbb{N}; c[x])c[\mathbf{succ}[x]])c[a], \\
\multicolumn{3}{l}{(c : (\mathbb{N})\mathbf{Type}, d : c[0], e : (x : \mathbb{N}; c[x])c[\mathbf{succ}[x]])} \\
(\mathbf{natrec}[c, 0, d, e] & = & d : c[0]), \\
\multicolumn{3}{l}{(c : (\mathbb{N})\mathbf{Type}, d : c[0], e : (x : \mathbb{N}; c[x])c[\mathbf{succ}[x]]; n : \mathbb{N})} \\
(\mathbf{natrec}[c, \mathbf{succ}[n], d, e] & = & e[n, \mathbf{natrec}[c, n, d, e]] : c[\mathbf{succ}[n]],
\end{array}
$$

$$
\begin{array}{rcl}
\Sigma & : & (a : \mathbf{Type}, (a)\mathbf{Type})\mathbf{Type}, \\
\mathbf{pair} & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, x : a, b[x])\Sigma[a, b], \\
\mathbf{split} & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, c : (\Sigma[a, b])\mathbf{Type}, \\
 & & p : \Sigma[a, b], d : (x : a, y : b[x])c[\mathbf{pair}[a, b, x, y]])c[p], \\
\multicolumn{3}{l}{(a : \mathbf{Type}, b : (a)\mathbf{Type}, c : (\Sigma[a, b])\mathbf{Type}, d : (x : a, y : b[x])c[\mathbf{pair}[a, b, x, y]], x : a, y : b[x])} \\
(\mathbf{split}[a, b, c, \mathbf{pair}[a, b, x, y], d] & = & d[x, y] : c[\mathbf{pair}[a, b, x, y]]),
\end{array}
$$

$$
\begin{array}{rcl}
W & : & (a : \mathbf{Type}, (a)\mathbf{Type})\mathbf{Type}, \\
\mathbf{sup} & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, x : a, (b[x])W[a, b])W[a, b], \\
\mathbf{wrec} & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, c : (W[a, b])\mathbf{Type}, x : W[a, b], \\
 & & \quad f : (y : a, z : (b[y])w[a, b], \\
 & & \qquad (x : b[y])c[z[x]])c[\mathbf{sup}[a, b, y, z]])c[x], \\
\multicolumn{3}{l}{(a : \mathbf{Type}, b : (a)\mathbf{Type}, c : (W[a, b])\mathbf{Type},} \\
\multicolumn{3}{l}{\quad f : (y : a, z : (b[y])w[a, b], (x : b[y])c[z[x]])c[\mathbf{sup}[a, b, y, z]], d : a, e : (b[d])W[a, b])} \\
(\mathbf{wrec}[a, b, c, \mathbf{sup}[a, b, d, e], f] & = & f[d, e, [x]\mathbf{wrec}[a, b, c, e[x], f]] : c[\mathbf{sup}[a, b, d, e]]),
\end{array}
$$

$$
\begin{array}{rcl}
U & : & \mathbf{Type}, \\
\mathbf{Set} & : & (U)\mathbf{Type}, \\
\hat{\mathbb{N}} & : & U, \\
\mathbf{Set}[\hat{\mathbb{N}}] & = & \mathbb{N} : \mathbf{Type}, \\
\hat{\Pi} & : & (a : U; (\mathbf{Set}[a])U)U, \\
\multicolumn{3}{l}{(a : U, b : (\mathbf{Set}[a])U)} \\
(\mathbf{Set}[\hat{\Pi}[a, b]] & = & \Pi[\mathbf{Set}[a], [x]\mathbf{Set}[b[x]]] : \mathbf{Type}), \\
\hat{\Sigma} & : & (a : U; (\mathbf{Set}[a])U)U, \\
\multicolumn{3}{l}{(a : U, b : (\mathbf{Set}[a])U)} \\
(\mathbf{Set}[\hat{\Sigma}[a, b]] & = & \Sigma[\mathbf{Set}[a], [x]\mathbf{Set}[b[x]]] : \mathbf{Type}), \\
\hat{W} & : & (a : U; (\mathbf{Set}[a])U)U, \\
\multicolumn{3}{l}{(a : U, b : (\mathbf{Set}[a])U)} \\
(\mathbf{Set}[\hat{W}[a, b]] & = & W[\mathbf{Set}[a], [x]\mathbf{Set}[b[x]]] : \mathbf{Type})
\end{array}
$$

Note our first use of a second-order variable, f, of arity $(\mathbf{0}, \mathbf{1}, \mathbf{1})$, in the declaration of **wrec**.

Lemma 4.7 Σ_{ML} sig

Proof Straightforward. $\square$

It is a difficult task to translate the whole of the of the Theory of Arities into the grammar of TF; for a discussion of this point, see the upcoming Section 4.4.1. Fortunately, Martin-Löf's Type Theory uses only a small part of the Theory of Arities, and this part fits into TF very neatly.

For each arity α of TF, we define the arity $\langle\!|\alpha|\!\rangle$ of Martin-Löf's Theory of Arities as follows:

$$\begin{array}{rcl} \langle\!|\mathbf{0}|\!\rangle & \equiv & \mathbf{0} \\ \langle\!|(\alpha_1, \ldots, \alpha_n)|\!\rangle & \equiv & \langle\!|\alpha_1|\!\rangle \otimes \cdots \otimes \langle\!|\alpha_n|\!\rangle \twoheadrightarrow \mathbf{0} \end{array}$$

We observe that every constant in Martin-Löf's Type Theory has an arity of the form $\langle\!|\alpha|\!\rangle$ for some α (and, in fact, that α is never higher than third-order). We also observe that there is no need to use any variable except those of arity $\langle\!|\alpha|\!\rangle$ for some α of first order or lower.

Thus, every term or type of Martin-Löf's Type Theory is of one of the following forms:

- A variable of arity $\mathbf{0}$.
- $x(a_1, \ldots, a_n)$, where x is a variable of arity $\overbrace{\mathbf{0} \otimes \cdots \otimes \mathbf{0}}^{n} \twoheadrightarrow \mathbf{0}$, and $a_1, \ldots, a_n$ are terms.
- $c((x_1)a_1, \ldots, (x_n)a_n)$, where c is a constant of one of the above arities, $a_1, \ldots, a_n$ are expressions (terms or types as appropriate), and each x_i is a variable of the appropriate arity (often a product arity).

We may thus translate the terms and types of Martin-Löf's Type Theory into terms of TF as follows. We shall identify the variables and constants of arity $\langle\!|\alpha|\!\rangle$ in the Theory of Arities with a subset α-ary variables and constants of TF. We also associate with each variable in the Theory of Arities of arity

$$\langle\!|\alpha_1|\!\rangle \otimes \cdots \otimes \langle\!|\alpha_n|\!\rangle$$

a sequence $x_1, \ldots, x_n$ of variables, x_i being an α_i-ary variable of TF.

We now define, for each term or type e of Martin-Löf's Type Theory, the term $[\![e]\!]$ of TF as follows:

- For x a variable of arity $\mathbf{0}$, $[\![x]\!] \equiv x$.
- For y a variable of arity $\overbrace{\mathbf{0} \otimes \cdots \otimes \mathbf{0}}^{n} \twoheadrightarrow \mathbf{0}$,

$$[\![y(e_1, \ldots, e_n)]\!] \equiv y[[\![e_1]\!], \ldots, [\![e_n]\!]]$$

$$
\begin{aligned}
[\![\Pi x : A.B]\!] &\equiv \Pi[[\![A]\!], [x]\,[\![B]\!]] \\
[\![\lambda(A,(x)B,(x)b)]\!] &\equiv \lambda[[\![A]\!], [x]\,[\![B]\!], [x]\,[\![b]\!]] \\
[\![\mathbf{apply}(A,(x)B,f,a)]\!] &\equiv \mathbf{apply}[[\![A]\!], [x]\,[\![B]\!], [\![f]\!], [\![a]\!]] \\
\\
[\![\mathbf{Id}(A,a,b)]\!] &\equiv \mathbf{Id}[[\![A]\!], [\![a]\!], [\![b]\!]] \\
[\![\mathbf{id}(A,a)]\!] &\equiv \mathbf{id}[[\![A]\!], [\![a]\!]] \\
[\![\mathbf{idpeel}(A,a,b,c,(x)C,(y)d)]\!] &\equiv \mathbf{idpeel}[[\![A]\!], [\![a]\!], [\![b]\!], [\![c]\!], \\
&\qquad [x_1,x_2,x_3]\,[\![[(x_1,x_2,x_3)/x]C]\!], [y]\,[\![d]\!]] \\
\\
[\![\mathbb{N}]\!] &\equiv \mathbb{N} \\
[\![0]\!] &\equiv 0 \\
[\![\mathbf{succ}(a)]\!] &\equiv \mathbf{succ}[[\![a]\!]] \\
[\![\mathbf{natrec}((x)C,a,d,(y)e)]\!] &\equiv \mathbf{natrec}[[x]\,[\![C]\!], [\![a]\!], [\![d]\!], [y_1,y_2]\,[\![[(y_1,y_2)/y]e]\!]] \\
\\
[\![\Sigma x : A.B]\!] &\equiv \Sigma[[\![A]\!], [x]\,[\![B]\!]] \\
[\![\mathbf{pair}(A,(x)B,a,b)]\!] &\equiv \mathbf{pair}[[\![A]\!], [x]\,[\![B]\!], [\![a]\!], [\![b]\!]] \\
[\![\mathbf{split}(A,(x)B,(v)C,c,(z)d)]\!] &\equiv \mathbf{split}[[\![A]\!], [x]\,[\![B]\!], [v]\,[\![C]\!], [\![c]\!], [z_1,z_2]\,[\![[(z_1,z_2)/z]d]\!]] \\
\\
[\![W x : A.B]\!] &\equiv W[[\![A]\!], [x]\,[\![B]\!]] \\
[\![\mathbf{sup}(A,(x)B,a,(x)b)]\!] &\equiv \mathbf{sup}[[\![A]\!], [x]\,[\![B]\!], [\![a]\!], [x]\,[\![b]\!]] \\
[\![\mathbf{wrec}(A,(x)B,(v)C,a,(z)c)]\!] &\equiv \mathbf{wrec}[[\![A]\!], [x]\,[\![B]\!], [v]\,[\![C]\!], [\![a]\!], [z_1,z_2,z_3]\,[\![[(z_1,z_2,z_3)/z]c]\!]] \\
\\
[\![U]\!] &\equiv U \\
[\![\mathbf{Set}(a)]\!] &\equiv \mathbf{Set}[[\![a]\!]] \\
[\![\hat{\mathbb{N}}]\!] &\equiv \hat{\mathbb{N}} \\
[\![\hat{\Pi}(a,(x)b)]\!] &\equiv \hat{\Pi}[[\![a]\!], [x]\,[\![b]\!]] \\
[\![\hat{\Sigma}(a,(x)b)]\!] &\equiv \hat{\Sigma}[[\![a]\!], [x]\,[\![b]\!]] \\
[\![\hat{W}(a,(x)b)]\!] &\equiv \hat{W}[[\![a]\!], [x]\,[\![b]\!]]
\end{aligned}
$$

We can also define, for each context Γ of Martin-Löf Type Theory, the TF context $[\![\Gamma]\!]$ as follows.

$$
\begin{aligned}
[\![\langle\rangle]\!] &\equiv \langle\rangle \\
[\![\Gamma, x \in A\;[y_1 \in A_1, \ldots, y_n \in A_n]]\!] &\equiv [\![\Gamma]\!], x : (y_1 : [\![A_1]\!], \ldots, y_n : [\![A_1]\!])\,[\![A]\!]
\end{aligned}
$$

We first need a lemma showing how substitution behaves:

Lemma 4.8 *1. For any base variable x and terms or types X, Y of Martin-Löf Type Theory,*

$$\{[\![X]\!]/x\}[\![Y]\!] \equiv [\![[X/x]Y]\!] \ .$$

2. For any $\mathbf{n}$-ary variable x, base variables $y_1, \ldots, y_n$, and terms or types X, Y of Martin-Löf Type Theory,

$$\{[y_1, \ldots, y_n][\![[(y_1, \ldots, y_n)/y]X]\!]/x\}[\![Y]\!] \equiv [\![[(y)X/x]Y]\!] \ .$$

Proof Each part is proven by induction on the expression Y. We deal here with the case in part 2 where Y is an application of the $\mathbf{n}$-ary variable x, which has arity $\overbrace{\mathbf{0} \otimes \cdots \otimes \mathbf{0}}^{n} \twoheadrightarrow \mathbf{0}$ as a variable in the Theory of Arities:

Let $Y \equiv x(M_1, \ldots, M_n)$, and let us write Z^* for $\{[\vec{y}][\![[(\vec{y})/y]X]\!]/x\}Z$ Then

$$\begin{array}{rcll} [\![Y]\!]^* & \equiv & (x[[\![M_1]\!], \ldots, [\![M_n]\!]])^* & \\ & \equiv & \{[\![M_1]\!]^*/y_1, \ldots, [\![M_n]\!]^*/y_n\}[\![[(\vec{y})/y]X]\!] & \\ & \equiv & \{[\![[(y)X/x]M_1]\!]/y_1, \ldots, [\![[(y)X/x]M_n]\!]/y_n\}[\![[(\vec{y})/y]X]\!] & \text{(i.h.)} \\ & \equiv & [\![[[(y)X/x](M_1, \ldots, M_n)/y]X]\!] \quad \text{(part 1)} & \\ & \equiv & [\![[(y)X/x]x(M_1, \ldots, M_n)]\!] & \end{array}$$

□

We can now show:

Theorem 4.9 *For any context Γ of Martin-Löf Type Theory:*

- *Γ is a valid context of Martin-Löf Type Theory iff* $[\![\Gamma]\!] \vdash_{\Sigma_{ML}}$ valid.
- *$[\![\,]\!]$ is a bijection between those single, saturated expressions A (i.e. expressions of arity $\mathbf{0}$) such that*

  $$A \text{ type } [\Gamma]$$

  and those terms M of TF such that

  $$[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M : \mathbf{Type}$$

- *The judgement*

  $$A = B\ [\Gamma]$$

  is derivable in Martin-Löf Type Theory iff

  $$[\![\Gamma]\!] \vdash_{\Sigma_{ML}} [\![A]\!] = [\![B]\!] : \mathbf{Type}$$

- *For any type A, $[\![\,]\!]$ is a bijection between those single, saturated expressions a such that*

  $$a \in A\ [\Gamma]$$

and those terms M of TF such that

$$[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M : [\![A]\!]$$

- *The judgement*

$$a = b \in A \; [\Gamma]$$

is derivable in Martin-Löf Type Theory iff

$$[\![\Gamma]\!] \vdash_{\Sigma_{ML}} [\![a]\!] = [\![b]\!] : [\![A]\!]$$

In each case, the bijection is up to the equality relation $a \equiv b : \mathbf{0}$ between expressions of the Theory of Arities, and up to α-conversion of the terms of TF.

Proof As with our logical theories, the proof consists essentially of simply observing that the rules of deduction that result in judgements of the form

$$[\![\Gamma]\!] \vdash_{\Sigma_{ML}} J$$

mirror the rules of deduction of Martin-Löf Type Theory.

Formally, it is easily shown that $[\![\;]\!]$ is injective. We now prove the following by induction on the derivation in Martin-Löf Type Theory:

- If Γ is a valid context, then $[\![\Gamma]\!] \vdash_{\Sigma_{ML}}$ valid.
- If A type $[\Gamma]$, then $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} [\![A]\!] : \mathbf{Type}$.
- If $A = B$ $[\Gamma]$, then $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} [\![A]\!] = [\![B]\!] : \mathbf{Type}$.
- If $a : A$ $[\Gamma]$, then $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} [\![a]\!] : [\![A]\!]$.
- If $a = b : A$ $[\Gamma]$, then $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} [\![a]\!] = [\![b]\!] : [\![A]\!]$.

And we prove these statements by induction on the derivation in TF:

- If $[\![\Gamma]\!] \vdash_{\Sigma_{ML}}$ valid, then Γ is a valid context.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M : \mathbf{Type}$, then $M \equiv [\![A]\!]$ for some A such that A type $[\Gamma]$.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M = N : \mathbf{Type}$, then $M \equiv [\![A]\!]$, $N \equiv [\![B]\!]$ for some A, B such that $A = B$ $[\Gamma]$.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M : N$, then $M \equiv [\![a]\!]$, $N \equiv [\![A]\!]$ for some a, A such that $a : A$ $[\Gamma]$.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M = N : P$, then $M \equiv [\![a]\!]$, $N \equiv [\![b]\!]$, $P \equiv [\![A]\!]$ for some a, b, A such that $a = b : A$ $[\Gamma]$.

No case is difficult; we give a few here as examples.

We consider two cases for the first induction:

Case One $\Gamma, y : B\ [x_1 : A_1, \ldots, x_n : A_n]$ is a valid context.

i.e. B type $[\Gamma, x_1 : A_1, \ldots, x_n : A_n]$. By the induction hypothesis,

$$\begin{array}{rcl} [\![\Gamma]\!], x_1 : [\![A_1]\!], \ldots, x_n : [\![A_n]\!] & \vdash_{\Sigma_{ML}} & [\![B]\!] : \mathbf{Type} \\ \therefore [\![\Gamma]\!], y : (x_1 : [\![A_1]\!], \ldots, x_n : [\![A_n]\!])\, [\![B]\!] & \vdash_{\Sigma_{ML}} & \text{valid} \end{array}$$

as required.

Case Two

$$\frac{A \text{ type } [\Gamma] \quad B \text{ type } [\Gamma, x : A] \quad b : B\ [\Gamma, x : A] \quad a : A\ [\Gamma]}{\mathbf{apply}(A, (x)B, \lambda(A, (x)B, (x)b), a) = [a/x]b : [a/x]B}$$

By the induction hypothesis,

$$\begin{array}{rcl} [\![\Gamma]\!] & \vdash_{\Sigma_{ML}} & [\![A]\!] : \mathbf{Type} \\ [\![\Gamma]\!], x : [\![A]\!] & \vdash_{\Sigma_{ML}} & [\![B]\!] : \mathbf{Type} \\ [\![\Gamma]\!], x : [\![A]\!] & \vdash_{\Sigma_{ML}} & [\![b]\!] : [\![B]\!] \\ [\![\Gamma]\!] & \vdash_{\Sigma_{ML}} & [\![a]\!] : [\![A]\!] \\ & & \\ \therefore [\![\Gamma]\!] & \vdash_{\Sigma_{ML}} & \mathbf{apply}[[\![A]\!], [x]\, [\![B]\!], \lambda[[\![A]\!], [x]\, [\![B]\!], [x]\, [\![b]\!]], [\![a]\!]] \\ & & \quad = [[\![a]\!] / x]\, [\![b]\!] : [[\![a]\!] / x]\, [\![B]\!] \end{array}$$

(using the fact that base instantiation is substitution), and, by Lemma 4.8, this is the desired conclusion.

We likewise consider two cases for the second induction:

Case One

$$(\text{var})\ \frac{\begin{array}{rcl} [\![\Gamma]\!] & \vdash_{\Sigma_{ML}} & M_1 : N_1 \\ [\![\Gamma]\!] & \vdash_{\Sigma_{ML}} & M_2 : [M_1/y_1]N_1 \\ & \vdots & \\ [\![\Gamma]\!] & \vdash_{\Sigma_{ML}} & M_n : [M_1/y_1, \ldots, M_{n-1}/y_{n-1}]N_n \end{array}}{[\![\Gamma]\!] \vdash_{\Sigma_{ML}} x[M_1, \ldots, M_n] : [M_1/y_1, \ldots, M_n/y_n]P}$$

where $x : (y_1 : N_1, \ldots, y_n : N_n)P \in [\![\Gamma]\!]$.

It must be that $x : B\ [y_1 : A_1, \ldots, y_n : A_n]$ is in Γ, where $[\![A_1]\!] \equiv N_1$, …, $[\![A_n]\!] \equiv N_n$, and $[\![B]\!] \equiv P$. By the induction hypothesis, $M_1 \equiv [\![a_1]\!]$, …, $M_n \equiv [\![a_n]\!]$, where

$$\begin{array}{rcl} a_1 & : & A_1\ [\Gamma] \\ a_2 & : & [a_1/y_1]A_2\ [\Gamma] \\ & \vdots & \\ a_n & : & [a_1/y_1, \ldots, a_{n-1}/y_{n-1}]A_n\ [\Gamma] \\ & & \\ \therefore x(a_1, \ldots, a_n) & : & [a_1/y_1, \ldots, a_n/y_n]B\ [\Gamma] \end{array}$$

as required.

Case Two

$$(\text{const\_eq})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{ML}} M = M' : \mathbf{Type} \quad [\![\Gamma]\!], x : M \vdash_{\Sigma_{ML}} N = N' : \mathbf{Type}}{[\![\Gamma]\!] \vdash_{\Sigma_{ML}} \Pi[M, [x]N] = \Pi[M', [x]N'] : \mathbf{Type}}$$

By the induction hypothesis, $M \equiv [\![A]\!]$, $M' \equiv [\![A']\!]$, $N \equiv [\![B]\!]$, $N' \equiv [\![B']\!]$. where

$$\begin{array}{rcl} A & = & A'\ [\Gamma] \\ B & = & B'\ [\Gamma, x : A] \\ \\ \therefore \Pi x : A.B & = & \Pi x : A'.B'\ [\Gamma] \end{array}$$

as required.

□

If we had wished to use the eliminator **funsplit** rather than **apply** for Π-types, we would again have needed to use a second-order variable:

$$\begin{array}{rcl} \mathbf{funsplit} & : & (a : \mathbf{Type}, b : (a)\mathbf{Type}, c : (\Pi[a, b])\mathbf{Type}, f : \Pi[a, b], \\ & & \quad d : (y : (x : a)b[x])c[\lambda[a, b, y]])c[f] \\ & & (a : \mathbf{Type}, b : (a)\mathbf{Type}, c : (\Pi[a, b])\mathbf{Type}, \\ & & \quad d : (y : (x : a)b[x])c[\lambda[a, b, y]], y : (x : a)b[x]) \\ (\mathbf{funsplit}[a, b, c, \lambda[a, b, y], d] & = & d[y] : c[\lambda[a, b, y]]) \end{array}$$

We can even, if we wished, include the rule giving extensional equality:

$$\frac{M : \mathbf{Id}(A, a, b)}{a = b : A}$$

by adding the computation rule

$$(a : \mathbf{Type}, x, y : a, p : I[a, x, y])(x = y : a)$$

4.4.1 Digression — Relating the Theory of Arities and the Grammar of TF

To translate from Martin-Löf's Theory of Arities into TF, we first define mapping that takes an arity α from Martin-Löf's Theory of Arities to an arity $[\![\alpha]\!]$ of TF as follows.

$$\begin{array}{rcl} [\![\mathbf{0}]\!] & \equiv & \mathbf{1} \\ [\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!] & \equiv & [\![\alpha_1]\!]\hat{}\cdots\hat{}[\![\alpha_n]\!] \end{array}$$

If $[\![\beta]\!] \equiv \langle \beta_1, \ldots, \beta_n \rangle$, then

$$[\![\alpha \twoheadrightarrow \beta]\!] \equiv ([\![\alpha]\!]\hat{\ }\beta_1, \ldots, [\![\alpha]\!]\hat{\ }\beta_n)$$

the left-hand side being defined if and only if the right-hand side is in the last case.

Note the unexpected nature of this definition. The intention is that we shall map each expression of arity α in Martin-Löf's Theory of Arities to an $[\![\alpha]\!]$-ary abstraction *sequence*. In particular, each single, saturated expression is mapped to a $\mathbf{1}$-ary abstraction sequence, which consists of a single term. As another example,

$$[\![(\mathbf{0} \otimes \mathbf{0}) \twoheadrightarrow (\mathbf{0} \otimes \mathbf{0})]\!] \equiv (\mathbf{2}, \mathbf{2})$$

so we are mapping an expression e of arity $(\mathbf{0} \otimes \mathbf{0}) \twoheadrightarrow (\mathbf{0} \otimes \mathbf{0})$ to a pair of $\mathbf{2}$-ary functions, namely the functions that take x and y to

$$(e(x, y)).1 \text{ and } (e(x, y)).2$$

respectively.

Now, we associate with each variable x of arity α a $[\![\alpha]\!]$-ary variable sequence $\overrightarrow{\overline{x}}^{[\![\alpha]\!]}$; if $[\![\alpha]\!] \equiv (\alpha_1, \ldots, \alpha_n)$, we shall denote the elements of this sequence by

$$\overline{x}^{\alpha_1}, \ldots, \overline{x}^{\alpha_n} .$$

Likewise, we associate with each constant c of arity α a $[\![\alpha]\!]$-ary constant sequence $\overrightarrow{\overline{c}}^{[\![\alpha]\!]}$

We can now define the translation from Martin-Löf's Theory of Arities into TF. We map each expression e of Martin-Löf's Theory of Arities of arity α to the $[\![\alpha]\!]$-ary abstraction sequence

$$\overrightarrow{[\![e]\!]}^{[\![\alpha]\!]} \equiv \langle [\![e]\!]^{\alpha_1}, \ldots, [\![e]\!]^{\alpha_n} \rangle$$

as follows.

- Each variable x of arity α is mapped to the η-long form of $\overrightarrow{\overline{x}}^{[\![\alpha]\!]}$.
- Each constant c of arity α is mapped to the η-long form of $\overrightarrow{\overline{c}}^{[\![\alpha]\!]}$.
- Suppose a is of arity $\alpha \twoheadrightarrow \beta$, and b is of arity α. Let

  $$[\![\beta]\!] \equiv (\beta_1, \ldots, \beta_n) .$$

  Then $\overrightarrow{[\![a]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]}$ is an $([\![\alpha]\!]\hat{\ }\beta_1, \ldots, [\![\alpha]\!]\hat{\ }\beta_n)$-ary abstraction sequence, and $\overrightarrow{[\![b]\!]}^{[\![\alpha]\!]}$ an $[\![\alpha]\!]$-ary abstraction sequence. Let

  $$\overrightarrow{[\![a]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]} \equiv \langle [\vec{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_1}]M_1, \ldots, [\vec{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_n}]M_n \rangle$$

  Then we take $a(b)$ to the β-ary abstraction sequence

  $$\overrightarrow{[\![a(b)]\!]}^{[\![\beta]\!]} \equiv \langle [\vec{y}^{\beta_1}]\{\overrightarrow{[\![b]\!]}^{[\![\alpha]\!]}/\vec{x}^{[\![\alpha]\!]}\}M_1, \ldots, [\vec{y}^{\beta_n}]\{\overrightarrow{[\![b]\!]}^{[\![\alpha]\!]}/\vec{x}^{[\![\alpha]\!]}\}M_n \rangle$$

- Suppose b is an expression of arity β and x a variable of arity α. Let

$$[\![\beta]\!] \equiv (\beta_1, \ldots, \beta_n)$$

and

$$\overrightarrow{[\![b]\!]}^{[\![\beta]\!]} \equiv \langle [\vec{y}^{\beta_1}]M_1, \ldots, [\vec{y}^{\beta_n}]M_n \rangle$$

Then we map $(x)b$ to the $([\![\alpha]\!]\hat{}\beta_1, \ldots, [\![\alpha]\!]\hat{}\beta_n)$-ary abstraction sequence

$$\overrightarrow{[\![(x)b]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]} \equiv \langle [\overrightarrow{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_1}]M_1, \ldots, [\overrightarrow{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_n}]M_n \rangle$$

- If a_1 is an expression of arity α_1, ..., a_n an expression of arity α_n, then we simply map $a_1, \ldots, a_n$ to the concatenation of $\overrightarrow{[\![a_1]\!]}^{[\![\alpha_1]\!]}$, ..., $\overrightarrow{[\![a_n]\!]}^{[\![\alpha_n]\!]}$:

$$\overrightarrow{[\![a_1, \ldots, a_n]\!]}^{[\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!]} \equiv \overrightarrow{[\![a_1]\!]}^{[\![\alpha_1]\!]}\hat{}\cdots\hat{}\overrightarrow{[\![a_n]\!]}^{[\![\alpha_n]\!]}$$

- If a is an expression of arity $\alpha_1 \otimes \cdots \otimes \alpha_n$, $(a).i$ is mapped to the sequence consisting of the corresponding components of $\overrightarrow{[\![a]\!]}^{[\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!]}$. That is, if $[\![\alpha_i]\!]$ is of length k_i for $i = 1, \ldots, n$, then

$$\overrightarrow{[\![(a).i]\!]}^{[\![\alpha_i]\!]}$$

consists of the $k_1 + \cdots + k_{i-1} + 1$st, $k_1 + \cdots + k_{i-1} + 2$nd, ..., $k_1 + \cdots + k_{i-1} + k_i$th components of

$$\overrightarrow{[\![a]\!]}^{[\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!]}$$

Our first task is to show how the operation of substitution behaves under this translation.

Lemma 4.10 *Let a be an expression of arity α, x a variable of arity β, and b an expression of arity β. Then*

$$\overrightarrow{[\![b[x := a]]\!]}^{[\![\beta]\!]} \equiv \{\overrightarrow{[\![a]\!]}^{[\![\alpha]\!]} / \overrightarrow{x}^{[\![\alpha]\!]}\} \overrightarrow{[\![b]\!]}^{[\![\beta]\!]}$$

Proof A simple induction on the term b. The case $b \equiv x$ is handled by Lemma 3.4; all other cases are trivial. $\square$

We shall also have use of the result

Lemma 4.11 *The free variables in $\overrightarrow{[\![a]\!]}^{[\![\alpha]\!]}$ are precisely the components of the sequences $\overrightarrow{x}^{[\![\beta]\!]}$, where x is any free variable of arity β that occurs free in a.*

Proof A simple induction on a. $\square$

We are now able to check that our mapping is well-defined and injective with respect to the relation of equality defined on expressions.

Lemma 4.12 *If a and b are expressions of arity α, and $a \equiv b :: \alpha$, then*

$$\overrightarrow{[\![a]\!]}^{[\![\alpha]\!]} \equiv \overrightarrow{[\![b]\!]}^{[\![\alpha]\!]}$$

Proof The proof is by induction on the definition of the relation $a \equiv b : \alpha$. The only difficult cases are the β-rule, η-rule, and the rules Combination 2 and Selection 2.

β-rule Let x be a variable of arity α, a an expression of arity α, and b an expression of arity β. Let

$$[\![\beta]\!] \equiv (\beta_1, \ldots, \beta_n)$$

and let the ith component of $\overrightarrow{b}^{[\![\beta]\!]}$ be

$$[\vec{y}^{\beta_i}]M_i$$

Then the ith component of $\overrightarrow{[\![(x)b]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]}$ is

$$[\overrightarrow{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_i}]M_i$$

hence the ith component of $\overrightarrow{[\![((x)b)(a)]\!]}^{[\![\beta]\!]}$ is

$$[\vec{y}^{\beta_i}]\{\overrightarrow{[\![a]\!]}^{[\![\alpha]\!]}/\overrightarrow{x}^{[\![\alpha]\!]}\}M_i$$

which, by Lemma 4.10, is the same as the ith component of $\overrightarrow{[\![b[x := a]]\!]}^{[\![\beta]\!]}$.

η-rule Let b be a variable of arity $\alpha \twoheadrightarrow \beta$, and x a variable of arity α that does not occur free in b. Let $[\![\beta]\!] \equiv (\beta_1, \ldots, \beta_n)$. By Lemma 4.11, none of $\overrightarrow{x}^{[\![\alpha]\!]}$ occurs free in $\overrightarrow{[\![b]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]}$; so by α-conversion, we may assume that the ith component of $\overrightarrow{[\![b]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]}$ is

$$[\overrightarrow{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_i}]M_i$$

Then the ith component of $\overrightarrow{[\![b(x)]\!]}^{[\![\beta]\!]}$ is

$$[\vec{y}^{\beta_i}]\{\overrightarrow{x}^{[\![\alpha]\!]}/\overrightarrow{x}^{[\![\alpha]\!]}\}M_i \equiv [\vec{y}^{\beta_i}]M_i \qquad \text{(Lemma 3.4)}$$

Therefore, the ith component of $\overrightarrow{[\![(x)b(x)]\!]}^{[\![\alpha \twoheadrightarrow \beta]\!]}$ is

$$[\overrightarrow{x}^{[\![\alpha]\!]}, \vec{y}^{\beta_i}]M_i$$

Combination 2 Let e be of arity $\alpha_1 \otimes \cdots \otimes \alpha_n$, and, for $i = 1, \ldots, n$, let k_i be the length of $[\![\alpha_i]\!]$. Then

$$\overrightarrow{[\![(e).1, \ldots, (e).n]\!]}^{[\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!]}$$

consists of the first k_1 components of $\overrightarrow{[\![e]\!]}$, then the next k_2, then the next k_3, and so forth; in short, it is the same sequence as

$$\overrightarrow{[\![e]\!]}^{[\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!]}$$

Selection 2 Let a_1 be of arity α_1, ..., a_n of arity α_n. Then

$$\overrightarrow{[\![a_1, \ldots, a_n]\!]}^{[\![\alpha_1 \otimes \cdots \otimes \alpha_n]\!]}$$

is the concatenation of the sequences $\overrightarrow{[\![a_1]\!]}^{[\![\alpha_1]\!]}, \ldots, \overrightarrow{[\![a_n]\!]}^{[\![\alpha_n]\!]}$; and

$$\overrightarrow{[\![(a_1,\ldots,a_n).i]\!]}^{[\![\alpha_i]\!]}$$

is the selection of the relevant components from this concatenation. In short,

$$\overrightarrow{[\![(a_1,\ldots,a_n).i]\!]}^{[\![\alpha_i]\!]} \equiv \overrightarrow{[\![a_i]\!]}^{[\![\alpha_i]\!]}$$

□

We can attempt to define an inverse to the mapping $[\![\;]\!]$. Let α be an arity of Martin-Löf's Theory of Arities, and let

$$[\![\alpha]\!] \equiv (\alpha_1,\ldots,\alpha_n)\,.$$

We shall define two functions. The first,

$$\mathrm{APP}_i(e, \hat{}\overrightarrow{M}^{\alpha_i}) \qquad (i = 1,\ldots,n)\,,$$

takes an expression $e :: \alpha$ and an α_i-ary abstraction sequence $\hat{}\overrightarrow{M}^{\alpha_i}$, and returns an expression of arity $\mathbf{0}$ in Martin-Löf's Theory of Arities. (If the ith member of $[\![e]\!]$ is $[\vec{x}^{\alpha_i}]N$, this shall be the expression corresponding to $\{\hat{}\overrightarrow{M}^{\alpha_i}/\vec{x}^{\alpha_i}\}N$.)

The second,

$$\left\langle\!\left| \hat{}\overrightarrow{M}^{[\![\alpha]\!]} \right|\!\right\rangle^{\alpha}$$

takes an $[\![\alpha]\!]$-ary abstraction sequence, and returns an expression of arity α; this shall be the inverse to $[\![\;]\!]$ required.

The two functions are defined simultaneously by recursion, firstly on the sum of the lengths of the terms involved in the abstraction sequence, secondly on the arity α. The definition is as follows:

$\alpha \equiv \mathbf{0}$: The definition of APP in this case is simple:

$$\mathrm{APP}_1(e) \equiv e$$

$\langle\!|\;|\!\rangle^{\mathbf{0}}$ must take a term and return an expression of arity $\mathbf{0}$. The term will either be an application of a variable or a constant. If z is a variable of arity α, and $[\![\alpha]\!] \equiv (\alpha_1,\ldots,\alpha_n)$, then

$$\left\langle\!\left| \overline{z}^{\alpha_i}[\hat{}\overrightarrow{M}^{\alpha_i}] \right|\!\right\rangle \equiv \mathrm{APP}_i(x, \hat{}\overrightarrow{M}^{\alpha_i})$$

$\alpha \equiv \alpha_1 \otimes \cdots \otimes \alpha_n$: We have $[\![\alpha]\!] \equiv [\![\alpha_1]\!]\hat{}\cdots\hat{}[\![\alpha_n]\!]$. Let $[\![\alpha_1]\!]$ be of length k_1, ..., $[\![\alpha_n]\!]$ be of length k_n.

For $1 \le i \le k_1 + \cdots + k_n$, there must be some j such that

$$i = k_1 + \cdots + k_{j-1} + r, \qquad 1 \le r \le k_j$$

Let

$$\mathrm{APP}_i(e, \hat{}\overrightarrow{M}) \equiv \mathrm{APP}_r((e).i, \hat{}\overrightarrow{M})$$

An $[\![\alpha]\!]$-ary abstraction sequence $\overrightarrow{\hat{}M}^{[\![\alpha]\!]}$ must be of the form

$$\overrightarrow{\hat{}M}^{[\![\alpha]\!]} \equiv \overrightarrow{\hat{}M}^{[\![\alpha_1]\!]}\hat{}\cdots\hat{}\overrightarrow{M}^{[\![\alpha_n]\!]}$$

Let

$$\left\langle\!\left|\overrightarrow{\hat{}M}^{[\![\alpha]\!]}\right|\!\right\rangle \equiv \left\langle\!\left|\overrightarrow{\hat{}M}^{[\![\alpha_1]\!]}\right|\!\right\rangle, \ldots, \left\langle\!\left|\overrightarrow{\hat{}M}^{[\![\alpha_n]\!]}\right|\!\right\rangle$$

$\alpha \equiv \beta \twoheadrightarrow \gamma$: Let $[\![\gamma]\!] \equiv (\gamma_1, \ldots, \gamma_n)$. Then $[\![\alpha]\!] \equiv ([\![\beta]\!]\hat{}\gamma_1, \ldots, [\![\beta]\!]\hat{}\gamma_n)$.

We set

$$\mathrm{APP}_i(e, \overrightarrow{\hat{}M}^{[\![\beta]\!]}, \overrightarrow{\hat{}N}^{\gamma_i}) \equiv \mathrm{APP}_i\left(e\left(\left\langle\!\left|\overrightarrow{\hat{}M}^{[\![\beta]\!]}\right|\!\right\rangle\right), \overrightarrow{\hat{}N}^{\gamma_i}\right)$$

By α-conversion, we may assume a $[\![\alpha]\!]$-ary abstraction sequence $\overrightarrow{\hat{}M}^{[\![\alpha]\!]}$ has the form

$$\langle[\overrightarrow{\overline{x}}^{[\![\beta]\!]}, \vec{y}^{\gamma_1}]M_1, \ldots, [\overrightarrow{\overline{x}}^{[\![\beta]\!]}, \vec{y}^{\gamma_n}]M_n\rangle$$

for some variable $x :: \beta$. We set

$$\left\langle\!\left|\overrightarrow{\hat{}M}^{[\![\alpha]\!]}\right|\!\right\rangle \equiv (x)\, \langle\!\langle [\vec{y}^{\gamma_1}]M_1, \ldots, [\vec{y}^{\gamma_n}]M_n \rangle\!\rangle$$

It is possible to prove:

Lemma 4.13 *1. For any $[\![\alpha]\!]$-ary abstraction sequence $\overrightarrow{\hat{}M}^{[\![\alpha]\!]}$,*

$$\left[\!\!\left[\left\langle\!\left|\overrightarrow{\hat{}M}^{[\![\alpha]\!]}\right|\!\right\rangle\right]\!\!\right] \equiv \overrightarrow{\hat{}M}^{[\![\alpha]\!]}$$

2. If e is an expression of arity α, and $[\![e]\!] \equiv \langle[\vec{x_1}]N_1, \ldots, [\vec{x_n}]N_n\rangle$, then

$$\left[\!\!\left[\mathrm{APP}_i(e, \overrightarrow{\hat{}M}^{[\![\alpha]\!]})\right]\!\!\right] \equiv \{\overrightarrow{\hat{}M}^{[\![\alpha]\!]}/\vec{x_i}\}N_i$$

Proof The two parts are proven simultaneously by double induction, first on the sum of the lengths of the terms in $\overrightarrow{\hat{}M}$, secondly on the arity α. □

However, the converse proves very difficult to establish. We therefore state it here merely as a conjecture:

Conjecture 4.14 *For any expression $e :: \alpha$,*

$$\langle\!\langle [\![e]\!] \rangle\!\rangle \equiv e :: \alpha\,.$$

The reason for the difficulty is the fact that several different arities of the Theory of Arities are mapped onto the same TF arity. For example, both $\mathbf{0} \otimes \mathbf{0} \twoheadrightarrow \mathbf{0} \otimes \mathbf{0}$ and $(\mathbf{0} \twoheadrightarrow \mathbf{0}) \otimes (\mathbf{0} \twoheadrightarrow \mathbf{0})$ are mapped onto $(\mathbf{2}, \mathbf{2})$. In particular, every arity $\alpha_1 \otimes \cdots \otimes \alpha_n \twoheadrightarrow \beta$ is mapped onto the same arity as its Curried form, $\alpha_1 \twoheadrightarrow \cdots \twoheadrightarrow \alpha_n \twoheadrightarrow \beta$. For this reason, we had to define $\langle\!\langle\ \rangle\!\rangle$ and APP_i by induction on the arity from the Theory of Arities, rather than the TF arity as would be more natural.

4.5 The Calculus of Constructions

For our final example of an object theory, we show how the calculus of constructions (CoC) [CH88] can be built inside TF.

Since TF makes a grammatical distinction between terms and kinds, we must construct a version of CoC that does the same; that is, the universe $*$ must be made into a universe á la Tarski. We shall also make use of equality judgements in our presentation of CoC. We could equally well have used an external notion of convertibility; the two systems can be proven equivalent by the method employed in Appendix B. We must also include typing information about both the domain and codomain in the λ-abstractions, and the Π-type used in the applications.

We define the set of *terms* and *types* of CoC by the grammar

$$\begin{array}{rcl} \text{Type } A & ::= & * \mid \mathrm{El}\,(M) \mid \Pi x : A.A \\ \text{Term } M & ::= & x \mid \lambda x : A.M : A \mid \mathbf{apply}_{\Pi x:A.A}(MM) \mid \pi x : A.M \end{array}$$

x is bound in B but not A within $\Pi x : A.B$ and $\pi x : A.B$, in M and B but not A in $\lambda x : A.M : B$, and in B but not A, M or N in $\mathbf{apply}_{\Pi x:A.B}(MN)$. (If $(x : A)M : *$, $\pi x : A.M$ is the appropriate Π-type as a member of $*$; $\mathrm{El}\,(\pi x : A.M)$ is then the Π-type considered as a type.)

The judgement forms of CoC are fivefold: $\Gamma \vdash \text{valid}$, $\Gamma \vdash A \text{ type}$, $\Gamma \vdash A = B$, $\Gamma \vdash M : A$, and $\Gamma \vdash M = N : A$.

The rules of deduction of CoC are as follows:

$$\frac{}{\langle\rangle\text{valid}} \qquad \frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \text{valid}}\,(x \notin \mathrm{dom}\,\Gamma) \qquad \frac{\Gamma\text{valid}}{\Gamma \vdash * \text{ type}}$$

$$\frac{\Gamma \vdash M : *}{\Gamma \vdash \mathrm{El}\,(M) \;\text{ type}} \quad \frac{\Gamma \vdash M = N : *}{\Gamma \vdash \mathrm{El}\,(M) = \mathrm{El}\,(N)}$$

$$\frac{\Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \Pi x : A.B \text{ type}} \quad \frac{\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash B_1 = B_2}{\Gamma \vdash \Pi x : A_1.B_1 = \Pi x : A_2.B_2}$$

$$\frac{\Gamma\text{valid}}{\Gamma \vdash x : A}\,(x : A \in \Gamma)$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : B : \Pi x : A.B} \quad \frac{\begin{array}{c}\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash B_1 = B_2 \\ \Gamma, x : A_1 \vdash M_1 = M_2 : B_1\end{array}}{\Gamma \vdash \lambda x : A_1.M_1 : B_1 = \lambda x : A_2.M_2 : B_2 : \Pi x : A_1.B_1}$$

$$\frac{\Gamma \vdash M : \Pi x : A.B \quad \Gamma \vdash N : A}{\Gamma \vdash \mathbf{apply}_{\Pi x:A.B}(MN) : [N/x]B}$$

$$\frac{\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash B_1 = B_2 \quad \Gamma \vdash M_1 = M_2 : \Pi x : A_1.B_1 \quad \Gamma \vdash N_1 = N_2 : A_1}{\Gamma \vdash \mathbf{apply}_{\Pi x:A_1.B_1}(M_1 N_1) = \mathbf{apply}_{\Pi x:A_2.B_2}(M_2 N_2) : [N_1/x]B_1}$$

$$\frac{\Gamma, x : A \vdash M : *}{\Gamma \vdash \pi x : A.M : *} \qquad \frac{\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash M_1 = M_2 : *}{\Gamma \vdash \pi x : A_1.M_1 = \pi x : A_2.M_2 : *}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A = A} \qquad \frac{\Gamma \vdash A = B}{\Gamma \vdash B = A} \qquad \frac{\Gamma \vdash A = B \quad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

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

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash A = B}{\Gamma \vdash M : B} \qquad \frac{\Gamma \vdash M = N : A \quad \Gamma \vdash A = B}{\Gamma \vdash M = N : B}$$

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

$$\frac{\Gamma, x : A \vdash M : *}{\Gamma \vdash \mathrm{El}\,(\pi x : A.M) = \Pi x : A.\,\mathrm{El}\,(M)}$$

Let Σ_{CoC} be the following signature:

$$\begin{array}{rcl}
\Pi & : & (A : \mathbf{Type}, (A)\mathbf{Type})\mathbf{Type}, \\
\lambda & : & (A : \mathbf{Type}, B : (A)\mathbf{Type}, (x : A)B[x])\Pi[A, B], \\
\mathbf{apply} & : & (A : \mathbf{Type}, B : (A)\mathbf{Type}, \Pi[A, B], a : A)B[a], \\
\multicolumn{3}{c}{(A : \mathbf{Type}, B : (A)\mathbf{Type}, b : (x : A)B[x], a : A)} \\
\multicolumn{3}{c}{(\mathbf{apply}[A, B, \lambda[A, B, b], a] = b[a] : B[a])} \\
 & & \\
* & : & \mathbf{Type}, \\
\mathrm{El} & : & (*)\mathbf{Type}, \\
\pi & : & (A : \mathbf{Type}, (A)*)*, \\
\multicolumn{3}{c}{(A : \mathbf{Type}, B : (A)*)} \\
(\mathrm{El}[\pi[A, B]] & = & \Pi[A, [x : A]\mathrm{El}[B[x]]])
\end{array}$$

We define, for each CoC term of type X, the TF term $[\![X]\!]$ as follows:

$$\begin{array}{rcl}
[\![x]\!] & \equiv & x \\
[\![\lambda x : A.M : B]\!] & \equiv & \lambda[[\![A]\!], [x]\, [\![B]\!], [x]\, [\![M]\!]] \\
[\![\mathbf{apply}_{\Pi x:A.B}(MN)]\!] & \equiv & \mathbf{apply}[[\![A]\!], [x]\, [\![B]\!], [\![M]\!], [\![N]\!]] \\
[\![\pi x : A.M]\!] & \equiv & \pi[[\![A]\!], [x]\, [\![M]\!]]
\end{array}$$

$$\begin{array}{rcl}
[\![*]\!] & \equiv & * \\
[\![\mathrm{El}\,(M)]\!] & \equiv & \mathrm{El}[[\![M]\!]] \\
[\![\Pi x : A.B]\!] & \equiv & \Pi[[\![A]\!], [x]\, [\![B]\!]]
\end{array}$$

We define $[\![\Gamma]\!]$ for a consistent context Γ thus:

$$\begin{array}{rcl}
[\![\langle\rangle]\!] & \equiv & \langle\rangle \\
[\![\Gamma, x : A]\!] & \equiv & [\![\Gamma]\!], x : [\![A]\!]_\Gamma
\end{array}$$

Theorem 4.15 *1.* Σ_{CoC} sig.

2. $\Gamma \vdash$ valid *iff* $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}}$ valid.

3. $[\![\ ]\!]_\Gamma$ *is a bijection between those types B of CoC such that*

$$\Gamma \vdash B \text{ type}$$

and those term M of TF such that

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M : \mathbf{Type}$$

4. $\Gamma \vdash B = B'$ *in CoC iff*

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![B]\!] = [\![B']\!] : \mathbf{Type}$$

in TF.

5. $[\![\ ]\!]_\Gamma$ *is a bijection between those terms M of CoC such that*

$$\Gamma \vdash M : B$$

and those terms N of TF such that

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} N : [\![B]\!]$$

6. $\Gamma \vdash M = N : B$ *in CoC iff*

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![M]\!] = [\![N]\!] : [\![B]\!]$$

in TF.

Proof Again, this follows more or less immediately from an inspection of the rules of TF with conclusions of the form

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M : K$$

which are as follows:

$$(\text{emp\_ctxt})\ \frac{\Sigma_{CoC}\ \text{sig}}{\vdash_{\Sigma_{CoC}}\ \text{valid}}$$

$$(\text{ctxt})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![A]\!] : \mathbf{Type}}{[\![\Gamma]\!], x : [\![A]\!] \vdash_{\Sigma_{CoC}} \text{valid}}\ (x \notin \operatorname{dom}\Gamma)$$

$$(\text{var})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \text{valid}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} x : [\![A]\!]}\ (x : [\![A]\!] \in \Gamma)$$

$$(\text{var\_eq})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \text{valid}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} x = x : [\![A]\!]}\ (x : [\![A]\!] \in \Gamma)$$

$$(\text{const})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B : \mathbf{Type}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \Pi[A, [x]B] : \mathbf{Type}}$$

$$(\text{const\_eq})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A = A' : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B = B' : \mathbf{Type}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \Pi[A, [x]B] = \Pi[A', [x]B'] : \mathbf{Type}}$$

$$(\text{const})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} b : B}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \lambda[A, [x]B, [x]b] : \Pi[A, [x]B]}$$

$$(\text{const\_eq})\ \frac{\begin{array}{c}[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A = A' : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B = B' : \mathbf{Type} \\ [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} b = b' : B\end{array}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \lambda[A, [x]B, [x]b] = \lambda[A', [x]B', [x]b'] : \Pi[A, [x]B]}$$

$$(\text{const})\ \frac{\begin{array}{cc}[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A : \mathbf{Type} & [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B : \mathbf{Type} \\ [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} f : \Pi[A, [x]B] & [\![\Gamma]\!] \vdash_{\Sigma_{CoC}} a : A\end{array}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathbf{apply}[A, [x]B, f, a] : [a/x]B}$$

$$(\text{const\_eq})\ \frac{\begin{array}{cc}[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A = A' : \mathbf{Type} & [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B = B' : \mathbf{Type} \\ [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} f = f' : \Pi[A, [x]B] & [\![\Gamma]\!] \vdash_{\Sigma_{CoC}} a = a' : A\end{array}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathbf{apply}[A, [x]B, f, a] = \mathbf{apply}[A', [x]B', f', a'] : [a/x]B}$$

$$(\text{eq\_use})\ \frac{\begin{array}{cc}[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A : \mathbf{Type} & [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} B : \mathbf{Type} \\ [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} b : B & [\![\Gamma]\!] \vdash_{\Sigma_{CoC}} a : A\end{array}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathbf{apply}[A, [x]B, \lambda[A, [x]B, [x]b], a] = [a/x]b : [a/x]B}$$

$$(\text{const})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \text{valid}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} * : \mathbf{Type}}$$

$$(\text{const\_eq})\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \text{valid}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} * = * : \mathbf{Type}}$$

$$\text{(const)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} a : *}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathrm{El}[a] : \mathbf{Type}}$$

$$\text{(const\_eq)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} a = a' : *}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathrm{El}[a] = \mathrm{El}[a'] : \mathbf{Type}}$$

$$\text{(const)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} b : *}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \pi[A, [x]b] : *}$$

$$\text{(const\_eq)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A = A' : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} b = b' : *}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \pi[A, [x]b] = \pi[A', [x]b'] : *}$$

$$\text{(var\_use)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A : \mathbf{Type} \quad [\![\Gamma]\!], x : A \vdash_{\Sigma_{CoC}} b : *}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathrm{El}[\pi[A, [x]b]] = \Pi[A, [x]\mathrm{El}[b]] : \mathbf{Type}}$$

$$\text{(ref)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M : K}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = M : K}$$

$$\text{(sym)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = N : K}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} N = M : K}$$

$$\text{(trans)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = N : K \quad [\![\Gamma]\!] \vdash_{\Sigma_{CoC}} N = P : K}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = P : K}$$

$$\text{(conv)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M : A \quad [\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A = B : \mathbf{Type}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M : B}$$

$$\text{(conv\_eq)}\ \frac{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = N : A \quad [\![\Gamma]\!] \vdash_{\Sigma_{CoC}} A = B : \mathbf{Type}}{[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = N : B}$$

We prove:

- If Γvalid in CoC, then $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}}$ valid in TF.
- If $\Gamma \vdash A$ type in CoC, then $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![A]\!] : \mathbf{Type}$ in TF.
- If $\Gamma \vdash A = B$ in CoC, then $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![A]\!] = [\![B]\!] : \mathbf{Type}$ in TF.
- If $\Gamma \vdash M : A$ in CoC, then $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![M]\!] : [\![A]\!]$ in TF.
- If $\Gamma \vdash M = N : A$ in CoC, then $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![M]\!] = [\![N]\!] : [\![A]\!]$ in TF.

by induction on the CoC derivation. We then prove:

- If $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}}$ valid in TF, then Γvalid in CoC.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M : \mathbf{Type}$, there exists a type A such that $M \equiv [\![A]\!]$ and $\Gamma \vdash A$ type.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} M = N : \mathbf{Type}$, there exist types A and B such that $M \equiv [\![A]\!]$, $N \equiv [\![B]\!]$, and $\Gamma \vdash A = B$.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} P : [\![A]\!]$, there exists M such that $P \equiv [\![M]\!]$ and $\Gamma \vdash M : A$.
- If $[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} P = Q : [\![A]\!]$, there exist M and N such that $P \equiv [\![M]\!]$, $Q \equiv [\![N]\!]$, and $\Gamma \vdash M = N : A$.

by induction on the TF derivation, using the fact that the rules above are the only ones to produce judgements of the appropriate form. This, together with the observation that $[\![\,]\!]$ is injective, suffice to prove the theorem. $\Box$

Chapter 5

Subsystems of TF

It is notable that all our example object logics in the previous two chapters have required variables and constants of low order only. Apart from Martin-Löf Type Theory and extended natural deduction, we required only variables of first order and below, and constants of second order and below. Martin-Löf Type Theory required second-order variables and third-order constants. Only extended natural deduction required us to go beyond this. Thus, including variables and constants of all orders in TF is, for most purposes, a case of overkill.

In the spirit of keeping our frameworks as weak as possible, we can consider forming various subsystems of TF in which the orders of variables and constants are restricted. In a lambda-calculus based type system, such an undertaking might involve quite a radical reworking of the system (we might, for example, consider introducing universes, one to hold all the base types, one for the first-order types, and so forth). In TF, however, all our grammatical entities are already classified by order (thanks, ultimately, to our use of arities), and so the definition of these susbsystems is simple; we simply exclude from our grammar those variables and constants above a certain order, and remove the corresponding instances of (sig), (eq_sig), (ctxt), (var), (var_eq), (const), (const_eq), and (eq_use) from our rules of deduction. We shall see that the result is a conservative subsystem of TF; the proof is not too difficult.

There are other important restrictions we might choose to make. We note that the logical systems (propositional and predicate logic and extended natural deduction) did not require the use of *large* parameters; that is, parameters with *large kinds* of the form $(\Delta)\mathbf{Type}$, which therefore represent variable *types* (with parameters). All their parameters had *small kinds* of the form $(\Delta)M$ for some term $M : \mathbf{Type}$; such variables represent variable objects. Only when building a type theory did we require the use of large parameters; for example, the kind of the Π constant in Martin-Löf Type Theory, when written out in full, is

$$\Pi : (a : \mathbf{Type}, b : (x : a)\mathbf{Type})\mathbf{Type}$$

and here a and b are large parameters.

We may therefore consider a further choice which can be made when forming a subsystem of TF: whether or not to allow large parameters. If we choose to disallow them, again the formation of the resulting subsystem is quite easy. Recall that, for us, $\Gamma \vdash_\Sigma K$ kind is a defined judgement form; the rule (ctxt) can therefore be considered as two rules:

$$(\mathrm{var}_s)\,\frac{\Gamma, \Delta \vdash_\Sigma M : \mathbf{Type}}{\Gamma, x : (\Delta)M \vdash_\Sigma \text{valid}}\,(x \notin \mathrm{dom}\,\Gamma) \qquad (\mathrm{var}_l)\,\frac{\Gamma, \Delta \vdash_\Sigma \text{valid}}{\Gamma, x : (\Delta)\mathbf{Type} \vdash_\Sigma \text{valid}}\,(x \notin \mathrm{dom}\,\Gamma)$$

If we wish to disallow large parameters, we simply remove the rules of deduction (var_l), as well as declarations of the form $x : (\Delta)\mathbf{Type}$ from the definition of contexts. Again, we shall see that the result is a conservative subsystem.

We shall define in this chapter two countably infinite families of subsystems of TF: $\mathbf{SPar}\,(n)$ and $\mathbf{LPar}\,(n)$, where n is either a natural number or ω. For n a natural number, $\mathbf{LPar}\,(n)$ and $\mathbf{SPar}\,(n)$ are subsystems of TF in which all constants are nth order or below, and all variables are $n-1$th order or below; $\mathbf{LPar}\,(n)$ allows large parameters, while $\mathbf{SPar}\,(n)$ does not. $\mathbf{SPar}\,(\omega)$ allows variables and constants of any order, but does not allow large parameters; and $\mathbf{LPar}\,(\omega)$ is TF itself.

Further, we note that the logical object theories do not require the declaration of any equations; only the type theories required this, to handle the computation rules. Thus, for each of the above systems S, we formulate a system S^- which lacks the ability to declare equations. Again, the result is a conservative subsystem.

Aside from the general principle that frameworks should be as weak as possible, this subdivision of TF is important for the following reasons:

- We shall find that a subsystem of TF can be embedded within an existing logical framework, while TF itself cannot. For example, $\mathbf{SPar}\,(\omega)^-$ can be embedded as a conservative subsystem within ELF, while TF cannot. See Chapter 8.

- Occasionally, the subsystems of TF give an even better representation of the object theory. The adequacy results in the previous chapter took the form "There is a bijection between the entities of this syntactic class, and the TF terms M such that $J(M)$ holds." We shall see that, in the cases of propositional logic in $\mathbf{SPar}\,(1)^-$ and extended natural deduction in $\mathbf{SPar}\,(\omega)^-$, these judgements $J(M)$ are the only judgements derivable under the appropriate signature.

- While we do not investigate the matter in this thesis, we suspect that it would be much easier to give semantics to a system built within a weaker framework than a stronger; when giving semantics to a system built in $\mathbf{SPar}\,(2)$ or $\mathbf{LPar}\,(2)$, we do not have to find interpretations for the 2nd-order, 3rd-order, ... variables.

- This work shall prepare us for Part II, in which we build up logical frameworks as collections of features, the addition of any one of which is a conservative extension to the framework.

It is possible to slice TF much more finely than we do in this chapter. For example, we could consider restricting the parameters on equations separately from those on constant declarations, allowing only equations between terms of a type (and not the kind **Type** itself), allowing large parameters of some orders but not others, and so forth. We aim here to give a collection of subsystems that is adequate for the purposes listed above, while not being too complex.

5.1 Small Parametrisation

We define here the system **SPar** (n), where $n \in \{0, 1, 2, \dots, \omega\}$. This system allows constants of order n or below, variables of order below n, and does not allow large parameters.

Grammar For each arity α of order n or below, fix an infinite set $\mathcal{C}^\alpha$ of *α-ary constants*. For each arity α of order $< n$, fix an infinite set $\mathcal{V}^\alpha$ of *α-ary variables*.

For $\alpha \equiv (\alpha_1, \dots, \alpha_m)$, we define an *$\alpha$-ary variable sequence* to be a sequence

$$\langle x^{\alpha_1}, \dots, x^{\alpha_m} \rangle$$

of m *distinct* variables, the ith variable x^{α_i} being of arity α_i.

The set of *terms* of **SPar** (n) is defined inductively thus:

- Let $\alpha \equiv (\alpha_1, \dots, \alpha_m)$ be an arity of order $\leq n$. If c^α is an α-ary constant, and, for $i = 1, \dots, m$, $\vec{x}^{\alpha_i}$ is an α_i-ary variable sequence and M_i a term, then

  $$c^\alpha[[\vec{x}^{\alpha_1}]M_1, \dots, [\vec{x}^{\alpha_m}]M_m]$$

  is a term.

- Let $\alpha \equiv (\alpha_1, \dots, \alpha_m)$ be an arity of order $< n$. If x^α is an α-ary variable, and, for $i = 1, \dots, m$, $\vec{x}^{\alpha_i}$ is an α_i-ary variable sequence and M_i a term, then

  $$x^\alpha[[\vec{x}^{\alpha_1}]M_1, \dots, [\vec{x}^{\alpha_m}]M_m]$$

Each member of $\vec{x}^{\alpha_i}$ is bound within M_i in the terms

$$c^\alpha[[\vec{x}^{\alpha_1}]M_1, \dots, [\vec{x}^{\alpha_m}]M_m]$$

and

$$x^\alpha[[\vec{x}^{\alpha_1}]M_1, \dots, [\vec{x}^{\alpha_m}]M_m] \, ,$$

and we identify terms up to α-conversion.

We define abstractions, abstraction sequences, and instantiation as we did in TF (but only for arities of order $\leq n$). Again, a *kind* is either a term or **Type**.

For $\alpha \equiv (\alpha_1, \dots, \alpha_m)$ an arity of order $\leq n$, an *α-ary context* is defined to be a sequence of *variable declarations*

$$x^{\alpha_1} : (\Delta^{\alpha_1})M_1, \dots, x^{\alpha_m} : (\Delta^{\alpha_m})M_m$$

where x^{α_1}, ..., x^{α_m} are distinct variables, x^{α_i} of arity α_i; each Δ^{α_i} is an α_i-ary context; and each M_i is a term. (Note the difference from TF or **LPar** (n), where **Type** could also appear in any of the M_i places.)

A *signature* is defined to be a sequence consisting of *constant declarations*

$$c^\alpha : (\Delta^\alpha)K$$

where α is an arity of order $\leq n$, c^α is an α-ary constant, Δ^α an α-ary context, and K a kind; or *equations*

$$(\Delta^\alpha)(M = N : K)$$

where α is an arity of order $\leq n$, Δ^α is an α-ary context, M and N are terms, and K is a kind.

A *judgement* is an expression of one of the three forms

$$\begin{array}{lll} \Sigma & & \text{sig} \\ \Gamma & \vdash_\Sigma & \text{valid} \\ \Gamma & \vdash_\Sigma & M : K \\ \Gamma & \vdash_\Sigma & M = N : K \end{array}$$

where Σ is a signature, Γ a context, M and N terms, and K a kind.

The rules of deduction of **SPar** (n) are then the rules of TF, restricted so that no variable or constant of too high an arity appears, and we never use the rule (ctxt) with $K \equiv$ **Type**. For the sake of completeness, we list all the rules of **SPar** (n) in Figure 5.1.

We can now prove that the family of systems **SPar** (n) extend one another conservatively:

Theorem 5.1 *If* $m < n \leqslant \omega$, *then* **SPar** (n) *is a conservative extension of* **SPar** (m).

Proof We prove, by induction on derivations, that, if J is a judgement of **SPar** (m) derivable in **SPar** (n), then J is derivable in **SPar** (n). Consider any rule of deduction of **SPar** (n):

$$(R)\,\frac{J_1 \quad \cdots \quad J_k}{J}$$

As the conclusion J is a judgement of **SPar** (m), no constant of arity $> m$ nor variable of arity $\geqslant m$ occurs in it, and every context that appears is small. Therefore (by inspection of the rules of deduction), no such constant or variable occurs the context or signature of any of J_1, …, J_k, and all their contexts are small. Each J_i is derivable in **SPar** (n), hence in TF; therefore, by the Free Variables Lemma for TF, no constant of arity $> m$ nor variable of arity $\geqslant m$ occurs anywhere in J_i. (Alternatively, one could quite simply prove the Free Variables Lemma for **SPar** (n).) Hence, by the induction hypothesis, each of J_1, …, J_k is derivable in **SPar** (m). Therefore, applying the rule (R) in **SPar** (m), J is derivable in **SPar** (m). □

5.2 Large Parametrisation

We form the system **LPar** (n) $(n \leqslant \omega)$ by extending **SPar** (n) in the following way:

Firstly, we restore the original definition of context from TF. If $\alpha \equiv (\alpha_1, \ldots, \alpha_m)$ is an arity of order $\leq n$, an α-ary context is an expression of the form

$$x^{\alpha_1} : (\Delta^{\alpha_1})K_1, \ldots, x^{\alpha_m} : (\Delta^{\alpha_m})K_m$$

where x^{α_1}, …, x^{α_m} are distinct variables, x^{α_i} of arity α_i; Δ^{α_i} is an α_i-ary context, and K_i is a *kind* (and not necessarily just a term).

$$\text{(emp\_sig)} \quad \frac{}{\langle\rangle \text{ sig}}$$

$$\text{(sig}_l\text{)}\ \frac{\Delta^\alpha \vdash_\Sigma \text{valid}}{\Sigma, c^\alpha : (\Delta^\alpha)\mathbf{Type} \text{ sig}}\ (c^\alpha \notin \operatorname{dom}\Sigma) \qquad \text{(sig}_s\text{)}\ \frac{\Delta^\alpha \vdash_\Sigma M : \mathbf{Type}}{\Sigma, c^\alpha : (\Delta^\alpha)M \text{ sig}}\ (c^\alpha \notin \operatorname{dom}\Sigma)$$

$$\text{(eq\_sig)} \quad \frac{\Delta^\alpha \vdash_\Sigma M : K \quad \Delta^\alpha \vdash_\Sigma N : K}{\Sigma, (\Delta^\alpha)(M = N : K)}$$

$$\text{(emp\_ctxt)}\ \frac{\Sigma \text{ sig}}{\vdash_\Sigma \text{valid}} \qquad \text{(ctxt}_s\text{)}\ \frac{\Gamma, \Delta^\alpha \vdash_\Sigma M : \mathbf{Type}}{\Gamma, x^\alpha : (\Delta^\alpha)M \vdash_\Sigma \text{valid}}\ (x^\alpha \notin \operatorname{dom}\Gamma)$$

$$\text{(const}_l\text{)} \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{M}^\alpha] : \mathbf{Type}} \quad (c^\alpha : (\Delta^\alpha)\mathbf{Type} \in \Sigma)$$

$$\text{(const\_eq}_l\text{)} \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{M}^\alpha] = c^\alpha[\overrightarrow{N}^\alpha] : \mathbf{Type}} \quad (c^\alpha : (\Delta^\alpha)\mathbf{Type} \in \Sigma)$$

$$\text{(const}_s\text{)} \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{M}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}P} \quad (c^\alpha : (\Delta^\alpha)P \in \Sigma)$$

$$\text{(const\_eq}_s\text{)} \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{M}^\alpha] = c^\alpha[\overrightarrow{N}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}P} \quad (c^\alpha : (\Delta^\alpha)P \in \Sigma)$$

$$\text{(var}_s\text{)} \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{M}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}P} \quad (x^\alpha : (\Delta^\alpha)P \in \Gamma)$$

$$\text{(var\_eq}_s\text{)} \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{M}^\alpha] = x^\alpha[\overrightarrow{N}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}P} \quad (x^\alpha : (\Delta^\alpha)P \in \Gamma)$$

$$\text{(ref)}\ \frac{\Gamma \vdash_\Sigma M : K}{\Gamma \vdash_\Sigma M = M : K} \qquad \text{(sym)}\ \frac{\Gamma \vdash_\Sigma M = N : K}{\Gamma \vdash_\Sigma N = M : K}$$

$$\text{(trans)} \quad \frac{\Gamma \vdash_\Sigma M = N : K \quad \Gamma \vdash_\Sigma N = P : K}{\Gamma \vdash_\Sigma M = P : K}$$

$$\text{(eq\_use)} \quad \frac{\Gamma \Vdash_\sigma \vec{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma \{\vec{M}^\alpha/\Delta^\alpha\}N = \{\vec{M}^\alpha/\Delta^\alpha\}P : \{\vec{M}^\alpha/\Delta^\alpha\}K} \quad ((\Delta^\alpha)(N = P : K))$$

$$\text{(conv)} \quad \frac{\Gamma \vdash_\Sigma M : P \quad \Gamma \vdash_\Sigma P = Q : \mathbf{Type}}{\Gamma \vdash_\Sigma M : Q}$$

$$\text{(conv\_eq)} \quad \frac{\Gamma \vdash_\Sigma M = N : P \quad \Gamma \vdash_\Sigma P = Q : \mathbf{Type}}{\Gamma \vdash_\Sigma M = N : Q}$$

Figure 5.1: Rules of Deduction of $\mathbf{SPar}\,(n)$

We now add the following three rules of deduction:

$$(\text{weak\_ctxt}_l) \quad \frac{\Gamma, \Delta^\alpha \vdash_\Sigma \text{valid}}{\Gamma, x^\alpha : (\Delta^\alpha)\mathbf{Type} \vdash_\Sigma \text{valid}} \quad (x^\alpha \notin \operatorname{dom}\Gamma)$$

$$(\text{var}_l) \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{M}^\alpha] : \mathbf{Type}} \quad (x^\alpha : (\Delta^\alpha)\mathbf{Type} \in \Gamma)$$

$$(\text{var\_eq}_l) \quad \frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{M}^\alpha] = x^\alpha[\overrightarrow{N}^\alpha] : \mathbf{Type}} \quad (x^\alpha : (\Delta^\alpha)\mathbf{Type} \in \Gamma)$$

Note that TF is the same system as $\mathbf{LPar}\,(\omega)$.

We have the following two conservativity results:

Theorem 5.2 *If $m < n \leqslant \omega$, then $\mathbf{LPar}\,(n)$ is a conservative extension of $\mathbf{LPar}\,(m)$.*

Proof The proof is exactly similar to the proof for the **SPar** systems. We prove, by induction on derivations, that if J is a judgement of $\mathbf{LPar}\,(m)$ derivable in $\mathbf{LPar}\,(n)$, then J is derivable in $\mathbf{LPar}\,(m)$. For each rule of deduction

$$(R)\,\frac{J_1 \quad \cdots \quad J_k}{J}$$

observe that, if J is a judgement of $\mathbf{LPar}\,(m)$, then no constant of arity $> m$ nor variable of arity $\geqslant m$ occurs in the signature or context of any premise J_i. Therefore, by the Free Variables Lemma, each J_i is a judgement of $\mathbf{LPar}\,(m)$; hence, by the induction hypothesis, each J_i is derivable in $\mathbf{LPar}\,(m)$; hence, by (R), J is derivable in $\mathbf{LPar}\,(m)$. □

Theorem 5.3 *For $n \leqslant \omega$, $\mathbf{LPar}\,(n)$ is a conservative extension of $\mathbf{SPar}\,(n)$.*

Proof Again, the proof follows the familiar pattern. We prove, by induction on derivations, that if J is a judgement of $\mathbf{SPar}\,(n)$ derivable in $\mathbf{LPar}\,(n)$, then J is derivable in $\mathbf{SPar}\,(n)$. For each rule of deduction

$$(R)\,\frac{J_1 \quad \cdots \quad J_k}{J}$$

if J is a judgement of $\mathbf{SPar}\,(n)$, then no large context occurs in the signature or context of J, hence no large context occurs in the signature or context of any premise J_i. Therefore, by the Free Variables Lemma, each J_i is a judgement of $\mathbf{SPar}\,(n)$, hence derivable in $\mathbf{SPar}\,(n)$ by the induction hypothesis. Now, (R) cannot be one of the new rules of deduction, as the conclusion of any of these is never a judgement of $\mathbf{SPar}\,(n)$. (In the conclusion of (weak_ctxt$_l$), the context of the conclusion is explicitly a large context; in (var$_l$) and (var_eq$_l$), the context Γ must be large by the side condition.) Therefore, (R) is a rule of deduction of $\mathbf{SPar}\,(n)$, and so we may conclude that J is derivable in $\mathbf{SPar}\,(n)$. □

Corollary 5.4 *If $m < n \leqslant \omega$, $\mathbf{LPar}\,(n)$ is a conservative extension of $\mathbf{SPar}\,(m)$.*

Note that all of our conservativity results are very strong. If J is a judgement of the smaller system, the proofs we have given show, not merely that if J is derivable in the larger system then it is derivable in the smaller, but in fact that any derivation of J in the larger system is itself a derivation of J in the smaller.

We can now go on to prove all the metatheoretic results we proved for TF for each of these systems. It is not difficult to prove each result for each system directly; however, we can also take advantage of the conservativity results. We give one example here:

Lemma 5.5 (Cut) *For each system S, either* $\mathbf{SPar}\,(n)$ *or* $\mathbf{LPar}\,(n)$*, if*

$$\Gamma_1, \Delta, \Gamma_2 \vdash_\Sigma J$$

and

$$\Gamma_1 \Vdash_\Sigma \overrightarrow{M} :: \Delta$$

then

$$\Gamma_1, \{\overrightarrow{M}/\Delta\}\Gamma_2 \vdash_\Sigma \{\overrightarrow{M}/\Delta\}J$$

Proof Suppose the premises hold in S. Then they are derivable in TF. Hence, by the Cut Lemma for TF, the conclusion is derivable in TF. Therefore, by the conservativity results (remember that TF is $\mathbf{LPar}\,(\omega)$), the conclusion holds in S. $\square$

5.3 Systems without Equations

We can remove from each system the ability to declare equations. Thanks to Lemma 3.23, we may as well remove judgement forms for equality entirely from such systems. Thus, for each of the above systems S, we define S^- to be the result of removing from S:

- The judgement form $\Gamma \vdash_\Sigma M = N : K$.
- The declaration form $(\Delta)(M = N : K)$ in signatures.
- The rules of deduction (eq_sig), (const_eq), (var_eq), (eq_use), (ref), (sym), (trans), (conv) and (conv_eq).

Theorem 5.6 *S is a conservative extension of S^-.*

Proof Let Δ be a derivation of an S^--judgement J in S. No equation can occur in the signature of J, hence no equation occurs in the signature of any of the judgements of Δ. Now, the lowest equality judgement in the derivation must be the premise of a use of the (conv) rule:

$$\frac{\begin{array}{c}\vdots \\ \Gamma \vdash_\Sigma M : N\end{array} \qquad \begin{array}{c}\vdots \\ \Gamma \vdash_\Sigma N = P : \mathbf{Type}\end{array}}{\Gamma \vdash_\Sigma M : P}$$

By Lemma 3.23, $N \equiv P$; therefore, the first premise $\Gamma \vdash_\Sigma M : N$ is the same as the conclusion $\Gamma \vdash_\Sigma M : P$. We may thus remove the judgement $\Gamma \vdash_\Sigma N = P : \mathbf{Type}$ and the entire tree above it, and replace the above node with

$$\begin{array}{c} \vdots \\ \Gamma \vdash_\Sigma M : N \end{array}$$

Repeating this process yields a derivation Δ' of J in which no equality judgements occur; hence no instance of the 9 removed rules of deduction occur, either. Δ' is therefore a derivation in S^-. $\square$

5.4 Typed Abstractions

When comparing our weak logical frameworks with other frameworks from the literature, it shall be very convenient to have versions of our systems in which the variables x_{ij} in terms such as

$$z[[x_{11}, \dots, x_{1k_1}]M_1, \dots, [x_{n1}, \dots, x_{nk_n}]M_n]$$

(z a variable or constant) are *typed*; that is, the product kinds that they inhabit are written into the terms:

$$z[[x_{11} : (\Delta_{11})K_{11}, \dots, x_{1k_1} : (\Delta_{1k_1})K_{1k_1}]M_1, \dots, [x_{n1} : (\Delta_{n1})K_{n1}, \dots, x_{nk_n} : (\Delta_{nk_n})K_{nk_n}]M_n]$$

With most type systems, this would be quite a large change; we would, for example, be replacing a single identity function $\lambda x.x$ with infinitely many functions $\lambda x : K.x$, one for each kind K. For our weak logical frameworks, however, every term is the application of a variable or constant z, and so (relative to a given signature and context) there is always only one choice of kinds; simply look up the kind of z in the signature or context. Thus, the version S_c of each of our systems S that we shall define shall be equivalent to S, in a sense that we shall make precise below.

We give here the details of TF_c only; the definitions for $\mathbf{LPar}\,(n)_c$, $\mathbf{SPar}\,(n)_c$, $\mathbf{LPar}\,(n)_c^-$ and $\mathbf{SPar}\,(n)_c^-$ follow a similar pattern.

For TF_c, we define the sets of *terms*, *kinds*, *α-ary product kinds*, *α-ary contexts*, *α-ary abstractions* and *α-ary abstraction sequences* simultaneously as follows:

Terms If z^α is an α-ary constant or variable, and $\overrightarrow{M}^\alpha$ an α-ary abstraction sequence, then

$$z^\alpha[\overrightarrow{M}^\alpha]$$

is a term.

Kinds A *(base) kind* is either a term or **Type**.

α-ary product kinds An α-ary product kind is an expression of the form

$$K^\alpha \equiv (\Delta^\alpha)K$$

where Δ^α is an α-ary context and K a kind. Every member of $\operatorname{dom}\Delta^\alpha$ is bound within this expression wherever it appears.

α-ary contexts If $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, an *$\alpha$-ary context* is a sequence of *variable declarations*

$$\Delta^\alpha \equiv \langle x^{\alpha_1} : K^{\alpha_1}, \ldots, x^{\alpha_n} : K^{\alpha_n}\rangle$$

where x^{α_1}, …, x^{α_n} are distinct variables, x^{α_i} of arity α_i, and K^{α_i} is an α_i-ary product kind. Its *domain* is the α-ary variable sequence

$$\langle x^{\alpha_1}, \ldots, x^{\alpha_n}\rangle$$

α-ary abstractions An *α-ary abstraction* is an expression of the form

$$M^\alpha \equiv [\Delta^\alpha]M$$

where Δ^α is an α-ary context and M a term. Every member of $\operatorname{dom}\Delta^\alpha$ is bound in this expression wherever it appears.

This is the only difference between the definition of TF_c and the definition of TF: that an abstraction is formed from a context and a term, rather than a variable sequence and term.

α-ary abstraction sequence If $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, an *$\alpha$-ary abstraction sequence* is a sequence

$$\overrightarrow{M}^\alpha \equiv \langle M^{\alpha_1}, \ldots, M^{\alpha_n}\rangle$$

whose ith member M^{α_i} is an α_i-ary abstraction.

A *signature* is now defined to be a sequence of *constant declarations*

$$c^\alpha : K^\alpha$$

where c^α is an α-ary constant and K^α an α-ary product kind, no two declarations being of the same constant.

There are four judgement forms in TF_c:

$$\begin{array}{lcl} \Sigma & & \text{sig} \\ \Gamma & \vdash_\Sigma & \text{valid} \\ \Gamma & \vdash_\Sigma & M : K \\ \Gamma & \vdash_\Sigma & M = N : K \end{array}$$

where Σ is a signature, Γ a context, M and N terms, and K a (base) kind.

We define instantiation exactly as for TF. For the defined judgement forms, we must define

- equality of abstraction sequences $\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$
- equality of kinds $\Gamma \Vdash_\Sigma K = K'$
- equality of contexts $\Gamma \Vdash_\Sigma \Delta = \Delta'$

all simultaneously. The definitions now take the following shape.

Let
$$\Delta \equiv x^{\alpha_1} : (\Delta^{\alpha_1})K_1, \dots, x^{\alpha_n} : (\Delta^{\alpha_n})K_n$$
be an α-ary context, and
$$\overrightarrow{M}^\alpha \equiv M^{\alpha_1}, \dots, M^{\alpha_n}$$
an α-ary abstraction sequence whose ith member has the form
$$M^{\alpha_i} \equiv [\Theta^{\alpha_i}]M_i \tag{5.1}$$
Then we define
$$\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha$$
to be the set consisting of the following n judgements:
$$\begin{array}{rcl}
\Gamma & \Vdash_\Sigma & \Delta^{\alpha_1} = \Theta^{\alpha_1}, \\
\Gamma, \Delta^{\alpha_1} & \vdash_\Sigma & M_1 : K_1, \\
\Gamma & \Vdash_\Sigma & \{M^{\alpha_1}/x^{\alpha_1}\}\Delta^{\alpha_2} = \Theta^{\alpha_2}, \\
\Gamma, \{M^{\alpha_1}/x^{\alpha_1}\}\Delta^{\alpha_2} & \vdash_\Sigma & M_2 : \{M^{\alpha_1}/x^{\alpha_1}\}K_2 \\
 & \vdots & \\
\Gamma, \{M^{\alpha_1}/x^{\alpha_1}, \dots, M^{\alpha_{n-1}}/x^{\alpha_{n-1}}\}\Delta^{\alpha_n} & \vdash_\Sigma & M_n : \{M^{\alpha_1}/x^{\alpha_1}, \dots, M^{\alpha_{n-1}}/x^{\alpha_{n-1}}\}K_n
\end{array}$$
We define $\Gamma \Vdash \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha$ similarly.

We define equality of kinds by:
$$\begin{array}{rcl}
\Gamma \Vdash (\Delta)\,\mathrm{El}\,(A) = (\Delta')\,\mathrm{El}\,(B) & \equiv & (\Gamma \Vdash \Delta = \Delta') \\
 & & \cup\{\Gamma \vdash A = B : \mathbf{Type}\} \\
\Gamma \Vdash (\Delta)\mathbf{Type} = (\Delta')\mathbf{Type} & \equiv & (\Gamma \Vdash \Delta = \Delta')
\end{array}$$
and equality of contexts as follows. If $\Delta \equiv x^{\alpha_1} : K^{\alpha_1}, \dots, x^{\alpha_n} : K^{\alpha_n}$ and $\Delta' \equiv x^{\alpha_1} : (K')^{\alpha_1}, \dots, x^{\alpha_n} : (K')^{\alpha_n}$, then $\Gamma \Vdash \Delta = \Delta'$ is the set of judgements
$$\begin{array}{c}
(\Gamma \Vdash K^{\alpha_1} = (K')^{\alpha_1}) \\
\cup(\Gamma, x^{\alpha_1} : K^{\alpha_1} \Vdash K^{\alpha_2} = (K')^{\alpha_2}) \\
\cup\cdots \\
\cup(\Gamma, x^{\alpha_1} : K^{\alpha_1}, \dots, x^{\alpha_{n-1}} : K^{\alpha_{n-1}} \Vdash K^{\alpha_n} = (K')^{\alpha_n})
\end{array}$$

The rules of deduction of TF_c, when written down, appear exactly the same as those of TF (Figure 3.1), with the understanding that the variables in abstractions are now typed, and the definitions of instantiation and the defined judgement forms are modified accordingly.

We can now define the function discussed above. This shall be our translation from TF to TF_c which we shall use to prove the two systems equivalent. In essence, it consists of little more than looking up the kinds as given by the relevant declaration in the signature or context. To fill in the types in

$$x[\vec{M}]$$

we use the kinds provided by the declaration of x in the context; to fill in the types in $c[\vec{M}]$, we use the kinds provided by the declaration of c in the signature. In detail, the definition is:

Definition 5.7 *Let the TF context Γ be consistent with respect to the TF signature Σ. We define, for every TF term M such that every free variable in M is declared in Γ and every constant in M is declared in Σ, the TF_c term*

$$[\![M]\!]^{\Gamma}_{\Sigma}\ ;$$

for every TF kind K satisfying the same conditions, the TF_c kind

$$[\![K]\!]^{\Gamma}_{\Sigma}\ ;$$

and, for every TF context Δ such that Γ, Δ is consistent with respect to Σ, the TF_c context of the same arity

$$[\![\Delta]\!]^{\Gamma}_{\Sigma}$$

The following abbreviations shall be convenient for this definition. Given an α-ary TF context

$$\Delta^{\alpha} \equiv x^{\alpha_1} : (\Delta^{\alpha_1})K_1, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})K_n$$

and an α-ary TF abstraction sequence

$$\overrightarrow{M}^{\alpha} \equiv [\operatorname{dom}\Delta^{\alpha_1}]M_1, \ldots, [\operatorname{dom}\Delta^{\alpha_n}]M_n$$

we write, for any TF expression X,

$$\begin{array}{rcl} X^{*} & \equiv & \{\overrightarrow{M}^{\alpha} / \operatorname{dom}\Delta^{\alpha}\}X \\ [\![M^{\alpha_i}]\!]^{\Gamma}_{\Sigma} & \equiv & [[\![(\Delta^{\alpha_i})^{*}]\!]^{\Gamma}_{\Sigma}]\, [\![M_i]\!]^{\Gamma,\Delta^{\alpha_i}}_{\Sigma} \\ \left[\!\left[\overrightarrow{M}^{\alpha}\right]\!\right]^{\Gamma}_{\Sigma} & \equiv & [\![M^{\alpha_1}]\!]^{\Gamma}_{\Sigma}, \ldots, [\![M^{\alpha_n}]\!]^{\Gamma}_{\Sigma} \end{array}$$

Note that all these abbreviations are relative to the given α-ary context Δ^{α}.

The definition is as follows:

- *If $c^{\alpha} : (\Delta^{\alpha})K \in \Sigma$, then*

$$\left[\!\left[c^{\alpha}[\overrightarrow{M}^{\alpha}]\right]\!\right]^{\Gamma}_{\Sigma} \equiv c^{\alpha}\left[\left[\!\left[\overrightarrow{M}^{\alpha}\right]\!\right]^{\Gamma}_{\Sigma}\right]$$

where the right-hand side is defined with respect to the context Δ^α.

- *If $x^\alpha : (\Delta^\alpha)K \in \Gamma$, then*
$$\left[\!\left[x^\alpha[\overrightarrow{M}^\alpha] \right]\!\right]^\Gamma_\Sigma \equiv x^\alpha \left[\left[\!\left[\overrightarrow{M}^\alpha \right]\!\right]^\Gamma_\Sigma \right]$$
where the right-hand side is define with respect to the context Δ^α.

- $[\![\mathbf{Type}]\!]^\Gamma_\Sigma \equiv \mathbf{Type}$

- $[\![\langle\rangle]\!]^\Gamma_\Sigma \equiv \langle\rangle$

- $[\![\Theta, x^\alpha : (\Delta^\alpha)K]\!]^\Gamma_\Sigma \equiv [\![\Theta]\!]^\Gamma_\Sigma , x^\alpha : ([\![\Delta^\alpha]\!]^{\Gamma,\Theta}_\Sigma)\, [\![K]\!]^{\Gamma,\Theta,\Delta^\alpha}_\Sigma$

We now define, for each TF signature Σ, the TF_c signature $[\![\Sigma]\!]$ thus:

$$\begin{array}{rcl}
[\![\langle\rangle]\!] & \equiv & \langle\rangle \\
[\![\Sigma, c^\alpha : (\Delta^\alpha)K]\!] & \equiv & [\![\Sigma]\!] , c^\alpha : ([\![\Delta^\alpha]\!]^{\langle\rangle}_\Sigma)\, [\![K]\!]^{\Delta^\alpha}_\Sigma \\
[\![\Sigma, (\Delta)(M = N : K)]\!] & \equiv & [\![\Sigma]\!] , ([\![\Delta]\!]^{\langle\rangle}_\Sigma)([\![M]\!]^\Delta_\Sigma = [\![N]\!]^\Delta_\Sigma : [\![K]\!]^\Delta_\Sigma)
\end{array}$$

Finally we define, for each TF judgement J, the TF_c judgement $[\![J]\!]$ thus:

$$\begin{array}{rcl}
[\![\Sigma \text{ sig}]\!] & \equiv & [\![\Sigma]\!] \text{ sig} \\
[\![\Gamma \vdash_\Sigma \text{valid}]\!] & \equiv & [\![\Gamma]\!]^{\langle\rangle}_\Sigma \vdash_{[\![\Sigma]\!]} \text{valid} \\
[\![\Gamma \vdash_\Sigma M : K]\!] & \equiv & [\![\Gamma]\!]^{\langle\rangle}_\Sigma \vdash_{[\![\Sigma]\!]} [\![M]\!]^\Gamma_\Sigma : [\![K]\!]^\Gamma_\Sigma \\
[\![\Gamma \vdash_\Sigma M = N : K]\!] & \equiv & [\![\Gamma]\!]^{\langle\rangle}_\Sigma \vdash_{[\![\Sigma]\!]} [\![M]\!]^\Gamma_\Sigma = [\![N]\!]^\Gamma_\Sigma : [\![K]\!]^\Gamma_\Sigma
\end{array}$$

The main work in this definition is hidden away inside one of the abbreviations we made at the start. When translating an abstraction M^{α_i} to form $[\![M^{\alpha_i}]\!]^\Gamma_\Sigma$, we infer the kinds that must be given to the bound variables from the given context Δ^α.

The following result, which can serve both as a Strengthening and a Weakening Lemma, is easily proven:

Lemma 5.8 *Let X be either a term, kind or context. If $\Gamma \subseteq \Delta$, every free variable in X is declared in Γ, and every constant in X is declared in Σ, then*

$$[\![X]\!]^\Gamma_\Sigma \equiv [\![X]\!]^\Delta_\Sigma$$

Proof An easy induction on X. □

We need the following result showing how instantiation behaves under the translation:

Lemma 5.9 *Let $\Gamma, \Delta^\alpha, \Theta$ be consistent with respect to Σ. Let $\overrightarrow{M}^\alpha$ be an α-ary abstraction sequence in which every free variable that occurs is declared in Γ, and every constant that occurs is declared in Σ.*

1. *If P is a term in which every free variable that occurs is declared in $\Gamma, \Delta^\alpha, \Theta$ and every constant that occurs is declared in Σ, then*

$$\left[\!\left[\{\overrightarrow{M}^\alpha/\Delta^\alpha\}P\right]\!\right]_\Sigma^{\Gamma,\Theta^*} \equiv \left\{\left[\!\left[\overrightarrow{M}^\alpha\right]\!\right]_\Sigma^\Gamma/\Delta^\alpha\right\}[\![P]\!]_\Sigma^{\Gamma,\Delta^\alpha,\Theta}$$

2. *If K is a kind in which every free variable that occurs is declared in $\Gamma, \Delta^\alpha, \Theta$ and every constant that occurs is declared in Σ, then*

$$\left[\!\left[\{\overrightarrow{M}^\alpha/\Delta^\alpha\}K\right]\!\right]_\Sigma^{\Gamma,\Theta^*} \equiv \left\{\left[\!\left[\overrightarrow{M}^\alpha\right]\!\right]_\Sigma^\Gamma/\Delta^\alpha\right\}[\![K]\!]_\Sigma^{\Gamma,\Delta^\alpha,\Theta}$$

3. *If Φ is a context such that $\Gamma, \Delta^\alpha, \Theta, \Phi$ is consistent with respect to Σ, then*

$$\left[\!\left[\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Phi\right]\!\right]_\Sigma^{\Gamma,\Theta^*} \equiv \left\{\left[\!\left[\overrightarrow{M}^\alpha\right]\!\right]_\Sigma^\Gamma/\Delta^\alpha\right\}[\![\Phi]\!]_\Sigma^{\Gamma,\Delta^\alpha,\Theta}$$

(Recall that Θ^* abbreviates $\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Theta$.)

Proof The three parts are proved simultaneously by double induction, firstly on the arity α, secondly on the expression P, K or Φ. The only difficult case is when P is an application of a variable in $\operatorname{dom}\Delta^\alpha$; say

$$P \equiv x^{\alpha_i}[\overrightarrow{N}^{\alpha_i}]$$

where

$$x^{\alpha_i} : (\Delta^{\alpha_i})K \in \Gamma$$

In this case,

$$\begin{aligned}
\left\{\left[\!\left[\overrightarrow{M}\right]\!\right]_\Sigma^\Gamma/\Delta\right\}[\![P]\!]_\Sigma^{\Gamma,\Delta,\Theta} &\equiv \left\{\left[\!\left[\overrightarrow{M}\right]\!\right]_\Sigma^\Gamma/\Delta\right\}x\left[\left[\!\left[\overrightarrow{N}\right]\!\right]_\Sigma^{\Gamma,\Delta,\Theta}\right] \\
&\equiv \left\{\left\{\left[\!\left[\overrightarrow{M}\right]\!\right]_\Sigma^\Gamma/\Delta\right\}\left[\!\left[\overrightarrow{N}\right]\!\right]_\Sigma^{\Gamma,\Delta,\Theta}/\Delta\right\}[\![M_i]\!]_\Sigma^\Gamma \\
&\equiv \left\{\left[\!\left[\left\{\overrightarrow{M}/\Delta\right\}\overrightarrow{N}\right]\!\right]_\Sigma^{\Gamma,\Theta^*}/\Delta\right\}[\![M_i]\!]_\Sigma^\Gamma \qquad \text{(i.h. on } P\text{)} \\
&\equiv \left\{\left[\!\left[\{\overrightarrow{M}/\Delta\}\overrightarrow{N}\right]\!\right]_\Sigma^{\Gamma,\Theta^*}/\Delta\right\}[\![M_i]\!]_\Sigma^{\Gamma,\Theta^*,\Delta} \qquad \text{(Lemma 5.8)} \\
&\equiv \left[\!\left[\{\{\overrightarrow{M}/\Delta\}\overrightarrow{N}/\Delta\}M_i\right]\!\right]_\Sigma^{\Gamma,\Theta^*} \qquad \text{(i.h. on } \alpha\text{)} \\
&\equiv \left[\!\left[\{\overrightarrow{M}/\Delta\}P\right]\!\right]_\Sigma^{\Gamma,\Theta^*}
\end{aligned}$$

□

Corollary 5.10

$$\begin{aligned}
\left[\!\left[\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha\right]\!\right] &\equiv [\![\Gamma]\!]_\Sigma^{\langle\rangle} \Vdash_{[\![\Sigma]\!]} \left[\!\left[\overrightarrow{M}^\alpha\right]\!\right]_\Sigma^\Gamma :: [\![\Delta^\alpha]\!]_\Sigma^\Gamma \\
\left[\!\left[\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha\right]\!\right] &\equiv [\![\Gamma]\!]_\Sigma^{\langle\rangle} \Vdash_{[\![\Sigma]\!]} \left[\!\left[\overrightarrow{M}^\alpha\right]\!\right]_\Sigma^\Gamma = \left[\!\left[\overrightarrow{N}^\alpha\right]\!\right]_\Sigma^\Gamma :: [\![\Delta^\alpha]\!]_\Sigma^\Gamma
\end{aligned}$$

Proof Let

$$\begin{array}{rcl} \Delta & \equiv & x^{\alpha_1} : (\Delta^{\alpha_1})K_1, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})K_n \\ \overrightarrow{M} & \equiv & [\mathrm{dom}\,\Delta^{\alpha_1}]M_1, \ldots, [\mathrm{dom}\,\Delta^{\alpha_n}]M_n \end{array}$$

Then $\Gamma \Vdash_\Sigma \overrightarrow{M} :: \Delta$ consists of n judgements, the ith of which is

$$\Gamma, (\Delta)^* \vdash_\Sigma M_i : K_i^*$$

By the lemma, for any expression X,

$$\begin{array}{rcl} [\![X^*]\!]^\Gamma_\Sigma & \equiv & \left[\!\!\left[\{\overrightarrow{M}/\Delta\}X\right]\!\!\right]^\Gamma_\Sigma \\ & \equiv & \left\{\left[\!\!\left[\overrightarrow{M}\right]\!\!\right]^\Gamma/\Delta\right\}[\![X]\!]^{\Gamma,\Delta}_\Sigma \end{array}$$

Let us write $Y^\dagger$ for

$$\left\{\left[\!\!\left[\overrightarrow{M}\right]\!\!\right]^\Gamma/\Delta\right\}Y$$

when Y is an expression of TF_c. Then we have just shown

$$[\![X^*]\!]^\Gamma_\Sigma \equiv ([\![X]\!]^{\Gamma,\Delta}_\Sigma)^\dagger\,.$$

Hence,

$$\begin{array}{rcl} [\![\Gamma, (\Delta)^* \vdash_\Sigma M_i : K_i^*]\!] & \equiv & [\![\Gamma, (\Delta)^*]\!]^{\langle\rangle}_\Sigma \vdash_{[\![\Sigma]\!]} [\![M_i]\!]^{\Gamma,(\Delta)^*}_\Sigma : [\![K_i^*]\!]^{\Gamma,(\Delta)^*}_\Sigma \\ & \equiv & [\![\Gamma]\!]^{\langle\rangle}_\Sigma, [\![(\Delta)^*]\!]^\Gamma_\Sigma \vdash_{[\![\Sigma]\!]} [\![M_i]\!]^{\Gamma,(\Delta)^*}_\Sigma : [\![K_i^*]\!]^{\Gamma,(\Delta)^*}_\Sigma \\ & & \quad\text{(by the way } [\![\Delta]\!]^\Gamma_\Sigma \text{ is defined)} \\ & \equiv & [\![\Gamma]\!]^{\langle\rangle}_\Sigma, ([\![\Delta]\!]^{\Gamma,\Delta}_\Sigma)^\dagger \vdash_{[\![\Sigma]\!]} [\![M_i]\!]^{\Gamma,(\Delta)^*}_\Sigma : ([\![K_i]\!]^{\Gamma,(\Delta)^*,\Delta}_\Sigma)^\dagger \end{array}$$

and (making use of Lemma 5.8) this is the ith judgement of

$$[\![\Gamma]\!]^{\langle\rangle}_\Sigma \Vdash_{[\![\Sigma]\!]} \left[\!\!\left[\overrightarrow{M}\right]\!\!\right]^\Gamma_\sigma :: [\![\Delta]\!]^\Gamma_\Sigma$$

The proof for equality judgements is similar. $\square$

We can now prove the soundness of the translation. We extend the translation to derivations; given a TF derivation, we translate each judgement under $[\![\,]\!]$ to form a tree of TF_c judgements. Not only is it true that the result is always a TF_c derivation, we have the following

Theorem 5.11 *$[\![\,]\!]$ is a bijection between the set of TF derivations and the set of TF_c derivations.*

Proof Given a TF_c expression X (a term, kind, judgement, or derivation), let us write X^- for the result of removing all the kinds on variables in X. Now that we have Lemma 5.9 and Corollary 5.10, two simple inductions on derivations show:

- If D is a TF derivation, then $[\![D]\!]$ is a TF_c derivation, and $[\![D]\!]^- \equiv D$.
- If D is a TF_c derivation, then D^- is a TF derivation, and $[\![D^-]\!] \equiv D$.

$\square$

5.5 Object Logics in the Subsystems of TF

We observed earlier that our representations of object theories in Chapter 4 used only constants and variables of low order. Each of these representations provides an adequate encoding of the object logic in any of our subsystems of TF in which the appropriate signature can be declared; this follows from the adequacy theorems proven in that chapter and the conservativity results proven in this. For each object theory, there is a minimal subsystem in which the corresponding signature can be declared, and in a few cases the *only* judgements derivable in that system under that signature are those that correspond to judgements of the object theory. One such case is propositional logic:

5.5.1 Propositional Logic

Let Σ_{prop} be the signature, and $[\![\,]\!]$ the translation, from Section 4.1. Note that Σ_{prop} is a signature of $\mathbf{SPar}\,(1)^-$, and each $[\![\phi]\!]$ and $[\![D]\!]$ is a term of $\mathbf{SPar}\,(1)^-$. We shall prove that the only judgements derivable under Σ_{prop} in $\mathbf{SPar}\,(1)^-$ are those of the form we considered in Theorem 4.1, possibly with the context rearranged.

In particular, a context is valid under the signature Σ_{prop} if and only if it is a rearrangement of a context of the form

$$p_1 : \text{form}, \ldots, p_n : \text{form}, q_1 : \mathbf{Prf}[[\![\phi_1]\!]], \ldots, q_m : \mathbf{Prf}[[\![\phi_m]\!]]\,,$$

where each ϕ_i is a formula whose propositional variables are among $p_1, \ldots, p_n$, such that, if p_j occurs in ϕ_i, then the declaration $q_i : \mathbf{Prf}[[\![\phi_i]\!]]$ may not be moved to before the declaration p_j : form. This set of contexts can also be characterised inductively as follows. We define the set of *correct* contexts thus:

- $\langle\rangle$ is correct.
- If Γ is correct, and p is a propositional variable not declared in Γ, then Γ, p : form is correct.
- If Γ is correct, ϕ is a formula every propositional variable in which is declared in Γ, and q is a occurrence marker for ϕ not declared in Γ, then $\Gamma, q : \mathbf{Prf}[[\![\phi]\!]]$ is correct.

We shall assume that, whenever a declaration is made in a context, the variable is either a propositional variable or an occurrence marker for the appropriate formula. (That there is always an appropriate formula is part of the theorem we are about to prove.) Our adequacy result is then as follows:

Theorem 5.12 *In* $\mathbf{SPar}\,(1)^-$*:*

1. Σ_{prop} sig.

2. $\Gamma \vdash_{\Sigma_{prop}}$ valid *iff* Γ *is correct.*

3. $\Gamma \vdash_{\Sigma_{prop}} M : \mathbf{Type}$ *iff* Γ *is correct, and either* $M \equiv \mathrm{form}$ *or* $M \equiv \mathbf{Prf}[[\![\phi]\!]]$ *for some formula* ϕ *every propositional variable in which is declared in* Γ.

4. $\Gamma \vdash_{\Sigma_{prop}} M : \mathrm{form}$ *iff* Γ *is correct and* $M \equiv \mathbf{Prf}[[\![\phi]\!]]$ *for some formula* ϕ *every propositional variable in which is declared in* Γ.

5. $\Gamma \vdash_{\Sigma_{prop}} M : \mathbf{Prf}[[\![\phi]\!]]$ *iff* Γ *is correct, every propositional variable in* ϕ *is declared in* Γ, *and* $M \equiv [\![D]\!]$ *for some derivation* D *of* ϕ *such that, if* q *is a occurrence marker for* ψ *occurring in* ϕ, *then* $q : \mathbf{Prf}[[\![\phi]\!]]$ *occurs in* Γ.

6. *Every judgement of the form* $\Gamma \vdash_{\Sigma_{prop}} J$ *derivable in* $\mathbf{SPar}\,(1)^-$ *is of one of the above forms.*

Proof Part 1 and the right-to-left directions of parts 2 – 5 can be proven directly for $\mathbf{SPar}\,(1)^-$ as we did for TF; we shall choose instead to derive them from the result for TF via the conservativity theorem.

1 follows from the fact that Σ_{prop} sig is derivable in TF (Theorem 4.1), and the conservativity result for $\mathbf{SPar}\,(1)^-$ in TF.

A simple induction shows that every correct context is valid under Σ_{prop} in TF. (We need to use Theorem 4.1.2 for this.) The right-to-left direction of part 2 is thus proven. The right-to-left directions of parts 4 and 5 follow from Theorem 4.1 and the Weakening lemma, which allows us to replace any context of the form

$$p_1 : \mathrm{form}, \ldots, p_n : \mathrm{form}, q_1 : \mathbf{Prf}[[\![\phi_1]\!]], \ldots, q_m : \mathbf{Prf}[[\![\phi_m]\!]]$$

with any correct rearrangement. The right-to-left direction of part 3 follows by the rule (const) applied to form and **Prf**.

The left-to-right directions of parts 2 – 5 are proven simulatneously by induction on the derivation in $\mathbf{SPar}\,(1)^-$. No case presents any difficulty.

For part 6, if J is valid or $M : \mathbf{Type}$, the result follows from part 2 or 3 respectively; so suppose $J \equiv M : N$, where N is a term. By Type Validity, $\Gamma \vdash_{\Sigma_{prop}} N : \mathbf{Type}$, so, by part 2, either $N \equiv \mathrm{form}$ or $N \equiv \mathbf{Prf}[[\![\phi]\!]]$ for some formula ϕ every propositional variable in which is declared in Γ. The conclusion now follows from parts 4 or 5 respectively. $\Box$

It would also have been possible to prove this theorem directly, again using little more than an inspection of the rules of deduction of $\mathbf{SPar}\,(1)^-$ under Σ_{prop}, then derive Theorem 4.1 via the conservativity result.

5.5.2 Predicate Logic

Let Σ_L and $[\![\,]\!]$ be as in Section 4.2. Σ_L is a signature of $\mathbf{SPar}\,(2)^-$, and each $[\![t]\!]$, $[\![\phi]\!]$ and $[\![D]\!]$ is a term of $\mathbf{SPar}\,(2)^-$. By the conservativity result, Theorem 4.5 still holds in $\mathbf{SPar}\,(2)^-$. It is not quite true that the only judgements derivable in $\mathbf{SPar}\,(2)^-$ are those discussed in Theorem

4.5, for it is possible to declare first-order variables, none of which appear in the contexts of $\mathbf{SPar}\,(2)^-$.

Even if we restrict the contexts to those declaring only base variables (i.e. first-order contexts), there is still one minor difficulty to be overcome: the fact that it is possible to declare variables of type form, which behave as propositional variables. Therefore, let us introduce propositional variables to the syntax of predicate logic, with every propositional variable being a base variable of $\mathbf{SPar}\,(2)^-$. The changes that must be made to the definitions in Section 4.2 are obvious.

Having made these changes, and if we restrict contexts to first-order, we can strengthen the adequacy theorem as follows.

Again we begin by characterising the set of *correct* contexts inductively.

- $\langle\rangle$ is correct.
- If Γ is correct, and x is an individual variable not declared in Γ, then $\Gamma, x : \mathrm{term}$ is correct.
- If Γ is correct, and p is a propositional variable not declared in Γ, then $\Gamma, p : \mathrm{form}$ is correct.
- If Γ is correct, ϕ is a formula every free (individual and propositional) variable in which is declared in Γ, and q is a occurrence marker for ϕ not occurring in Γ, then $\Gamma, q : \mathbf{Prf}[[\![\phi]\!]]$ is correct.

Again, we assume that, whenever a declaration is made in a context, the variable is of the appropriate class. Our adequacy theorem now reads as follows:

Theorem 5.13 *Let L be a first-order language. In $\mathbf{SPar}\,(2)^-$:*

1. Σ_L sig.
2. *If Γ is a first-order context, then $\Gamma \vdash_{\Sigma_L}$ sig iff Γ is correct.*
3. *If Γ is a first-order context, then $\Gamma \vdash_{\Sigma_L} M : \mathbf{Type}$ iff Γ is correct and either $M \equiv \mathrm{term}$, $M \equiv \mathrm{form}$, or $M \equiv \mathbf{Prf}[[\![\phi]\!]]$ for some formula ϕ every free variable in which is declared in Γ.*
4. *If Γ is a first-order context, then $\Gamma \vdash_{\Sigma_L} M : \mathrm{term}$ iff Γ is correct and $M \equiv [\![t]\!]$ for some term t every variable in which is declared in Γ.*
5. *If Γ is a first-order context, then $\Gamma \vdash_{\Sigma_L} M : \mathrm{form}$ iff Γ is correct and $M \equiv [\![\phi]\!]$ for some formula ϕ every free variable in which is declared in Γ.*
6. *If Γ is a first-order context, then $\Gamma \vdash_{\Sigma_L} M : \mathbf{Prf}[[\![\phi]\!]]$ iff Γ is correct, ϕ is a formula every free variable in which is declared in Γ, and M is a derivation of ϕ every free variable and occurrence marker in which is declared in Γ.*
7. *Every judgement of the form $\Gamma \vdash_{\Sigma_L} J$ with Γ first-order derivable in $\mathbf{SPar}\,(2)^-$ is of one of the above forms.*

Proof 1 follows from the fact that Σ_L sig is derivable in TF and the conservativity result for $\mathbf{SPar}\,(2)^-$ in TF.

A simple induction, using Theorem 4.5.2, shows that every correct context Γ is valid under $\mathbf{SPar}\,(2)^-$ in TF. The right-to-left direction of part 2 follows by the conservativity result. The right-to-left directions of parts 4 – 6 now follow from Theorem 4.5, the Weakening Lemma for TF (allowing the context in each case to be replaced by any correct rearrangement) and conservativity. The right-to-left direction of part 3 follows by the rule (const).

The left-to-right directions of parts 2 – 6 are proven simultaneously by induction on the derivation in $\mathbf{SPar}\,(2)^-$. No case is difficult.

For part 7, the result is covered by part 2 if $J \equiv$ valid, or part 3 if $J \equiv M : \mathbf{Type}$. If $J \equiv M : N$, then $\Gamma \vdash_{\Sigma_L} N : \mathbf{Type}$ by Type Validity. Therefore, by part 3, $N \equiv$ term, $N \equiv$ form, or $N \equiv \mathbf{Prf}[[\![\phi]\!]]$ for some formula ϕ, every free variable in which is declared in Γ. The result now follows from part 4, 5 or 6 respectively. $\square$

Again, it would have been possible, and not at all difficult, to prove this theorem directly for $\mathbf{SPar}\,(2)^-$, and deduce the result for TF.

5.6 Extended Natural Deduction

Extended natural deduction uses parameters of small type of every order (and no equations), so the appropriate subsystem of TF is $\mathbf{SPar}\,(\omega)^-$. Let Σ be the first part of the signature representing predicate logic, up to $\mathbf{Prf}$: (form)$\mathbf{Type}$. Note that this is a signature of $\mathbf{SPar}\,(\omega)^-$; in fact, it is a signature of $\mathbf{SPar}\,(2)^-$.

As usual, we begin by defining the set of *correct* contexts as follows:

- $\langle\rangle$ is correct.
- If Γ is correct, and x is an individual variable not declared in Γ, then $\Gamma, x :$ term is correct.
- If Γ is correct, and p is a propositional variable not declared in Γ, then $\Gamma, p :$ form is correct.
- Let R be the rule of deduction
$$\frac{R_1 \quad \cdots \quad R_n}{\phi}$$
where R_i is of arity α_i. If
$$\Gamma, p_1 : [\![R_1]\!]\,, \ldots, p_n : [\![R_n]\!]$$
is correct, every free variable in R is declared in Γ, and q is a occurrence marker for R not declared in Γ, then
$$\Gamma, q : [\![R]\!]$$
is correct.

Once again, we shall assume that every variable in a context declaration is from the appropriate class. Our adequacy result is now:

Theorem 5.14 *In* $\mathbf{SPar}\,(\omega)^-$*, assume that the only variables declared in* Γ *of higher order than base are occurrence markers. Then:*

1. Σ sig.
2. $\Gamma \vdash_\Sigma$ valid *iff* Γ *is correct.*
3. $\Gamma \vdash_\Sigma M : \mathbf{Type}$ *iff* Γ *is correct, and either* $M \equiv \mathrm{term}$*,* $M \equiv \mathrm{form}$*, or* $M \equiv \mathbf{Prf}[\phi]$ *for some formula* ϕ *every free variable in which is declared in* Γ*.*
4. $\Gamma \vdash_\Sigma M : \mathrm{term}$ *iff* Γ *is correct and* $M \equiv [\![t]\!]$ *for some term* t*, every free variable in which is declared in* Γ*.*
5. $\Gamma \vdash_\Sigma M : \mathrm{form}$ *iff* Γ *is correct and* $M \equiv [\![\phi]\!]$ *for some formula* ϕ*, every free variable in which is declared in* Γ*.*
6. $\Gamma \vdash_\Sigma M : \mathbf{Prf}[[\![\phi]\!]]$ *iff* Γ *is correct,* ϕ *is a formula every free variable in which is declared in* Γ*, and* $M \equiv [\![D]\!]$ *for some derivation* D *of* ϕ *every free variable in which is declared in* Γ*.*
7. *Every derivable judgement of the form* $\Gamma \vdash_{\Sigma'} J$ *is of one of the above forms.*

Proof The proof follows our usual pattern.

Σ sig is easy to establish.

A simple induction shows that $\Gamma \vdash_\Sigma$ valid for every correct Γ. This yields the right-to-left direction of part 2. The right-to-left direction of parts 4–6 follow from Theorem 4.6 and conservativity. The right-to-left direction of part 3 follows by Weakening and (const) for **Prf**.

The left-to-right directions of parts $2-6$ are proven simultaneously by induction on the derivation in $\mathbf{SPar}\,(\omega)^-$.

Part 7 follows by the usual analysis on cases provided by part 3. □

5.6.1 Martin-Löf Type Theory

For Martin-Löf Type Theory, the appropriate subsystem of TF is $\mathbf{LPar}\,(3)$.

This time, rather than defining the set of correct contexts, we shall define a translation from the contexts of Martin-Löf Type Theory into the contexts of $\mathbf{LPar}\,(3)$. The range of this translation shall then play the role that the correct contexts did for our previous object theories.

The set of *contexts* of Martin-Löf Type Theory is given as follows:

- $\langle\rangle$ is a context.
- If x is a variable of arity $\overbrace{\mathbf{0} \otimes \cdots \otimes \mathbf{0}}^{n} \twoheadrightarrow \mathbf{0}$, and

  $$B \text{ type } [\Gamma, x_1 \in A_1, \ldots, x_n \in A_n]$$

  then $\Gamma, x \in B\ [x_1 \in A_1, \ldots, x_n \in A_n]$ is a context.

This latter clause handles both the declaration of base variables (when $n = 0$) and first-order variables (when $n > 0$).

The translation into contexts of **LPar** (3) is just as might be expected:

$$\begin{aligned} [\![\langle\rangle]\!] &\equiv \langle\rangle \\ [\![\Gamma, x \in B\,[x_1 \in A_1, \ldots, x_n \in A_n]]\!] &\equiv [\![\Gamma]\!]\,, x : (x_1 : [\![A_1]\!]\,, \ldots, x_n : [\![A_n]\!])\,[\![B]\!] \end{aligned}$$

Our adequacy theorem has the following form:

Theorem 5.15 *Let Γ be a context in which no declaration is second-order, and every declaration is in a small product kind. In* **LPar** (3)*:*

1. Σ_{ML} sig.
2. $\Gamma \vdash_{\Sigma_{ML}}$ valid *iff* $\Gamma \equiv [\![\Delta]\!]$ *for some context* Δ *of Martin-Löf Type Theory.*
3. $\Gamma \vdash_{\Sigma_{ML}} M :$ **Type** *iff* $\Gamma \equiv [\![\Delta]\!]$ *and* $M \equiv [\![A]\!]$ *for some* Δ*,* A *such that*
$$A \text{ type } [\Delta]\,.$$
4. $\Gamma \vdash_{\Sigma_{ML}} M : [\![A]\!]$ *iff* $\Gamma \equiv [\![\Delta]\!]$ *and* $M \equiv [\![a]\!]$ *for some* Δ*,* a *such that*
$$a \in A\,[\Delta]\,.$$
5. *Every derivable judgement of the form* $\Gamma \vdash_{\Sigma_{ML}} J$ *is of one of the above forms.*

Proof Σ_{ML} sig is easy to establish.

The left-to-right directions of parts 2 – 4 are established by an induction on the derivations in **LPar** (3); the right-to-left directions by an induction on the derivations in Martin-Löf Type Theory. No case presents much difficulty.

For part 5, if $J \equiv$ valid the result is given by part 2. If $J \equiv M :$ **Type**, the result is given by part 3. If $J \equiv M : N$, Type Validity gives $\Gamma \vdash_{\Sigma_{ML}} N :$ **Type**; therefore, by part 3, $N \equiv [\![A]\!]$ for some appropriate A. The result is now given by part 4. $\square$

5.6.2 Calculus of Constructions

In this case, the appropriate subsystem is **LPar** (2).

We map CoC contexts to **LPar** (2) contexts thus:

$$\begin{aligned} [\![\langle\rangle]\!] &\equiv \langle\rangle \\ [\![\Gamma, x : A]\!] &\equiv \Gamma, x : [\![A]\!] \end{aligned}$$

Theorem 5.16 *Let Γ be a context in which no variables of higher order than base are declared, and in which every kind is a type. In* **LPar** (2)*:*

1. Σ_{CoC} sig.

2. $\Gamma \vdash_{\Sigma_{CoC}}$ valid *iff* $\Gamma \equiv [\![\Delta]\!]$ *for some valid context* Δ *of CoC.*

3. $\Gamma \vdash_{\Sigma_{CoC}} M : \mathbf{Type}$ *iff* $\Gamma \equiv [\![\Delta]\!]$, $M \equiv [\![A]\!]$ *for some* Δ, A *such that*

$$\Delta \vdash A \text{ type}$$

4. $\Gamma \vdash_{\Sigma_{CoC}} M : [\![A]\!]$ *iff* $\Gamma \equiv [\![\Delta]\!]$, $M \equiv [\![a]\!]$ *for some* Δ, a *such that*

$$\Delta \vdash a : A$$

5. *Every derivable judgement of the form* $\Gamma \vdash_{\Sigma_{CC}} J$ *is of one of the above forms.*

Proof Σ_{CoC} sig follows from Theorem 4.15 and conservativity.

The right-to-left directions of parts 2 – 4 follow from Theorem 4.15 and conservativity. The left-to-right directions are proven simultaneously by an induction on derivations in **LPar** (2).

Part 5 is proven similarly to the final part of the adequacy theorems for any of the other object theories. □

5.6.3 Discussion

It is worth noting that only in the case of propositional logic could we state the adequacy theorem in the form we would like; in all other cases, we had to place some restriction on the contexts. For Martin-Löf Type Theory, for example, we needed the power of **LPar** (3) to build the signature Σ_{ML}, but from then on we had to use only the power of **SPar** (2) to avoid straying outside the judgements of the object theory itself. This suggests the possiblity of forming further subsystems of TF, in which a different set of rules is used for building signatures as for working within a signature. The investigation of such a possibility, however, is outside the scope of this thesis.

Chapter 6

Lambda-Free Subsystems of Existing Frameworks

We shall now show how the lambda-free systems we have built can be embedded within existing logical frameworks; that is, for many existing logical frameworks F, there is a subsystem S of F that is isomorphic to one of our lambda-free frameworks, and of which F is a conservative extension.

The meaning of the word 'isomorphic' here will be different in the case of each framework. Formulating a general notion would require first giving a general definition of 'logical framework' which could cover all our examples; while this would doubtlessly be an interesting project, it is beyond the scope of this thesis. Nevertheless, there are several features in common between the various isomorphism results that we shall prove. Each shall involve defining a translation from a weak framework into F, and showing this translation to be a derivability-preserving bijection between the judgements of the weak framework and a subset of the judgements of F. Each shall be sufficient for us to deduce, from the adequacy results of the previous chapter, that the translations of the signatures given there under these isomorphisms give an adequate representation of the object theory in F itself.

In most cases, we are able to embed the whole of TF (or **LPar** (ω)) within the framework. However, we can only embed a subsystem of **SPar** (ω) within ELF, since in that framework abstraction is only permitted over types, not large kinds, and one cannot declare computation rules. Similarly, we are only able to fit **LPar** (1) within PAL.

The proof of each result follows the same pattern. In showing how a weak frame W can be embedded in an existing frame F, we begin by providing a translation, which for the moment we denote by Φ, from the terms, kinds, etc. of the weak logical framework W into the expressions of the framework F; in particular, we associate with each judgement J of W a judgement $\Phi(J)$ of F. Proving that this translation is sound — that is, that it preserves derivability — is rarely a problem; the difficult part is proving the translation complete — that is, that if a judgement of F has the form $\Phi(J)$, then $\Phi(J)$ is derivable only when J is derivable.

This is typically done by providing an inverse translation from F into W. This cannot be done directly, as not every term in F has a counterpart in W; for example, F may contain lambda-abstractions while W does not. We cannot represent a lambda-abstraction as a term in F, but we *can* represent the terms obtained by fully applying lambda-abstraction to a sequence of arguments. Moreover, it is sufficient to consider the case where the arguments all have the form of abstractions of W.

The proof to which this idea gives rise proceeds as follows. We assign *arities* to the terms and kinds of F. This can usually be done in quite a natural way, and we find that the terms and kinds of W are always mapped (by Φ) to *base* terms and kinds.

We then define:

- For every α-ary term M of F, and every α-ary abstraction sequence $\vec{N}$ from W, the term
$$\mathrm{APP}(M, \vec{N})$$
of W. This is the term in W that corresponds to the term $M(\Phi(\vec{N}))$ in F.
- For every α-ary kind K of F, an α-ary context $\mathrm{DOM}\, K$ of W, and a kind $\mathrm{COD}\, K$ of W.

Roughly speaking, the α-ary kind K is represented in W by the product kind $(\mathrm{DOM}\, K)\, \mathrm{COD}\, K$; we are assured that $\mathrm{APP}(M, \vec{N})$ is typable provided $\vec{N} :: \mathrm{DOM}\, K$, in which case it has type $\{\vec{N}/\, \mathrm{DOM}\, K\}\, \mathrm{COD}\, K$. More formally, we are able to prove the following result:

If $\Phi(\Gamma) \vdash_{\Phi(\Sigma)} M : K$ in F, and $\Gamma \Vdash_\Sigma \vec{N} :: \mathrm{DOM}\, K$ in W, then
$$\Gamma \vdash_\Sigma \mathrm{APP}(M, \vec{N}) : \{\vec{N}/\, \mathrm{DOM}\, K\}\, \mathrm{COD}\, K$$
in W.

Recall that Φ maps the terms and kinds of W to *base* terms and kinds of F. When given a base term, APP requires no further arguments; we shall find that APP and COD form the promised inverses to Φ. That is,
$$\mathrm{APP}(\Phi(M)) \equiv M, \qquad \mathrm{COD}(\Phi(K)) \equiv K$$
for a term M and kind K of W; thus, the result above, in the case $\alpha \equiv \mathbf{0}$, becomes the conservativity result we seek:

If $\Phi(J)$ is derivable in F, then J is derivable in W.

We shall be very grateful that we have two equivalent versions of each weak framework, W and W_c. We define the translation Φ from W_c to F; it shall be useful to have the typing information provided by the terms of W_c in order to form the terms of F. Conversely, we define the functions APP, DOM and COD from F into W; it shall be highly convenient not to have to worry about providing the types in the abstractions when forming terms in W.

6.1 The Edinburgh Logical Framework, ELF

This framework was originally presented in [HHP93], the paper in which the phrase 'logical framework' was first coined. Considered as a type system, it is one of the members of the lambda-cube; λP, to be precise.

We give a presentation here that uses a judgement form for equality of objects, families and kinds, rather than using the external notion of β-convertibility. This is equivalent to the form in which ELF is usually presented and implemented; see Appendix B for the precise statement and proof of this claim, as well as several results in the metatheory of the following system.

Grammar

$$
\begin{array}{lcll}
\text{Kind} & K & ::= & \mathbf{Type} \mid \Pi x : A.K \\
\text{Family} & A & ::= & a \mid \Pi x : A.A \mid \lambda x : A.A \mid AM \\
\text{Object} & M & ::= & c \mid x \mid \lambda x : A.M \mid MM
\end{array}
$$

A *signature* in ELF is a sequence of declarations of the form $a : K$, where a is a family-level constant and K a kind, or $c : A$, where c is an object-level constant and A a family. A *context* is a sequence of declarations $x : A$, where x is a variable and A a family.

The judgement forms in ELF are:

$$
\begin{array}{lll}
\Sigma & & \text{sig} \\
\Gamma & \vdash_\Sigma & \cdot \\
\Gamma & \vdash_\Sigma & K \\
\Gamma & \vdash_\Sigma & K_1 = K_2 \\
\Gamma & \vdash_\Sigma & A : K \\
\Gamma & \vdash_\Sigma & A = B : K \\
\Gamma & \vdash_\Sigma & M : A \\
\Gamma & \vdash_\Sigma & M = N : A
\end{array}
$$

Defined Operations We define the relation of *β-convertibility*, both between families and between objects, as usual based on the contractions:

$$
\begin{array}{rcl}
(\lambda x : A.B)M & \rightsquigarrow & [M/x]B \\
(\lambda x : A.M)N & \rightsquigarrow & [N/x]M
\end{array}
$$

Rules of Deduction The rules of deduction of ELF are as follows.

Valid Signatures

$$
(\texttt{B-EMPTY-SIG}) \qquad \frac{}{\langle\rangle \ \text{sig}}
$$

$$
(\texttt{B-KIND-SIG}) \qquad \frac{\Sigma \ \text{sig} \quad \vdash_\Sigma K}{\Sigma, a : K \ \text{sig}} \qquad (a \notin \operatorname{dom} \Sigma)
$$

$$
(\texttt{B-TYPE-SIG}) \qquad \frac{\Sigma \ \text{sig} \quad \vdash_\Sigma A : \mathbf{Type}}{\Sigma, c : A \ \text{sig}} \qquad (c \notin \operatorname{dom} \Sigma)
$$

Valid Contexts

(B-EMPTY-CTX) $$\frac{\Sigma \text{ sig}}{\vdash_\Sigma \cdot}$$

(B-TYPE-CTX) $$\frac{\Gamma \vdash_\Sigma \cdot \quad \Gamma \vdash_\Sigma A : \mathbf{Type}}{\Gamma, x : A \vdash_\Sigma \cdot} \quad (x \notin \operatorname{dom} \Gamma)$$

Valid Kinds

(B-TYPE-KIND) $$\frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma \mathbf{Type}}$$

(B-PI-KIND) $$\frac{\Gamma, x : A \vdash_\Sigma K}{\Gamma \vdash_\Sigma \Pi x : A.K}$$

(B-PI-KIND-EQ) $$\frac{\Gamma, \vdash_\Sigma A = A' : \mathbf{Type} \quad \Gamma, x : A \vdash_\Sigma K = K'}{\Gamma \vdash_\Sigma \Pi x : A.K = \Pi x : A'.K'}$$

Valid Families

(B-CONST-FAM) $$\frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma a : K} \quad (a : K \in \Sigma)$$

(B-PI-FAM) $$\frac{\Gamma, x : A \vdash_\Sigma B : \mathbf{Type}}{\Gamma \vdash_\Sigma \Pi x : A.B : \mathbf{Type}}$$

(B-PI-FAM-EQ) $$\frac{\Gamma \vdash_\Sigma A = A' : \mathbf{Type} \quad \Gamma, x : A \vdash_\Sigma B = B' : \mathbf{Type}}{\Gamma \vdash_\Sigma \Pi x : A.B = \Pi x : A'.B' : \mathbf{Type}}$$

(B-ABS-FAM) $$\frac{\Gamma, x : A \vdash_\Sigma B : K}{\Gamma \vdash_\Sigma \lambda x : A.B : \Pi x : A.K}$$

(B-ABS-FAM-EQ) $$\frac{\Gamma \vdash_\Sigma A = A' : \mathbf{Type} \quad \Gamma, x : A \vdash_\Sigma B = B' : K}{\Gamma \vdash_\Sigma \lambda x : A.B = \lambda x : A'.B' : \Pi x : A.K}$$

(B-APP-FAM) $$\frac{\Gamma \vdash_\Sigma A : \Pi x : B.K \quad \Gamma \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma AM : [M/x]K}$$

(B-APP-FAM-EQ) $$\frac{\Gamma \vdash_\Sigma A = A' : \Pi x : B.K \quad \Gamma \vdash_\Sigma M = M' : B}{\Gamma \vdash_\Sigma AM = A'M' : [M/x]K}$$

(B-CONV-FAM) $$\frac{\Gamma \vdash_\Sigma A : K \quad \Gamma \vdash_\Sigma K = K'}{\Gamma \vdash_\Sigma A : K'}$$

(B-CONV-FAM-EQ) $$\frac{\Gamma \vdash_\Sigma A = A' : K \quad \Gamma \vdash_\Sigma K = K'}{\Gamma \vdash_\Sigma A = A' : K'}$$

Valid Objects

$$\text{(B-CONST-OBJ)} \quad \frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma c : A} \quad (c : A \in \Sigma)$$

$$\text{(B-VAR-OBJ)} \quad \frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma x : A} \quad (x : A \in \Gamma)$$

$$\text{(B-ABS-OBJ)} \quad \frac{\Gamma, x : A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda x : A.M : \Pi x : A.B}$$

$$\text{(B-ABS-OBJ-EQ)} \quad \frac{\Gamma \vdash_\Sigma A = A' : \mathbf{Type} \quad \Gamma, x : A \vdash_\Sigma M = M' : B}{\Gamma \vdash_\Sigma \lambda x : A.M = \lambda x : A'.M' : \Pi x : A.B}$$

$$\text{(B-APP-OBJ)} \quad \frac{\Gamma \vdash_\Sigma M : \Pi x : A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : [N/x]B}$$

$$\text{(B-APP-OBJ-EQ)} \quad \frac{\Gamma \vdash_\Sigma M = M' : \Pi x : A.B \quad \Gamma \vdash_\Sigma N = N' : A}{\Gamma \vdash_\Sigma MN = M'N' : [N/x]B}$$

$$\text{(B-CONV-OBJ)} \quad \frac{\Gamma \vdash_\Sigma M : A \quad \Gamma \vdash_\Sigma A = A' : \mathbf{Type}}{\Gamma \vdash_\Sigma M : A'}$$

$$\text{(B-CONV-OBJ-EQ)} \quad \frac{\Gamma \vdash_\Sigma M = M' : A \quad \Gamma \vdash_\Sigma A = A' : \mathbf{Type}}{\Gamma \vdash_\Sigma M = M' : A'}$$

Equality

$$(\texttt{B-REF-KIND}) \quad \frac{\Gamma \vdash_\Sigma K}{\Gamma \vdash_\Sigma K = K}$$

$$(\texttt{B-SYM-KIND}) \quad \frac{\Gamma \vdash_\Sigma K = K'}{\Gamma \vdash_\Sigma K' = K}$$

$$(\texttt{B-TRANS-KIND}) \quad \frac{\Gamma \vdash_\Sigma K = K' \quad \Gamma \vdash_\Sigma K' = K''}{\Gamma \vdash_\Sigma K = K''}$$

$$(\texttt{B-REF-FAM}) \quad \frac{\Gamma \vdash_\Sigma A : K}{\Gamma \vdash_\Sigma A = A : K}$$

$$(\texttt{B-SYM-FAM}) \quad \frac{\Gamma \vdash_\Sigma A = A' : K}{\Gamma \vdash_\Sigma A' = A : K}$$

$$(\texttt{B-TRANS-FAM}) \quad \frac{\Gamma \vdash_\Sigma A = A' : K \quad \Gamma \vdash_\Sigma A' = A'' : K}{\Gamma \vdash_\Sigma A = A'' : K}$$

$$(\texttt{B-REF-OBJ}) \quad \frac{\Gamma \vdash_\Sigma M : A}{\Gamma \vdash_\Sigma M = M : A}$$

$$(\texttt{B-SYM-OBJ}) \quad \frac{\Gamma \vdash_\Sigma M = M' : A}{\Gamma \vdash_\Sigma M' = M : A}$$

$$(\texttt{B-TRANS-OBJ}) \quad \frac{\Gamma \vdash_\Sigma M = M' : A \quad \Gamma \vdash_\Sigma M' = M'' : A}{\Gamma \vdash_\Sigma M = M'' : A}$$

$$(\texttt{B-BETA-FAM}) \quad \frac{\Gamma, x : A \vdash_\Sigma B : K \quad \Gamma \vdash_\Sigma M : A}{\Gamma \vdash_\Sigma (\lambda x : A.B)M = [M/x]B : [M/x]K}$$

$$(\texttt{B-BETA-OBJ}) \quad \frac{\Gamma, x : A \vdash_\Sigma N : B \quad \Gamma \vdash_\Sigma M : A}{\Gamma \vdash_\Sigma (\lambda x : A.N)M = [M/x]N : [M/x]B}$$

Let us say that an ELF signature Σ is *consistent* iff

- For every entry $a : K$ in Σ, say $\Sigma \equiv \Sigma_0, a : K, \Sigma_1$, then K has no free variables, and every constant in K is declared in Σ_0.
- For every entry $c : A$ in Σ, say $\Sigma \equiv \Sigma_0, c : A, \Sigma_1$, then A has no free variables, and every constant in A is declared in Σ_0.

Likewise, we say that an ELF context Γ is consistent iff, for every entry $x : A$ in Γ, say $\Gamma \equiv \Gamma_0, x : A, \Gamma_1$, then every free variable in A is declared in Γ_0.

We shall make use of the following abbreviations. If Δ is an ELF context, say

$$\Delta \equiv x_1 : A_1, \ldots, x_n : A_n ,$$

we write

$$\begin{aligned}\lambda\Delta.X &\equiv \lambda x_1 : A_1.\cdots.\lambda x_n : A_n.X \qquad (X \text{ an object or family})\\ \Pi\Delta.X &\equiv \Pi x_1 : A_1.\cdots.\Pi x_n : A_n.X \qquad (X \text{ a family or kind})\end{aligned}$$

If

$$\overrightarrow{M} \equiv M_1, \ldots, M_n$$

is a sequence of n objects, and

$$\Delta \equiv x_1 : A_1, \ldots, x_n : A_n$$

a context of length n, we write

$$\Gamma \Vdash_\Sigma \overrightarrow{M} : \Delta$$

for the set of n judgements

$$\begin{aligned}\Gamma &\vdash_\Sigma M_1 : A_1\\ \Gamma &\vdash_\Sigma M_2 : [M_1/x_1]A_1\\ &\vdots\\ \Gamma &\vdash_\Sigma M_n : [M_1/x_1, \ldots, M_{n-1}/x_{n-1}]A_n\end{aligned}$$

The following lemma is easy to prove:

Lemma 6.1 *If $\Gamma \Vdash_\Sigma \overrightarrow{M} : \Delta$, and $\Gamma \vdash_\Sigma X : \Pi\Delta.Y$, then*

$$\Gamma \vdash_\Sigma X\overrightarrow{M} : [\overrightarrow{M}/\operatorname{dom}\Delta]Y\,.$$

(Here, either X is an object and Y a family, or X is a family and Y a kind.)

Proof n applications of the rule (`B-APP-OBJ`) or (`B-APP-FAM`). □

6.1.1 SPar $(\omega)_c^-$ as a subsystem of ELF

We proceed to embed **SPar** $(\omega)_c^-$ in ELF. Specifically, we shall show that the judgements of **SPar** $(\omega)_c^-$ are essentially those judgements of ELF that are in normal form. Thanks to the equivalence between **SPar** $(\omega)^-$ and **SPar** $(\omega)_c^-$, a similar conclusion shall follow for **SPar** $(\omega)^-$.

The embedding is just as might be expected, and consists essentially of replacing

$$[x_1 : K_1, \ldots, x_n : K_n]M \text{ with } \lambda x_1 : K_1.\cdots.\lambda x_n : K_n.M$$

and

$$(x_1 : K_1, \ldots, x_n : K_n)K \text{ with } \Pi x_1 : K_1.\cdots.\Pi x_n : K_n.K\,.$$

We shall assume that the set of variables used for ELF is the same as the set $\bigcup_{\alpha in \mathcal{A}} \mathcal{V}^\alpha$ for **SPar** $(\omega)_c^-$, and the set of constants (for families or objects) is the same as $\bigcup_{\alpha\in\mathcal{A}} \mathcal{C}^\alpha$. Further, we shall assume that, whenever a declaration $c^\alpha : (\Delta^\alpha)K$ is made in $\mathbf{SPar}\,(\omega)_c^-$, the constant c^α involved is an object-level constant of ELF if K is a term, and a family-level constant if $K \equiv \mathbf{Type}$.

The embedding is a partial function; not every raw expression of $\mathbf{SPar}\,(\omega)_c^-$ has a counterpart in ELF. (For example, if $c^{\mathbf{1}}$ is a $\mathbf{1}$-ary family-level constant, and $d^{\mathbf{0}}$ is a base family-level constant, then $c^{\mathbf{1}}[d^{\mathbf{0}}]$ has no counterpart in ELF. We cannot form the family $c^{\mathbf{1}}d^{\mathbf{0}}$, because $d^{\mathbf{0}}$ is a family, not an object.) However, we shall show that it is defined on every well-typed term of $\mathbf{SPar}\,(\omega)_c^-$.

The definition of the embedding is as follows.

Definition 6.2 *We define simultaneously:*

- *for a term M of $\mathbf{SPar}\,(\omega)_c^-$, an ELF expression*

$$\langle\!| M |\!\rangle$$

  that is either an object or a family.

- *for an abstraction M of $\mathbf{SPar}\,(\omega)_c^-$, an ELF expression*

$$\langle\!| M |\!\rangle$$

  that is either an object or a family.

- *for a context Δ of $\mathbf{SPar}\,(\omega)_c^-$, an ELF context*

$$\langle\!| \Delta |\!\rangle$$

The definition is as follows:

- $$\left\langle\!\left| c^\alpha[\overrightarrow{M}^\alpha] \right|\!\right\rangle \equiv c^\alpha \left\langle\!\left| \overrightarrow{M}^\alpha \right|\!\right\rangle$$

  This shall be an object if c^α is an object-level constant, and a family if c^α is a family-level constant. It is defined iff each $\langle\!| M^{\alpha_i} |\!\rangle_\Sigma^\Gamma$ is defined and is an object.

- $$\left\langle\!\left| x^\alpha[\overrightarrow{M}^\alpha] \right|\!\right\rangle \equiv x^\alpha \left\langle\!\left| \overrightarrow{M}^\alpha \right|\!\right\rangle$$

  This shall be an object. It is defined iff each $\langle\!| M^{\alpha_i} |\!\rangle_\Sigma^\Gamma$ is defined and is an object.

- $$\langle\!| [\Delta]M |\!\rangle \equiv \lambda \langle\!| \Delta |\!\rangle \,.\, \langle\!| M |\!\rangle$$

  This shall be an object if $\langle\!| M |\!\rangle$ is an object, and a family if $\langle\!| M |\!\rangle$ is a family. It is defined iff each expression in the right-hand side is defined.

- $$\langle\!| \langle\rangle |\!\rangle \equiv \langle\rangle$$

- $$\langle\!| \Theta, x^\alpha : (\Delta^\alpha) M |\!\rangle \equiv \langle\!| \Theta |\!\rangle, x^\alpha : \Pi \langle\!| \Delta^\alpha |\!\rangle . \langle\!| M |\!\rangle$$

  This is defined iff the expressions in the right-hand side are defined and $\langle\!| M |\!\rangle^\Gamma_\Sigma$ *is a family.*

We extend this function to all the kinds of $\mathbf{SPar}(\omega)^-$ *by setting*

$$\langle\!| \mathbf{Type} |\!\rangle \equiv \mathbf{Type}$$

We now define, for signatures Σ *of* $\mathbf{SPar}(\omega)^-$*, an ELF signature* $\langle\!| \Sigma |\!\rangle$ *as follows:*

$$\begin{array}{rcl} \langle\!| \langle\rangle |\!\rangle & \equiv & \langle\rangle \\ \langle\!| \Sigma, c^\alpha : (\Delta^\alpha) K |\!\rangle & \equiv & \langle\!| \Sigma |\!\rangle, c^\alpha : \Pi \langle\!| \Delta^\alpha |\!\rangle . \langle\!| K |\!\rangle \end{array}$$

This latter signature is defined iff each expression on the right-hand side is defined.

Finally, we define the translation on judgements by

$$\begin{array}{rcl} \langle\!| \Sigma \text{ sig} |\!\rangle & \equiv & \langle\!| \Sigma |\!\rangle \text{ sig} \\ \langle\!| \Gamma \vdash_\Sigma \text{valid} |\!\rangle & \equiv & \langle\!| \Gamma |\!\rangle \vdash_{\langle\!| \Sigma |\!\rangle} \cdot \\ \langle\!| \Gamma \vdash_\Sigma M : K |\!\rangle & \equiv & \langle\!| \Gamma |\!\rangle \vdash_{\langle\!| \Sigma |\!\rangle} \langle\!| M |\!\rangle : \langle\!| K |\!\rangle \end{array}$$

The first two are defined iff each expression on the right is defined. The last is defined iff every expression on the right-hand side is defined, and either $\langle\!| M |\!\rangle$ *is an object and* $\langle\!| K |\!\rangle$ *a family,* or $\langle\!| M |\!\rangle$ *is a family and* $\langle\!| K |\!\rangle$ *a kind.*

The following lemma shows how instantiation in $\mathbf{SPar}(\omega)^-_c$ and substitution in ELF are connected via this translation:

Lemma 6.3 *Let* $\vec{x}^\alpha$ *be an* α*-ary variable sequence, and* $\overrightarrow{M}^\alpha$ *an* α*-ary abstraction sequence. Let* X *be an expression; either a term, abstraction, kind, context, signature or judgement of* $\mathbf{SPar}(\omega)^-_c$*. If* $\langle\!| X |\!\rangle$ *is defined, and each member of* $\left\langle\!\left| \overrightarrow{M}^\alpha \right|\!\right\rangle$ *is defined and is an object, then*

$$\left\langle\!\left| \{\overrightarrow{M}^\alpha / \vec{x}^\alpha\} X \right|\!\right\rangle$$

is defined and is in the same syntactic class as $\langle\!| X |\!\rangle$*. Further,*

$$\left[\left\langle\!\left| \overrightarrow{M}^\alpha \right|\!\right\rangle / \vec{x}^\alpha\right] \langle\!| X |\!\rangle \twoheadrightarrow_\beta \left\langle\!\left| \{\overrightarrow{M}^\alpha / \vec{x}^\alpha\} X \right|\!\right\rangle$$

Proof The proof is by double induction, firstly on the arity α, secondly on X following the structure of the recursion in the definition of $\langle\!| \; |\!\rangle$ (that is, we prove the cases that X is a term, abstraction and context simultaneously, then prove the result for X a kind, a signature, and then a judgement). The only difficult case is that where X is a term beginning with one of the members of $\vec{x}^\alpha$.

Suppose $X \equiv x^{\alpha_i}[\overrightarrow{N}^{\alpha_i}]$, and let the ith member of $\overrightarrow{M}^\alpha$ be

$$[\Delta^{\alpha_i}] M_i$$

We must show that

$$\left\langle\!\left| \left\{ \left\{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \right\} \overrightarrow{N}^{\alpha_i} / \Delta^{\alpha_i} \right\} M_i \right|\!\right\rangle$$

is defined and is an object, and that

$$\left[\left\langle\!\left| \overrightarrow{M}^{\alpha} \right|\!\right\rangle / \vec{x}^{\alpha} \right] x^{\alpha_i} \left\langle\!\left| \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle \twoheadrightarrow_{\beta} \left\langle\!\left| \{ \{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \} \overrightarrow{N}^{\alpha_i} / \Delta^{\alpha_i} \} M_i \right|\!\right\rangle$$

We are given that $\langle\!| X |\!\rangle$ is defined, hence each of $\left\langle\!\left| \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle$ is defined and is an object. Therefore, by the induction hypothesis on X, each member of

$$\left\langle\!\left| \{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \} \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle$$

is defined and is an object. By the induction hypothesis on α, therefore,

$$\left\langle\!\left| \{ \{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \} \overrightarrow{N}^{\alpha_i} / \Delta^{\alpha_i} \} M_i \right|\!\right\rangle$$

is defined and is an object, as required. (We know $\langle\!| M_i |\!\rangle$ is an object because we are given that $\langle\!| M^{\alpha_i} |\!\rangle$ is.)

Now,

$$\begin{array}{rcll}
\left[\left\langle\!\left| \overrightarrow{M}^{\alpha} \right|\!\right\rangle / \vec{x}^{\alpha} \right] x^{\alpha_i} \left\langle\!\left| \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle & \equiv & \lambda \langle\!| \Delta^{\alpha_i} |\!\rangle \,.\, \langle\!| M_i |\!\rangle \, [\overrightarrow{M}^{\alpha} / \vec{x}^{\alpha}] \left\langle\!\left| \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle & \\
& \twoheadrightarrow_{\beta} & \lambda \langle\!| \Delta^{\alpha_i} |\!\rangle \,.\, \langle\!| M_i |\!\rangle \left\langle\!\left| \{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \} \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle & \text{(i.h. on } X) \\
& \twoheadrightarrow_{\beta} & \left[\left\langle\!\left| \{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \} \overrightarrow{N}^{\alpha_i} \right|\!\right\rangle / \Delta^{\alpha_i} \right] \langle\!| M_i |\!\rangle & \\
& \twoheadrightarrow_{\beta} & \left\langle\!\left| \{ \{ \overrightarrow{M}^{\alpha} / \vec{x}^{\alpha} \} \overrightarrow{N}^{\alpha_i} / \Delta^{\alpha_i} \} M_i \right|\!\right\rangle & \text{(i.h. on } \alpha)
\end{array}$$

□

Proving this translation sound is quite simple:

Theorem 6.4 *If J is derivable in* $\mathbf{SPar}\,(\omega)^{-}$, *then* $\langle\!| J |\!\rangle$ *is derivable in ELF.*

Proof An induction on the derivation of J. We deal with one case here:

$$(\mathrm{var}_{\alpha}) \, \frac{\Gamma \Vdash_{\Sigma} \overrightarrow{M}^{\alpha} :: \Delta^{\alpha}}{\Gamma \vdash_{\Sigma} x^{\alpha}[\overrightarrow{M}^{\alpha}] : \{ \overrightarrow{M}^{\alpha} / \Delta^{\alpha} \} N} \, (x^{\alpha} : (\Delta^{\alpha}) N \in \Gamma)$$

Let

$$\begin{array}{rcl}
\Delta^{\alpha} & \equiv & x^{\alpha_1} : (\Delta^{\alpha_1}) P_1, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n}) P_n \\
\overrightarrow{M}^{\alpha} & \equiv & [\Delta^{\alpha_1}] M_1, [(\Delta^{\alpha_2})^*] M_2, \ldots, [(\Delta^{\alpha_n})^*] M_n
\end{array}$$

Then the ith member of $\Gamma \Vdash_{\Sigma} \overrightarrow{M}^{\alpha} :: \Delta^{\alpha}$ is

$$\Gamma, (\Delta^{\alpha_i})^* \vdash_{\Sigma} M_i : P_i^*$$

where, as always, * indicates instantiation of Δ^α with $\overrightarrow{M}^\alpha$.

Claim We shall prove that

$$\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} \langle\!|M^{\alpha_i}|\!\rangle : [\langle\!|M^{\alpha_1}|\!\rangle / x^{\alpha_1}, \ldots, \langle\!|M^{\alpha_{i-1}}|\!\rangle / x^{\alpha_{i-1}}]\Pi\,\langle\!|\Delta^{\alpha_i}|\!\rangle \,.\, \langle\!|P_i|\!\rangle$$

for $i = 1, \ldots, n$ by induction on i.

Proof The induction hypothesis for the derivation gives us

$$\begin{array}{rll} \langle\!|\Gamma|\!\rangle , \langle\!|(\Delta^{\alpha_i})^*|\!\rangle & \vdash_{\langle\!|\Sigma|\!\rangle} & \langle\!|M_i|\!\rangle : \langle\!|P_i^*|\!\rangle \\ \therefore \langle\!|\Gamma|\!\rangle & \vdash_{\langle\!|\Sigma|\!\rangle} & \langle\!|M^{\alpha_i}|\!\rangle : \Pi\,\langle\!|(\Delta^{\alpha_i})^*|\!\rangle \,.\, \langle\!|P_i^*|\!\rangle \qquad (\mathtt{B-LAM-OBJ}) \end{array}$$

Now, $x^\alpha : \Pi\,\langle\!|\Delta^\alpha|\!\rangle \,.\, \langle\!|N|\!\rangle$ is in $\langle\!|\Gamma|\!\rangle$. Context Validity for ELF (Lemma B.1) gives us

$$\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} \Pi\,\langle\!|\Delta^\alpha|\!\rangle \,.\, \langle\!|N|\!\rangle : \mathbf{Type}$$

Hence Generation gives

$$\langle\!|\Gamma|\!\rangle , x^{\alpha_1} : \Pi\,\langle\!|\Delta^{\alpha_1}|\!\rangle \,.\, \langle\!|P_1|\!\rangle , \ldots, x^{\alpha_{i-1}} : \Pi\,\langle\!|\Delta^{\alpha_{i-1}}|\!\rangle \,.\, \langle\!|P_{i-1}|\!\rangle , \langle\!|\Delta^{\alpha_i}|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} \langle\!|P_i|\!\rangle : \mathbf{Type}$$

Therefore, by the induction hypothesis on i and the Substitution Lemma (Lemma B.3) $i-1$ times,

$$\begin{array}{rll} \langle\!|\Gamma|\!\rangle , \left[\left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle / \vec{x}^\alpha\right] \langle\!|\Delta^{\alpha_i}|\!\rangle & \vdash_{\langle\!|\Sigma|\!\rangle} & \left[\left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle / \vec{x}^\alpha\right] \langle\!|P_i|\!\rangle : \mathbf{Type} \\ \therefore \langle\!|\Gamma|\!\rangle & \vdash_{\langle\!|\Sigma|\!\rangle} & \left[\left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle / \vec{x}^\alpha\right] \Pi\,\langle\!|\Delta^{\alpha_i}|\!\rangle \,.\, \langle\!|P_i|\!\rangle : \mathbf{Type} \qquad (\mathtt{B\text{-}PI\text{-}FAM}) \end{array}$$

The previous lemma gives

$$\left[\left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle / \vec{x}^\alpha\right] \Pi\,\langle\!|\Delta^{\alpha_i}|\!\rangle \,.\, \langle\!|P_i|\!\rangle \twoheadrightarrow_\beta \Pi\,\langle\!|(\Delta^{\alpha_i})^*|\!\rangle \,.\, \langle\!|P_i^*|\!\rangle$$

and so Predicate Expansion (Lemma B.40) gives

$$\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} \langle\!|M^{\alpha_i}|\!\rangle : \Pi\,\langle\!|(\Delta^{\alpha_i})^*|\!\rangle \,.\, \langle\!|P_i^*|\!\rangle$$

as required. □

This claim shows that

$$\langle\!|\Gamma|\!\rangle \Vdash_{\langle\!|\Sigma|\!\rangle} \left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle :: \langle\!|\Delta^\alpha|\!\rangle$$

Now, (B-VAR-OBJ) gives us

$$\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} x^\alpha : \Pi\,\langle\!|\Delta^\alpha|\!\rangle \,.\, \langle\!|N|\!\rangle \ .$$

Hence (B-APP-OBJ) n times yields

$$\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} x^\alpha \left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle : \left[\left\langle\!\left|\overrightarrow{M}^\alpha\right|\!\right\rangle / \operatorname{dom}\Delta^\alpha\right] \langle\!|N|\!\rangle$$

Predicate Reduction (Lemma B.39) together with the previous lemma finally gives us

$$\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} x^\alpha \left\langle\left|\overrightarrow{M}^\alpha\right|\right\rangle : \left\langle\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}N\right|\right\rangle$$

as required. □

Proving the translation complete is more difficult. We proceed in the following manner.

Firstly, we assign arities to the entities of ELF. We assign one arity to kinds and objects, and two arities to families. The intended meaning of these arities is as follows:

- An α-ary object is an α-ary function whose values are entities of the object theory.
- An $\langle\alpha, \beta\rangle$-ary family is an α-ary function whose values are types of β-ary objects.
- An α-ary kind is a kind of $(\alpha, \mathbf{0})$-ary families.

Recall our simplifying assumption that ELF and $\mathbf{SPar}\,(\omega)_c^-$ use the same sets of variables and constants, hence every variable and constant of ELF has an arity. This will be used to give arities to variables and constants when they appear as families or objects; however, we shall need to assume that if $x : A$ occurs in a context, and A is a $\langle\mathbf{0}, \beta\rangle$-ary family, then x is a β-ary variable; similarly for constants with families or kinds. (An alternative would be to have assumed no connection between the two systems' sets of variables and constants, and instead, when translating from one system to the other, to simply choose a fresh variable or constant of the second system whenever it is needed. In this case, we would assign arities to kinds, families and objects relative to a given context and signature. These arities would then determine the arities of the constants or variables of $\mathbf{SPar}\,(\omega)_c^-$ chosen.)

The assignment of arities to the entities of ELF is made as follows. Note that not every piece of raw grammar will have an arity assigned to it; but we shall show that every well-typed expression does.

- **Type** is a base kind.
- If A is an $\langle\mathbf{0}, \alpha\rangle$-ary family, and K a β-ary kind, then $\Pi x : A.K$ is an $(\alpha)\hat{}\beta$-ary kind.
- If a is an α-ary family-level constant, then a is an $\langle\alpha, \mathbf{0}\rangle$-ary family.
- If A is an $\langle\mathbf{0}, \alpha\rangle$-ary family, and B an $\langle\mathbf{0}, \gamma\rangle$-ary family, then $\Pi x : A.B$ is an $\langle\mathbf{0}, (\alpha)\hat{}\beta\rangle$-ary family.
- If A is an $\langle\mathbf{0}, \alpha\rangle$-ary family, and B an $\langle\beta, \gamma\rangle$-ary family, then $\lambda x : A.B$ is an $\langle(\alpha)\hat{}\beta, \gamma\rangle$-ary family.
- If A is an $\langle(\alpha)\hat{}\beta, \gamma\rangle$-ary family, and M an α-ary object, then AM is an $\langle\beta, \gamma\rangle$-ary family.
- If c is an α-ary object-level constant, then c is an α-ary object.
- If x is an α-ary variable, then x is an α-ary object.
- If A is an $\langle\mathbf{0}, \alpha\rangle$-ary family, and M a β-ary object, then $\lambda x : A.M$ is an $(\alpha)\hat{}\beta$-ary object.
- If M is an $(\alpha)\hat{}\beta$-ary object, and N an α-ary object, then MN is a β-ary object.

The assumption on ELF that we must make is as follows:

> Whenever the rule (`B-KIND-SIG`) is used, if K is an α-ary kind, then a is an α-ary family-level constant.
>
> Whenever the rule (`B-TYPE-SIG`) is used, if A is an $\langle \mathbf{0}, \alpha \rangle$-ary family, then c is an α-ary object-level constant.
>
> Whenever the rule (`B-TYPE-CTX`) is used, if A is an $\langle \mathbf{0}, \alpha \rangle$-ary family, then x is an α-ary variable.

The way we translate from ELF into $\mathbf{SPar}\,(\omega)^-$ is as follows. We shall define functions APP that, given an α-ary function from ELF and an α-ary abstraction sequence from $\mathbf{SPar}\,(\omega)^-_c$, returns a term or kind of $\mathbf{SPar}\,(\omega)^-_c$. In particular, APP takes base entities of ELF and returns terms and kinds of $\mathbf{SPar}\,(\omega)^-_c$; this shall be our translation that we shall prove is an inverse to the translation $(\!|\ |\!)$ defined above.

Definition 6.5 *We define:*

- *For each α-ary kind K of ELF, the large α-ary product kind $(\!| K |\!)$ of $\mathbf{SPar}\,(\omega)^-$.*
- *For each $\langle \alpha, \beta \rangle$-ary family A of ELF, and each α-ary abstraction sequence $\overrightarrow{M}^\alpha$ of $\mathbf{SPar}\,(\omega)^-$, the small β-ary product kind $\mathrm{APP}(A, \overrightarrow{M}^\alpha)$.*
- *For each α-ary object M of ELF, and each α-ary abstraction sequence $\overrightarrow{N}^\alpha$ of $\mathbf{SPar}\,(\omega)^-$, the term $\mathrm{APP}(M, \overrightarrow{N}^\alpha)$.*

simultaneously as follows:

Kinds

- $(\!| \mathbf{Type} |\!) \equiv \mathbf{Type}$
- *Let A be an $\langle \mathbf{0}, \alpha \rangle$-ary family, and K a β-ary kind. If $(\!| K |\!) \equiv (\Delta)\mathbf{Type}$, then*

$$(\!| \Pi x : A.K |\!) \equiv (x : \mathrm{APP}(A), \Delta)\mathbf{Type}\,.$$

  (Note that, by our assumptions, x is an α-ary variable.)

Families

- *If a is an α-ary family-level constant, and $\overrightarrow{M}^\alpha$ an α-ary abstraction sequence, then*

$$\mathrm{APP}(a, \overrightarrow{M}^\alpha) \equiv a[\overrightarrow{M}^\alpha]$$

- *Let A be an $\langle \mathbf{0}, \alpha \rangle$-ary family, and B an $\langle \mathbf{0}, \beta \rangle$-ary family. If $\mathrm{APP}(B) \equiv (\Delta)P$, then*

$$\mathrm{APP}(\Pi x : A.B) \equiv (x : \mathrm{APP}(A), \Delta)P\,.$$

- *Let A be an $\langle \mathbf{0}, \alpha\rangle$-ary family, and M an $\langle \beta, \gamma\rangle$-ary family. For $M^\alpha, \overrightarrow{N}^\beta$ an $(\alpha)\hat{}\beta$-ary abstraction sequence,*

$$\mathrm{APP}(\lambda x : A.M, M^\alpha, \overrightarrow{N}^\beta) \equiv \{M^\alpha/x\}\mathrm{APP}(M, \overrightarrow{N}^\beta)$$

- *Let A be an $\langle (\alpha)\hat{}\beta, \gamma\rangle$-ary family, and M an α-ary object. For $\overrightarrow{N}^\beta$ a β-ary abstraction sequence,*

$$\mathrm{APP}(AM, \overrightarrow{N}^\beta) \equiv \mathrm{APP}(A, [\vec{x}^\alpha]\mathrm{APP}(M, \vec{x}^\alpha), \overrightarrow{N}^\beta)$$

where $\vec{x}^\alpha$ is a fresh α-ary variable sequence.

Objects

- *If c is an α-ary object-level constant, then*

$$\mathrm{APP}(c, \overrightarrow{M}^\alpha) \equiv c[\overrightarrow{M}^\alpha]$$

- *If x is an α-ary variable, then*

$$\mathrm{APP}(x, \overrightarrow{M}^\alpha) \equiv x[\overrightarrow{M}^\alpha]$$

- *If A is an $\langle \mathbf{0}, \alpha\rangle$-ary family, and M a β-ary object, then*

$$\mathrm{APP}(\lambda x : A.M, N^\alpha, \overrightarrow{P}^\beta) \equiv \{N^\alpha/x\}\mathrm{APP}(M, \overrightarrow{P}^\beta)$$

- *If M is an $(\alpha)\hat{}\beta$-ary object, and N an α-ary object, then*

$$\mathrm{APP}(MN, \overrightarrow{P}^\beta) \equiv \mathrm{APP}(M, [\vec{x}^\alpha]\mathrm{APP}(N, \vec{x}^\alpha), \overrightarrow{P}^\beta)$$

where $\vec{x}^\alpha$ is a fresh α-ary variable sequence.

We can show that this translation is an inverse to $\langle\!| \; |\!\rangle$, via the translations $[\![\;]\!]$ and $X \mapsto X^-$ between $\mathbf{SPar}(\omega)^-$ and $\mathbf{SPar}(\omega)^-_c$ discussed in Section 5.4.

Lemma 6.6 *1. For any term M of $\mathbf{SPar}(\omega)^-_c$, if $\langle\!| M |\!\rangle$ is defined,*

$$\mathrm{APP}(\langle\!| M |\!\rangle) \equiv M^- .$$

2. For any α-ary abstraction $[\Delta]M$ of $\mathbf{SPar}(\omega)^-_c$, and α-ary abstraction sequence $\vec{N}$ of $\mathbf{SPar}(\omega)^-$, if $\langle\!| [\vec{x}]M |\!\rangle$ is defined,

$$\mathrm{APP}(\langle\!| [\Delta]M |\!\rangle, \vec{N}) \equiv \{\vec{N}/\vec{x}\}M^- .$$

3. For any object M typable under $\langle\!| \Gamma |\!\rangle$ and $\langle\!| \Sigma |\!\rangle$, $\left\langle\!\left| \left[\!\left[\mathrm{APP}(M, \vec{N}) \right]\!\right] \right|\!\right\rangle \equiv M \left\langle\!\left| \left[\!\left[\vec{N} \right]\!\right] \right|\!\right\rangle$.

4. For any family A typable under $\langle\!| \Gamma |\!\rangle$ and $\langle\!| \Sigma |\!\rangle$, $\left\langle\!\left| \left[\!\left[\mathrm{APP}(A, \vec{N}) \right]\!\right] \right|\!\right\rangle \equiv A \left\langle\!\left| \vec{N} \right|\!\right\rangle$.

Proof The proofs of the first two parts are simultaneous, by induction on M and $[\Delta]M$.

1. For z an α-ary variable or constant, and

$$\vec{N}^{\alpha} \equiv [\Delta^{\alpha_1}]N_1, \ldots, [\Delta^{\alpha_n}]N_n$$

an α-ary abstraction sequence, with $\vec{x}_i^{\alpha} \equiv \operatorname{dom}\Delta^{\alpha_i}$,

$$\begin{array}{rcl} \mathrm{APP}(\left\langle\!\left| z[\vec{N}^{\alpha}] \right|\!\right\rangle) & \equiv & \mathrm{APP}\left(z \left\langle\!\left| \vec{N}^{\alpha} \right|\!\right\rangle\right) \\ & \equiv & \mathrm{APP}\,(z, [\vec{x}^{\alpha_1}]\mathrm{APP}(\langle\!|[\Delta^{\alpha_1}]N_1|\!\rangle\,, \vec{x}^{\alpha_1}), \ldots, \\ & & \quad [\vec{x}^{\alpha_n}]\mathrm{APP}(\langle\!|[\Delta^{\alpha_n}]N_n|\!\rangle\,, \vec{x}^{\alpha_n})) \\ & \equiv & \mathrm{APP}\left(z, [\vec{x}^{\alpha_1}]N_1^-, \ldots, [\vec{x}^{\alpha_n}]N_n^-\right) \qquad \text{(i.h.)} \\ & \equiv & z[[\vec{x}^{\alpha_1}]N_1^-, \ldots, [\vec{x}^{\alpha_n}]N_n^-] \end{array}$$

as required.

2.

$$\begin{array}{rcl} \mathrm{APP}(\langle\!|[\Delta]M|\!\rangle\,, \vec{N}) & \equiv & \mathrm{APP}(\lambda\,\langle\!|\Delta|\!\rangle\,.\,\langle\!|M|\!\rangle\,, \vec{N}) \\ & \equiv & \{\vec{N}/\Delta\}\mathrm{APP}(\langle\!|M|\!\rangle) \\ & \equiv & \{\vec{N}/\Delta\}M^- \qquad \text{(i.h.)} \end{array}$$

The last two parts are proven by induction on M and A respectively. No case is difficult. $\Box$

We are now able to prove the translation formed by $\langle\!|\ |\!\rangle$ and APP sound.

Theorem 6.7

1. *If* $\langle\!|\Sigma|\!\rangle$ sig *is derivable in ELF, then* Σ sig *is derivable in* $\mathbf{SPar}\,(\omega)^-$.

2. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} \cdot$ *is derivable in ELF, then* $\Gamma \vdash_\Sigma$ valid *is derivable in* $\mathbf{SPar}\,(\omega)^-$.

3. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} K$ *is derivable in ELF, and* $(\!|K|\!) \equiv (\Delta)\mathbf{Type}$*, then* $\Gamma, \Delta \vdash_\Sigma$ valid *is derivable in* $\mathbf{SPar}\,(\omega)^-$.

4. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} A : K$ *is derivable in ELF, where* $(\!|K|\!) \equiv (\Delta)\mathbf{Type}$*;* $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$ *is derivable in* $\mathbf{SPar}\,(\omega)^-$*; and* $\mathrm{APP}(A, \vec{M}) \equiv (\Theta)N$*; then*

$$\Gamma, \Theta \vdash_\Sigma N : \mathbf{Type}$$

is derivable in $\mathbf{SPar}\,(\omega)^-$.

5. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} A : K$ *is derivable in ELF, where* $(\!|K|\!) \equiv (\Delta^\alpha)\mathbf{Type}$*;* $\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \Delta$ *is derivable in* $\mathbf{SPar}\,(\omega)^-$*; and*

$$\begin{array}{rcl} \mathrm{APP}(A, \vec{M}) & \equiv & (\Theta)N \\ \mathrm{APP}(A, \vec{N}) & \equiv & (\Theta')N' \end{array}$$

then

$$\begin{array}{rcl} \Gamma & \Vdash_\Sigma & \Theta = \Theta' \\ \Gamma, \Theta & \vdash & N = N' : \mathbf{Type} \end{array}$$

are derivable in $\mathbf{SPar}\,(\omega)^-$.

6. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} A = B : K$ *is derivable in ELF, where* $(\!|K|\!) \equiv (\Delta)\mathbf{Type}$; $\Gamma \Vdash_\Sigma \vec{M} :: \Delta$ *is derivable in* $\mathbf{SPar}\,(\omega)^-$*; and*

$$\begin{array}{rcl} \mathrm{APP}(A, \vec{M}) & \equiv & (\Theta)N \\ \mathrm{APP}(B, \vec{M}) & \equiv & (\Theta')N' \end{array}$$

then

$$\begin{array}{rcl} \Gamma & \Vdash_\Sigma & \Theta = \Theta' \\ \Gamma, \Theta & \vdash_\Sigma & N = N' : \mathbf{Type} \end{array}$$

are derivable in $\mathbf{SPar}\,(\omega)^-$.

7. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} M : A$ *is derivable in ELF, where* $\mathrm{APP}(A) \equiv (\Delta)P$*, and* $\Gamma \Vdash_\Sigma \vec{N} :: \Delta$ *is derivable in* $\mathbf{SPar}\,(\omega)^-$*, then*

$$\Gamma \vdash_\Sigma \mathrm{APP}(M, \vec{N}) : \{\vec{N}/\Delta\}P$$

is derivable in $\mathbf{SPar}\,(\omega)^-$.

8. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} M : A$ *is derivable in ELF, where* $\mathrm{APP}(A) \equiv (\Delta)Q$*, and* $\Gamma \Vdash_\Sigma \vec{N} = \vec{P} :: \Delta$ *is derivable in* $\mathbf{SPar}\,(\omega)^-$*, then*

$$\Gamma \vdash_\Sigma \mathrm{APP}(M, \vec{N}) = \mathrm{APP}(M, \vec{P}) : \{\vec{N}/\Delta\}Q$$

is derivable in $\mathbf{SPar}\,(\omega)^-$.

9. *If* $\langle\!|\Gamma|\!\rangle \vdash_{\langle\!|\Sigma|\!\rangle} M = N : A$ *is derivable in ELF, where* $\mathrm{APP}(A) \equiv (\Delta)Q$*, and* $\Gamma \Vdash_\Sigma \vec{P} :: \Delta$ *is derivable in* $\mathbf{SPar}\,(\omega)^-$*, then*

$$\Gamma \vdash_\Sigma \mathrm{APP}(M, \vec{P}) = \mathrm{APP}(N, \vec{P}) : \{\vec{P}/\Delta\}Q$$

is derivable in $\mathbf{SPar}\,(\omega)^-$.

Proof The proof is by induction on the derivation in ELF. No case presents too much difficulty. $\square$

Corollary 6.8 *For any judgement* J *of* $\mathbf{SPar}\,(\omega)^-_c$*, if* $\langle\!|J|\!\rangle$ *is derivable in ELF, then* J *is derivable in* $\mathbf{SPar}\,(\omega)^-_c$.

Proof This follows from the case $\alpha \equiv \mathbf{0}$ of the above theorem, as well as the correspondence between $\mathbf{SPar}\,(\omega)^-$ and $\mathbf{SPar}\,(\omega)^-_c$ (Theorem 5.11), using Lemma 6.6. $\square$

6.2 PAL

The system PAL, or Primitive Automath Language, is the weakest of the systems that make up the AUTOMATH family, as presented in [NGV94]. Again, we deal with a variant of the framework which uses a judgement form for equality; see Appendix B for the justification. We also take a more modern approach to contexts.

Grammar

$$\begin{array}{llll} \text{Expression} & E & ::= & x \mid c(E, \ldots, E) \\ \text{Type} & T & ::= & E \mid \mathbf{Type} \end{array}$$

A *signature* of PAL consists of a sequence of declarations of the form $(\Gamma, c, \mathtt{PN}, T)$ or (Γ, c, E, T). A *context* of PAL consists of a sequence of declarations of the form $x : T$. The judgement forms in PAL are:

$$\begin{array}{lll} \Sigma & & \text{sig} \\ \Gamma & \vdash_\Sigma & \mathtt{OK} \\ \Gamma & \vdash_\Sigma & E : T \\ \Gamma & \vdash_\Sigma & E = E : T \end{array}$$

Rules of Deduction

$$\frac{}{\langle\rangle \text{ sig}} \qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Sigma, (\Gamma, c, \mathtt{PN}, \mathbf{Type}) \text{ sig}} \; (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E : \mathbf{Type}}{\Sigma, (\Gamma, c, \mathtt{PN}, E) \text{ sig}} \; (c \notin \operatorname{dom} \Sigma) \qquad \frac{\Gamma \vdash_\Sigma E : T}{\Sigma, (\Gamma, c, E, T) \text{ sig}} \; (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Sigma \text{ sig}}{\vdash_\Sigma \mathtt{OK}} \qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Gamma, x : \mathbf{Type} \vdash_\Sigma \mathtt{OK}} \; (x \notin \operatorname{dom} \Gamma)$$

$$\frac{\Gamma \vdash_\Sigma E : \mathbf{Type}}{\Gamma, x : E \vdash_\Sigma \mathtt{OK}} \; (x \notin \operatorname{dom} \Gamma) \qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Gamma \vdash_\Sigma x : T} \; (x : T \in \Gamma)$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E}/\operatorname{dom} \Delta]T} \; ((\Delta, c, \mathtt{PN}, T) \in \Sigma) \qquad \frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E}/\operatorname{dom} \Delta]T} \; ((\Delta, c, E, T) \in \Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E : T}{\Gamma \vdash_\Sigma E = E : T} \qquad \frac{\Gamma \vdash_\Sigma E_1 = E_2 : T}{\Gamma \vdash_\Sigma E_2 = E_1 : T}$$

$$\frac{\Gamma \vdash_\Sigma E_1 = E_2 : T \quad \Gamma \vdash_\Sigma E_2 = E_3 : T}{\Gamma \vdash_\Sigma E_1 = E_3 : T}$$

$$\frac{\Gamma \vdash_\Sigma E_1 : E_2 \quad \Gamma \vdash_\Sigma E_2 = E_3 : \mathbf{Type}}{\Gamma \vdash_\Sigma E_1 : E_3}$$

$$\frac{\Gamma \vdash_\Sigma E_1 = E_2 : E_3 \quad \Gamma \vdash_\Sigma E_3 = E_4 : \mathbf{Type}}{\Gamma \vdash_\Sigma E_1 = E_2 : E_4}$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) = [\vec{E}/\operatorname{dom}\Delta]E : [\vec{E}/\operatorname{dom}\Delta]T} \quad ((\Delta, c, E, T) \in \Sigma)$$

PAL lies halfway between $\mathbf{LPar}\,(1)^-$ and $\mathbf{LPar}\,(1)$; it is equivalent to the theory obtained by insisting that every equation in a signature of $\mathbf{LPar}\,(1)$ is of the form

$$(\Delta)(c[\operatorname{dom}\Delta] = M : K)$$

and that no constant c takes part in two such equations. This shall be proven in Chapter 8; for now, we only prove that $\mathbf{LPar}\,(1)^-$ is a conservative subsystem.

Note that, in $\mathbf{LPar}\,(1)^-$, every constant is either base or of some first-order arity $\mathbf{n}$, and every variable is a base variable. Define, for each term M of $\mathbf{LPar}\,(1)^-$, the PAL term $[\![M]\!]$ thus:

$$\begin{array}{rcl} [\![c^{\mathbf{n}}[M_1, \ldots, M_n]]\!] & \equiv & c^{\mathbf{n}}([\![M_1]\!], \ldots, [\![M_n]\!]) \\ [\![x^{\mathbf{0}}]\!] & \equiv & x^{\mathbf{0}} \end{array}$$

We can extend this to a mapping from $\mathbf{LPar}\,(1)^-$ kinds to PAL types by setting

$$[\![\mathbf{Type}]\!] \equiv \mathbf{Type}$$

We can now define, for each $\mathbf{LPar}\,(1)^-$ context Γ, a PAL context Γ thus:

$$[\![x_1^{\mathbf{0}} : K_1, \ldots, x_n^{\mathbf{0}} : K_n]\!] \equiv x_1^{\mathbf{0}} : [\![K_1]\!], \ldots, x_n^{\mathbf{0}} : K_n$$

and, for each $\mathbf{LPar}\,(1)^-$ signature Σ, a PAL book $[\![\Sigma]\!]$ thus:

$$\begin{array}{rcl} [\![\langle\rangle]\!] & \equiv & \langle\rangle \\ [\![\Sigma, c^{\mathbf{n}} : (\Delta^{\mathbf{n}})K]\!] & \equiv & [\![\Sigma]\!], ([\![\Delta^{\mathbf{n}}]\!], c, \mathtt{PN}, [\![K]\!]) \end{array}$$

We extend this translation to judgements and derivations in the obvious manner:

$$\begin{array}{rcl} [\![\Sigma \text{ sig}]\!] & \equiv & [\![\Sigma]\!] \text{ sig} \\ [\![\Gamma \vdash_\Sigma \text{valid}]\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \mathtt{OK} \\ [\![\Gamma \vdash_\Sigma M : K]\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![M]\!] : [\![K]\!] \end{array}$$

Theorem 6.9 $[\![\,]\!]$ *is a bijection between the derivable judgements of* $\mathbf{LPar}\,(1)^-$ *and the derivable judgements of PAL which are not equality judgements, and in which no declaration of the form* (Δ, c, E, T) *appears in the book.*

Proof First note that, if no such declarations appear in Σ, then whenever $\Gamma \vdash_\Sigma E_1 = E_2 : T$ is derivable we have $E_1 \equiv E_2$. This is proven in a similar manner to Lemma 3.23.

That $[\![\,]\!]$ is injective is easy to see. A simple induction on derivations in $\mathbf{LPar}\,(1)^-$ proves that, if J is derivable in $\mathbf{LPar}\,(1)^-$, then $[\![J]\!]$ is derivable in PAL; an induction on derivations in PAL, using the result of the first paragraph, shows that if J is a derivable PAL judgement of the required form, then $J \equiv [\![J']\!]$ for some derivable J' in $\mathbf{LPar}\,(1)^-$. $\square$

6.3 AUT-68

Another of the AUTOMATH systems, as presented in [NGV94]. The following is again a variant of the framework using judgemental equality.

Grammar

Expression	E	$::=$	$x \mid c(E, \ldots, E) \mid [x : E]E \mid \langle E \rangle E$
Kind	T	$::=$	$E \mid \mathbf{Type}$

A *signature* in AUT-68 is a sequence of declarations of the form $(\Gamma, c, \mathtt{PN}, T)$ or (Γ, c, E, T). A *context* is a sequence of declarations of the form $x : T$. The judgement forms in AUT-68 are:

$$
\begin{array}{lll}
\Sigma & & \text{sig} \\
\Gamma & \vdash_\Sigma & \mathtt{OK} \\
\Gamma & \vdash_\Sigma & E : T \\
\Gamma & \vdash_\Sigma & E = E : T
\end{array}
$$

Rules of Deduction

$$\frac{}{\langle\rangle \text{ sig}} \qquad\qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Sigma, (\Gamma, c, \mathtt{PN}, \mathbf{Type}) \text{ sig}} \quad (c \notin \mathrm{dom}\,\Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E : \mathbf{Type}}{\Sigma, (\Gamma, c, \mathtt{PN}, E) \text{ sig}} \quad (c \notin \mathrm{dom}\,\Sigma) \qquad \frac{\Gamma \vdash_\Sigma E : T}{\Sigma, (\Gamma, c, E, T) \text{ sig}} \quad (c \notin \mathrm{dom}\,\Sigma)$$

$$\frac{\Sigma \text{ sig}}{\Gamma \vdash_\Sigma \mathtt{OK}} \qquad\qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Gamma, x : \mathbf{Type} \vdash_\Sigma \mathtt{OK}} \quad (x \notin \mathrm{dom}\,\Gamma)$$

$$\frac{\Gamma \vdash_\Sigma E : \mathbf{Type}}{\Gamma, x : E \vdash_\Sigma \mathtt{OK}} \quad (x \notin \mathrm{dom}\,\Gamma) \qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Gamma \vdash_\Sigma x : T} \quad (x : T \in \Gamma)$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E}/\mathrm{dom}\,\Delta]T} \quad ((\Delta, c, \mathtt{PN}, T) \in \Sigma) \qquad \frac{\Gamma, x : E_1 \vdash_\Sigma E_2 : \mathbf{Type}}{\Gamma \vdash_\Sigma [x : E_1]E_2 : \mathbf{Type}}$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E}/\mathrm{dom}\,\Delta]T} \quad ((\Delta, c, E', T) \in \Sigma) \qquad \frac{\Gamma, x : E_1 \vdash_\Sigma E_2 : E_3}{\Gamma \vdash_\Sigma [x : E_1]E_2 : [x : E_1]E_3}$$

$$\frac{\Gamma \vdash_\Sigma E_1 : [x : E_2]E_3 \quad \Gamma \vdash_\Sigma E_4 : E_2}{\Gamma \vdash_\Sigma \langle E_4\rangle E_1 : [E_4/x]E_3}$$

$$\frac{\Gamma \vdash_\Sigma E : T}{\Gamma \vdash_\Sigma E = E : T}$$

$$\frac{\Gamma \vdash_\Sigma E_1 = E_2 : T}{\Gamma \vdash_\Sigma E_2 = E_1 : T}$$

$$\frac{\Gamma \vdash_\Sigma E_1 = E_2 : T \quad \Gamma \vdash_\Sigma E_2 = E_3 : T}{\Gamma \vdash_\Sigma E_1 = E_3 : T}$$

$$\frac{\Gamma \vdash_\Sigma E_1 : E_2 \quad \Gamma \vdash_\Sigma E_2 = E_3 : \mathbf{Type}}{\Gamma \vdash_\Sigma E_1 : E_3}$$

$$\frac{\Gamma \vdash_\Sigma E_1 = E_2 : E_3 \quad \Gamma \vdash_\Sigma E_3 = E_4 : \mathbf{Type}}{\Gamma \vdash_\Sigma E_1 = E_2 : E_4}$$

$$\frac{\Gamma, x : E_1 \vdash_\Sigma E_2 : E_3 \quad \Gamma \vdash_\Sigma E_4 : E_1}{\Gamma \vdash_\Sigma \langle E_4\rangle[x : E_1]E_2 = [E_4/x]E_2 : [E_4/x]E_3}$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) = [\vec{E}/\mathrm{dom}\,\Delta]E : [\vec{E}/\mathrm{dom}\,\Delta]T} \quad ((\Delta, c, E, T) \in \Sigma)$$

We shall show here how $\mathbf{SPar}\,(\omega)_c^-$ can be embedded within AUT-68.

Define, for each term M of $\mathbf{SPar}\,(\omega)_c^-$, the expression $(\!| M |\!)$ of AUT-68; for each abstraction

M, the expression $(\!|M|\!)$; and, for each context Γ, the context $(\!|\Gamma|\!)$ as follows:

Terms

$$\begin{aligned}\left(\!\left|x^\alpha[\overrightarrow{M}^\alpha]\right|\!\right) &\equiv \langle(\!|M^{\alpha_n}|\!)\rangle\cdots\langle(\!|M^{\alpha_1}|\!)\rangle x^\alpha \\ \left(\!\left|c^\alpha[\overrightarrow{M}^\alpha]\right|\!\right) &\equiv c^\alpha(\left(\!\left|\overrightarrow{M}^\alpha\right|\!\right))\end{aligned}$$

Abstractions

$$(\!|[\Delta^\alpha]M|\!) \equiv [(\!|\Delta^\alpha|\!)]\,(\!|M|\!)$$

Contexts

$$\begin{aligned}(\!|\langle\rangle|\!) &\equiv \langle\rangle \\ (\!|\Gamma, x^\alpha : (\Delta^\alpha)M|\!) &\equiv (\!|\Gamma|\!)\,, x^\alpha : [(\!|\Delta^\alpha|\!)]\,(\!|M|\!)\end{aligned}$$

We extend this to define, for each kind K of $\mathbf{SPar}\,(\omega)_c^-$, the kind $(\!|K|\!)$ of AUT-68 by setting

$$(\!|\mathbf{Type}|\!) \equiv \mathbf{Type}$$

We now define, for each signature Σ of $\mathbf{SPar}\,(\omega)_c^-$, the signature $(\!|\Sigma|\!)$ of AUT-68 thus:

$$\begin{aligned}(\!|\langle\rangle|\!) &\equiv \langle\rangle \\ (\!|\Sigma, c^\alpha : (\Delta^\alpha)K|\!) &\equiv (\!|\Sigma|\!)\,, ((\!|\Delta^\alpha|\!)\,, c^\alpha, \mathtt{PN}, (\!|K|\!))\end{aligned}$$

Finally, we define, for each judgement J of $\mathbf{SPar}\,(\omega)_c^-$, the judgement $(\!|J|\!)$ of AUT-68 as follows:

$$\begin{aligned}(\!|\Sigma \text{ sig}|\!) &\equiv (\!|\Sigma|\!) \text{ sig} \\ (\!|\Gamma \vdash_\Sigma \text{valid}|\!) &\equiv (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \mathtt{OK} \\ (\!|\Gamma \vdash_\Sigma M : K|\!) &\equiv (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|M|\!) : (\!|K|\!) \\ (\!|\Gamma \vdash_\Sigma M = N : K|\!) &\equiv (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|M|\!) = (\!|N|\!) : (\!|K|\!)\end{aligned}$$

The proof of conservativity follows our by now familiar pattern. We can prove:

Lemma 6.10

$$\left[\left(\!\left|\overrightarrow{N}^\alpha\right|\!\right)/\vec{x}^\alpha\right](\!|M|\!) \twoheadrightarrow \left(\!\left|\{\overrightarrow{N}^\alpha/\vec{x}^\alpha\}M\right|\!\right)$$

Proof For the purposes of this proof, we shall write

$$\begin{aligned}X^* &\equiv \left[\left(\!\left|\overrightarrow{N}^\alpha\right|\!\right)/\vec{x}^\alpha\right]X \\ Y^\dagger &\equiv \{\overrightarrow{N}^\alpha/\vec{x}^\alpha\}Y\end{aligned}$$

for each expression X of AUT-68 and Y of $\mathbf{SPar}\,(\omega)_c^-$.

The proof is by double induction, firstly on the arity α, secondly on the term M. The case where M is not the application of one of the variables $\vec{x}^\alpha$ is straightforward.

If
$$M \equiv x^{\alpha_i}[\overrightarrow{P^{\alpha_i}}]$$
then
$$(\!|M|\!) \equiv \langle (\!|P^{\alpha_{ir_i}}|\!)\rangle \cdots \langle (\!|P^{\alpha_{i1}}|\!)\rangle x^{\alpha_i} .$$
So
$$\begin{array}{rcll}
(\!|M|\!)^* & \equiv & \langle (\!|P^{\alpha_{ir_i}}|\!)^*\rangle \cdots \langle (\!|P^{\alpha_{i1}}|\!)^*\rangle [(\!|\Delta^{\alpha_i}|\!)]\, (\!|M_i|\!) & \\
 & \twoheadrightarrow & \langle \left(\!\left|(P^{\alpha_{ir_i}})^\dagger\right|\!\right)\rangle \cdots \langle \left(\!\left|(P^{\alpha_{i1}})^\dagger\right|\!\right)\rangle [(\!|\Delta^{\alpha_i}|\!)]\, (\!|M_i|\!) & \text{(i.h. on } M) \\
 & \twoheadrightarrow & \left[\left(\!\left|(\overrightarrow{P}^\alpha)^\dagger\right|\!\right) / \operatorname{dom}\Delta^{\alpha_i}\right] (\!|M_i|\!) & \\
 & \twoheadrightarrow & \left(\!\left|\{(\overrightarrow{P}^\alpha)^\dagger / \operatorname{dom}\Delta^{\alpha_i}\} M_i\right|\!\right) & \text{(i.h. on } \alpha) \\
 & \equiv & \left(\!\left|M^\dagger\right|\!\right) &
\end{array}$$

□

Using this, we can deduce:

Theorem 6.11 *If J is derivable in* $\mathbf{SPar}\,(\omega)_c^-$*, then $(\!|J|\!)$ is derivable in AUT-68.*

Proof Induction on the derivation of J. We deal with one of the most difficult cases here:

$$(\text{var})\,\frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{M}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}A}\,(x^\alpha : (\Delta^\alpha)A \in \Gamma)$$

Let
$$\Delta^\alpha \equiv x^{\alpha_1} : (\Delta^{\alpha_1})A_1, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})A_n$$

We must show that
$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \left\langle \left(\!\left|\overrightarrow{M}^\alpha\right|\!\right) \right\rangle x^\alpha : \left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}A\right|\!\right)$$

By the previous lemma and Predicate Reduction, it is sufficient to show that
$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \left\langle \left(\!\left|\overrightarrow{M}^\alpha\right|\!\right) \right\rangle x^\alpha : \left[\left(\!\left|\overrightarrow{M}^\alpha\right|\!\right) / \Delta^\alpha\right] (\!|A|\!)$$

Now, we have
$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} x^\alpha : [(\!|\Delta^\alpha|\!)]\, (\!|A|\!)$$

Therefore, by the application rules, our conclusion will follow if we can show, for each member M^{α_i} of $\overrightarrow{M}^\alpha$,
$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|M^{\alpha_i}|\!) : \left[\left(\!\left|\overrightarrow{M}^\alpha\right|\!\right) / \Delta^\alpha\right] [(\!|\Delta^{\alpha_i}|\!)]\, (\!|A_i|\!)$$

i.e.
$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \left[\left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i}\right|\!\right)\right] (\!|M_i|\!) : \left[\left(\!\left|\overrightarrow{M}^\alpha\right|\!\right) / \Delta^\alpha\right] [(\!|\Delta^{\alpha_i}|\!)]\, (\!|A_i|\!)$$

Choose one particular i, and assume, as a secondary induction hypothesis, that we have proved this result for all the elements of $\overrightarrow{M}^\alpha$ before M^{α_i}. By the conversion rules, this will follow from

$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \left[\left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i}\right|\!\right)\right] (\!|M_i|\!) : \left[\left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i}\right|\!\right)\right] \left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}A_i\right|\!\right) \tag{6.1}$$

and

$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \left[\left(\!\left|\overrightarrow{M}^\alpha\right|\!\right)/\Delta^\alpha\right] [(\!|\Delta^{\alpha_i}|\!)] (\!|A_i|\!) = \left[\left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i}\right|\!\right)\right] \left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}A_i\right|\!\right) : \mathbf{Type} \tag{6.2}$$

(6.1) will follow, by the abstraction rules, from

$$(\!|\Gamma|\!), \left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i}\right|\!\right) \vdash_{(\!|\Sigma|\!)} (\!|M_i|\!) : \left(\!\left|\{\overrightarrow{M}^\alpha/\Delta^\alpha\}A_i\right|\!\right)$$

which is given to us by the primary induction hypothesis. (6.2) will follow from Subject Reduction (by the previous lemma) provided we can show

$$(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \left[\left(\!\left|\overrightarrow{M}^\alpha/\Delta^\alpha\right|\!\right)\right] [(\!|\Delta^{\alpha_i}|\!)] (\!|A_i|\!) : \mathbf{Type}$$

This will follow by Substitution and the secondary induction hypothesis from

$$(\!|\Gamma|\!), x^{\alpha_1} : ((\!|\Delta^{\alpha_1}|\!)) (\!|A_1|\!), \ldots, x^{\alpha_{i-1}} : ((\!|\Delta^{\alpha_{i-1}}|\!)) (\!|A_{i-1}|\!) \vdash_{(\!|\Sigma|\!)} [(\!|\Delta^{\alpha_i}|\!)] (\!|A_i|\!) : \mathbf{Type}$$

This will follow from the fact that

$$(\!|\Gamma|\!), (\!|\Delta^\alpha|\!) \vdash_{(\!|\Sigma|\!)} \mathtt{OK}$$

as given to us by the primary induction hypothesis. □

For the other direction, we assign an arity $\mathrm{Ar}\,(X)^\Gamma_\Sigma$ to each expression or kind X, given a context Γ and signature Σ, as follows:

- If x is an α-ary variable of $\mathbf{SPar}\,(\omega)^-_c$,

$$\mathrm{Ar}\,(x) \equiv \alpha$$

- If c is an α-ary constant of $\mathbf{SPar}\,(\omega)^-_c$,

$$\mathrm{Ar}\,(c(E_1, \ldots, E_n)) \equiv \mathrm{Ar}\,(c) \text{ with its first } n \text{ entries removed}$$

- $\mathrm{Ar}\,([x : E_1]E_2) \equiv (\mathrm{Ar}\,(E_1))\hat{}\,\mathrm{Ar}\,(E_2)$
- $\mathrm{Ar}\,(\langle E_1\rangle E_2)^\Gamma_\Sigma \equiv \mathrm{Ar}\,(E_2)$ with its first entry removed.
- $\mathrm{Ar}\,(\mathbf{Type}) \equiv \mathbf{0}$

We extend this definition to contexts by defining:

$$\begin{array}{rcl} \mathrm{Ar}\,(\langle\rangle)_\Sigma & \equiv & \mathbf{0} \\ \mathrm{Ar}\,(\Gamma, x : T)_\Sigma & \equiv & \mathrm{Ar}\,(\Gamma)_\Sigma\,\hat{}\,(\mathrm{Ar}\,(T)^\Gamma_\Sigma) \end{array}$$

We assume that, whenever a declaration $x : T$ is made in a context, that x is an $\mathrm{Ar}\,(T)$-ary variable; and that, whenever the declaration

$$(\Delta, c, E, T)$$

is made, that c is an $\mathrm{Ar}\,(\Delta)\,\hat{}\,\mathrm{Ar}\,(T)$-ary constant of $\mathbf{SPar}\,(\omega)^-_c$.

We now define, for each term E of AUT-68 with $\mathrm{Ar}\,(E) \equiv \alpha$ and each α-ary abstraction sequence $\overrightarrow{M}^\alpha$ of $\mathbf{SPar}\,(\omega)^-$, the term $\mathrm{APP}(E, \overrightarrow{M}^\alpha)$ of $\mathbf{SPar}\,(\omega)^-$ thus:

$$\begin{array}{rcl} \mathrm{APP}(x^\alpha, \overrightarrow{M}^\alpha) & \equiv & x^\alpha[\overrightarrow{M}^\alpha] \\ \mathrm{APP}(c^{\vec{\beta}\alpha}(E_1, \ldots, E_m), \overrightarrow{M}^\alpha) & \equiv & c^{\vec{\beta}\alpha}[[\vec{y}^{\beta_1}]\mathrm{APP}(E_1, \vec{y}^{\beta_1}), \ldots, [\vec{y}^{\beta_m}]\mathrm{APP}(E_m, \vec{y}^{\beta_m}), \overrightarrow{M}^\alpha] \\ \mathrm{APP}([x^{\alpha_0} : E_1]E_2, M^{\alpha_0}, \overrightarrow{M}^\alpha) & \equiv & \{M^{\alpha_0}/x^{\alpha_0}\}\mathrm{APP}(E_2, \overrightarrow{M}^\alpha) \\ \mathrm{APP}(\langle E_1\rangle E_2, \overrightarrow{M}^\alpha) & \equiv & \mathrm{APP}(E_2, [\vec{x}^{\alpha_0}]\mathrm{APP}(E_1, \vec{x}^{\alpha_0}), \overrightarrow{M}^\alpha) \end{array}$$

Recall the translation $[\![\,]\!]$ from $\mathbf{SPar}\,(\omega)^-$ to $\mathbf{SPar}\,(\omega)^-_c$ discussed in Chapter 5.

Lemma 6.12 *1. If M is a term of $\mathbf{SPar}\,(\omega)^-_c$,*

$$[\![\mathrm{APP}((\!| M |\!))]\!] \equiv M$$

2. If $M^\alpha \equiv [\Delta^\alpha]M$ is an α-ary abstraction of $\mathbf{SPar}\,(\omega)^-_c$,

$$[\![\mathrm{APP}((\!| M^\alpha |\!)\,, \mathrm{dom}\,\Delta^\alpha)]\!] \equiv M$$

Proof Simultaneous induction on M and M^α.

$$\begin{array}{rcl}
\left[\!\!\left[\mathrm{APP}\left(\left(\!\!\left| x^\alpha[\overrightarrow{M}^\alpha]\right|\!\!\right)\right)\right]\!\!\right] & \equiv & [\![\mathrm{APP}(\langle(\!| M^{\alpha_n} |\!)\rangle \cdots \langle(\!| M^{\alpha_1} |\!)\rangle x^\alpha)]\!] \\
& \equiv & [\![\mathrm{APP}(x^\alpha, [\vec{x}^{\alpha_1}]\mathrm{APP}((\!| M^{\alpha_1} |\!), \vec{x}^{\alpha_1}), \ldots, \\
& & \quad [\vec{x}^{\alpha_n}]\mathrm{APP}((\!| M^{\alpha_n} |\!), \vec{x}^{\alpha_n}))]\!] \\
& \equiv & [\![\mathrm{APP}(x^\alpha, M^{\alpha_1}, \ldots, M^{\alpha_n})]\!] \qquad \text{(i.h.)} \\
& \equiv & \left[\!\!\left[x^\alpha[\overrightarrow{M}^\alpha]\right]\!\!\right] \\
& \equiv & x^\alpha[\overrightarrow{M}^\alpha] \\
\left[\!\!\left[\mathrm{APP}\left(\left(\!\!\left| c^\alpha[\overrightarrow{M}^\alpha]\right|\!\!\right)\right)\right]\!\!\right] & \equiv & \left[\!\!\left[\mathrm{APP}(c^\alpha(\left(\!\!\left|\overrightarrow{M}^\alpha\right|\!\!\right)))\right]\!\!\right] \\
& \equiv & [\![c^\alpha[[\vec{x}^{\alpha_1}]\mathrm{APP}((\!| M^{\alpha_1} |\!), \vec{x}^{\alpha_1}), \ldots, [\vec{x}^{\alpha_n}]\mathrm{APP}((\!| M^{\alpha_n} |\!), \vec{x}^{\alpha_n})]]\!] \\
& \equiv & \left[\!\!\left[c^\alpha[\overrightarrow{M}^\alpha]\right]\!\!\right] \qquad \text{(i.h.)} \\
& \equiv & c^\alpha[\overrightarrow{M}^\alpha] \\
[\![\mathrm{APP}((\!| [\Delta^\alpha]M |\!), \mathrm{dom}\,\Delta^\alpha)]\!] & \equiv & [\![\mathrm{APP}([(\!|\Delta^\alpha|\!)]\,(\!| M |\!), \mathrm{dom}\,\Delta^\alpha)]\!] \\
& \equiv & [\![\{\mathrm{dom}\,\Delta^\alpha/\Delta^\alpha\}\mathrm{APP}((\!| M |\!))]\!] \\
& \equiv & [\![\mathrm{APP}((\!| M |\!))]\!] \\
& \equiv & M \qquad \text{(i.h.)}
\end{array}$$

□

We define the *domain* $\mathrm{DOM}\,T$ for each type T, which is an α-ary context of $\mathbf{SPar}\,(\omega)^-$, and the *codomain* $\mathrm{COD}\,T$, a kind of $\mathbf{SPar}\,(\omega)^-$, as follows:

If $x : (\Delta^\alpha)A$ is in Γ,

$$\mathrm{DOM}\,x \equiv \Delta^\alpha, \qquad \mathrm{COD}\,x \equiv A$$

If $c : (\Delta^\beta, \Theta^\alpha)K$ is in Σ, where β is of length n,

$$\begin{array}{rcl}
\mathrm{DOM}\,c(E_1, \ldots, E_n) & \equiv & \{[\vec{y}^{\beta_1}]\mathrm{APP}(E_1, \vec{y}^{\beta_1}), \ldots, [\vec{y}^{\beta_n}]\mathrm{APP}(E_n, \vec{y}^{\beta_n})/\Delta^\beta\}\Theta^\alpha \\
\mathrm{COD}\,c(E_1, \ldots, E_n) & \equiv & \{[\vec{y}^{\beta_1}]\mathrm{APP}(E_1, \vec{y}^{\beta_1}), \ldots, [\vec{y}^{\beta_n}]\mathrm{APP}(E_n, \vec{y}^{\beta_n})/\Delta^\beta\}K
\end{array}$$

If $\mathrm{DOM}\,E_1 \equiv x^\alpha : (\Delta^\alpha)A, \Gamma$, then

$$\begin{array}{rcl}
\mathrm{DOM}\langle E_2\rangle E_1 & \equiv & \{[\mathrm{dom}\,\Delta^\alpha]\mathrm{APP}(E_2, \mathrm{dom}\,\Delta^\alpha)/x^\alpha\}\Gamma \\
\mathrm{COD}\langle E_2\rangle E_1 & \equiv & \{[\mathrm{dom}\,\Delta^\alpha]\mathrm{APP}(E_2, \mathrm{dom}\,\Delta^\alpha)/x^\alpha\}\,\mathrm{COD}\,E_1 \\
\mathrm{DOM}[x : E_1]E_2 & \equiv & x : (\mathrm{DOM}\,E_1)\,\mathrm{COD}\,E_1, \mathrm{DOM}\,E_2 \\
\mathrm{COD}[x : E_1]E_2 & \equiv & \mathrm{COD}\,E_2 \\
\mathrm{DOM}\,\mathbf{Type} & \equiv & \langle\rangle \\
\mathrm{COD}\,\mathbf{Type} & \equiv & \mathbf{Type}
\end{array}$$

Lemma 6.13 *For any base kind K of $\mathbf{SPar}\,(\omega)^-_c$,*

$$[\![\mathrm{COD}\,(\!| K |\!)]\!] \equiv K$$

Proof A simple induction on K. $\square$

Theorem 6.14

1. *If* $(\!|\Sigma|\!)$ sig *in AUT-68, then* Σ sig *in* $\mathbf{SPar}\,(\omega)^-$.
2. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \texttt{OK}$ *in AUT-68, then* $\Gamma \vdash_\Sigma$ valid *in* $\mathbf{SPar}\,(\omega)^-$.
3. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} E : T$ *in AUT-68, and*
$$\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \mathrm{DOM}\,T$$
in $\mathbf{SPar}\,(\omega)^-$, *then*
$$\Gamma \vdash_\Sigma \mathrm{APP}(E, \overrightarrow{M}^\alpha) : \{\overrightarrow{M}^\alpha / \,\mathrm{DOM}\,T\}\,\mathrm{COD}\,T$$
in $\mathbf{SPar}\,(\omega)^-$.
4. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} E : T$ *in AUT-68, and*
$$\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \mathrm{DOM}\,T$$
in $\mathbf{SPar}\,(\omega)^-$, *then*
$$\Gamma \vdash_\Sigma \mathrm{APP}(E, \overrightarrow{M}^\alpha) = \mathrm{APP}(E, \overrightarrow{N}^\alpha) : \{\overrightarrow{M}^\alpha / \,\mathrm{DOM}\,T\}\,\mathrm{COD}\,T$$
5. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} E_1 = E_2 : T$ *in AUT-68, and*
$$\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \mathrm{DOM}\,T$$
in $\mathbf{SPar}\,(\omega)^-$, *then*
$$\Gamma \vdash_\Sigma \mathrm{APP}(E_1, \overrightarrow{M}^\alpha) : \{\overrightarrow{M}^\alpha / \,\mathrm{DOM}\,T\}\,\mathrm{COD}\,T$$

Proof Simultaneous, by induction on the derivation in AUT-68. Again, no case is difficult. $\square$

Theorem 6.15 *For any judgement J of* $\mathbf{SPar}\,(\omega)^-_c$, *$J$ is derivable in* $\mathbf{SPar}\,(\omega)^-_c$ *if and only if $(\!|J|\!)$ is derivable in AUT-68.*

Proof One direction is given by Theorem 6.11. For the other:

1. If $(\!|\Sigma|\!)$ sig then Σ sig by Theorem 6.14.
2. If $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)}$ valid, then $\Gamma \vdash_\Sigma$ valid by Theorem 6.14.
3. If $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|E|\!) : (\!|T|\!)$, then
$$\Gamma \vdash_\Sigma \mathrm{APP}((\!|E|\!)) : \mathrm{COD}\,(\!|T|\!)$$

by Theorem 6.14, and so

$$\Gamma \vdash_\Sigma E : T$$

by Lemmas 6.12 and 6.13.

4. If $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|E_1|\!) = (\!|E_2|\!) : (\!|T|\!)$, then

$$\Gamma \vdash_\Sigma \mathrm{APP}((\!|E_1|\!)) = \mathrm{APP}((\!|E_2|\!)) : \mathrm{COD}\,(\!|T|\!)$$

by Theorem 6.14, and so

$$\Gamma \vdash_\Sigma E_1 = E_2 : T$$

by Lemmas 6.12 and 6.13.

□

6.4 PAL$^+$

This system was presented in [Luo03]. Its design philosophy is in many ways similar to that of TF. We show here that **LPar** (ω), TF itself, is a conservative subsystem of PAL$^+$. The system PAL$^+$ is as follows.

Grammar We define classes of i-ary objects and kinds for each natural number i as follows. Following the terminology introduced for TF, we shall call the 0-ary objects and kinds *base* objects and kinds.

$$\begin{array}{llcl}
\text{Base object} & k^0 & ::= & x^0 \mid c^0 \mid \mathrm{let}\, x^i\, [x^{r_1} : K^{r_1}, \ldots, x^{r_i} : K^{r_i}] = k^0 \,\mathrm{in}\, K^0 k^0 \\
& & & \mid k^i[k^{r_1}, \ldots, k^{r_i}] \\
i\text{-ary object} & k^i & ::= & x^i \mid c^i \mid \mathrm{let}\, x^j\, [x^{r_1} : K^{r_1}, \ldots, x^{r_j} : K^{r_j}] = k^0 \,\mathrm{in}\, K^0 k^i \\
\text{Base kind} & K^0 & ::= & \mathbf{Type} \mid \mathrm{El}\left(k^0\right) \mid \mathrm{let}\, x^i\, [x^{r_1} : K^{r_1}, \ldots, x^{r_1} : K^{r_1}] = k^0 \,\mathrm{in}\, K^0 K^0 \\
i\text{-ary kind} & K^i & ::= & (x^{r_1} : K^{r_1}, \ldots, x^{r_i} : K^{r_i}) K^0 \\
& & & \mid \mathrm{let}\, x^j\, [x^{r_1} : K^{r_1}, \ldots, x^{r_j} : K^{r_j}] = k^0 \,\mathrm{in}\, K^0 K^i
\end{array}$$

A *context* in PAL$^+$ is a sequence of declarations of the form $x : K$, where x is a variable and K a kind of the same arity, or

$$x[y_1 : K_1, \ldots, y_n : K_n] = k : K$$

where x is an n-ary variable, each y_i is a variable of the same arity as the kind K_i, k is a base object, and K a base kind.

A *signature* in PAL$^+$ is a sequence of declarations of the form $c : K$, where c is a constant and K a kind of the same arity, or

$$(x_1 : K_1, \ldots, x_n : K_n) k = k' : K$$

where each x_i is a variable of the same arity as the kind K_i, k and k' are base objects, and K

is a base kind. The judgement forms in PAL$^+$ are:

$$\begin{array}{lll} \Sigma & & \text{sig} \\ \Gamma & \vdash_\Sigma & \text{valid} \\ \Gamma & \vdash_\Sigma & K \text{ kind} \\ \Gamma & \vdash_\Sigma & K = K' \\ \Gamma & \vdash_\Sigma & k : K \\ \Gamma & \vdash_\Sigma & k = k' : K \end{array}$$

In the last three judgement forms, we insist that all the entities on the right-hand side have the same arity.

We shall not reproduce all the rules of deduction of PAL$^+$ here; refer to [Luo03] for the full list.

We shall show that $\mathbf{LPar}\,(\omega)_c$ is a conservative subsystem of PAL$^+$.

We assume that the set of all n-ary variables in PAL$^+$ is the same as $\bigcup_\alpha \mathcal{V}^\alpha$, where the union is taken over all arities α of *length* n; that is, all arities of the form $(\alpha_1, \ldots, \alpha_n)$. In particular, the set of all 0-ary variables in PAL$^+$ is the same as the set of all base variables in $\mathbf{LPar}\,(\omega)_c$. We make a similar assumption regarding constants.

Definition 6.16 *We define simultaneously, given an* $\mathbf{LPar}\,(\omega)_c$ *signature* Σ *and context* Γ *such that* (Σ, Γ) *is consistent:*

- *For each* $\mathbf{LPar}\,(\omega)_c$ *term* M*, every free variable in which is in* $\mathrm{dom}\,\Gamma$ *and every constant in which is in* $\mathrm{dom}\,\Sigma$*, the* 0*-ary PAL*$^+$ *term* $(\!| M |\!)^\Gamma_\Sigma$.
- *For each* $\mathbf{LPar}\,(\omega)_c$ *kind* K*, every free variable in which is in* $\mathrm{dom}\,\Gamma$ *and every constant in which is in* $\mathrm{dom}\,\Sigma$*, the* 0*-ary PAL*$^+$ *type* $(\!| K |\!)^\Gamma_\Sigma$.
- *For each* $\mathbf{LPar}\,(\omega)_c$ *context* Δ *such that* Γ, Δ *is consistent w.r.t.* Σ*, the PAL*$^+$ *context* $(\!| \Delta |\!)^\Gamma_\Sigma$.

The definition is as follows:

Terms

- *If*
$$c^\alpha : (x^{\alpha_1} : (\Delta^{\alpha_1})K_1, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})K_n) \in \Sigma\,,$$
then
$$\left(\!\left| c^\alpha[\overrightarrow{M^\alpha}] \right|\!\right)^\Gamma_\Sigma \equiv c^\alpha[\left(\!\left| \overrightarrow{M^\alpha} \right|\!\right)^\Gamma_\Sigma]$$
where
$$(\!| [(\Delta^{\alpha_i})^*]M_i |\!)^\Gamma_\Sigma \equiv \mathrm{let}\, x^{\alpha_i} \left[(\!| (\Delta^{\alpha_i})^* |\!)^\Gamma_\Sigma \right] = (\!| M_i |\!)^{\Gamma,(\Delta^{\alpha_i})^*}_\Sigma : (\!| K_i^* |\!)^\Gamma_\Sigma \,\mathrm{in}\, x^{\alpha_i}$$
- *If*
$$x^\alpha : (x^{\alpha_1} : (\Delta^{\alpha_1})K_1, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})K_n) \in \Gamma\,,$$

then
$$\left(\!\left| x^\alpha[\overrightarrow{M}^\alpha] \right|\!\right)_\Sigma^\Gamma \equiv x^\alpha[\left(\!\left| \overrightarrow{M}^\alpha \right|\!\right)_\Sigma^\Gamma]$$
where
$$(\!|[(\Delta^{\alpha_i})^*]M_i|\!)_\Sigma^\Gamma \equiv \mathrm{let}\, x^{\alpha_i} \left[(\!|(\Delta^{\alpha_i})^*|\!)_\Sigma^\Gamma\right] = (\!|M_i|\!)_\Sigma^{\Gamma,(\Delta^{\alpha_i})^*} : (\!|K_i^*|\!)_\Sigma^\Gamma \,\mathrm{in}\, x^{\alpha_i}$$

Kinds

$$\begin{aligned} (\!|M|\!)_\Sigma^\Gamma &\equiv \mathrm{El}\left((\!|M|\!)_\Sigma^\Gamma\right) \\ (\!|\mathbf{Type}|\!)_\Sigma^\Gamma &\equiv \mathbf{Type} \end{aligned}$$

Contexts

$$\begin{aligned} (\!|\langle\rangle|\!)_\Sigma^\Gamma &\equiv \langle\rangle \\ (\!|\Delta, x^\alpha : (\Theta^\alpha)K|\!)_\Sigma^\Gamma &\equiv (\!|\Delta|\!)_\Sigma^\Gamma, x^\alpha : ((\!|\Theta^\alpha|\!)_\Sigma^{\Gamma,\Delta})\,(\!|K|\!)_\Sigma^{\Gamma,\Delta,\Theta^\alpha} \end{aligned}$$

We can now define a PAL$^+$ signature $(\!|\Sigma|\!)$ for each $\mathbf{LPar}\,(\omega)_c$ *signature Σ thus:*

$$\begin{aligned} (\!|\langle\rangle|\!) &\equiv \langle\rangle \\ (\!|\Sigma, c^\alpha : (\Delta^\alpha)K|\!) &\equiv (\!|\Sigma|\!), c^\alpha : ((\!|\Delta^\alpha|\!)_\Sigma^{\langle\rangle})\,(\!|K|\!)_\Sigma^{\Delta^\alpha} \\ (\!|\Sigma, (\Delta)(M = N : K)|\!) &\equiv (\!|\Sigma|\!), ((\!|\Delta|\!)_\Sigma^{\langle\rangle})((\!|M|\!)_\Sigma^\Delta = (\!|N|\!)_\Sigma^\Delta : (\!|K|\!)_\Sigma^\Delta) \end{aligned}$$

and a PAL$^+$ judgement $(\!|J|\!)$ for each $\mathbf{LPar}\,(\omega)_c$ *judgement J thus:*

$$\begin{aligned} (\!|\Sigma \text{ sig}|\!) &\equiv (\!|\Sigma|\!) \text{ sig} \\ (\!|\Gamma \vdash_\Sigma \text{valid}|\!) &\equiv (\!|\Gamma|\!)_\Sigma^{\langle\rangle} \vdash_{(\!|\Sigma|\!)} \text{valid} \\ (\!|\Gamma \vdash_\Sigma M : K|\!) &\equiv (\!|\Gamma|\!)_\Sigma^{\langle\rangle} \vdash_{(\!|\Sigma|\!)} (\!|M|\!)_\Sigma^\Gamma : (\!|K|\!)_\Sigma^\Gamma \\ (\!|\Gamma \vdash_\Sigma M = N : K|\!) &\equiv (\!|\Gamma|\!)_\Sigma^{\langle\rangle} \vdash_{(\!|\Sigma|\!)} (\!|M|\!)_\Sigma^\Gamma = (\!|N|\!)_\Sigma^\Gamma : (\!|K|\!)_\Sigma^\Gamma \end{aligned}$$

Lemma 6.17 *If $\Gamma \subseteq \Delta$,*
$$(\!|X|\!)_\Sigma^\Gamma \equiv (\!|X|\!)_\Sigma^\Delta$$

Proof An easy induction on X. □

Lemma 6.18 *Let Δ be an α-ary context, and $\overrightarrow{M}$ an α-ary abstraction; say*

$$\begin{aligned} \Delta &\equiv x_1 : (\Delta_1)K_1, \ldots, x_n : (\Delta_n)K_n \\ \overrightarrow{M} &\equiv [\Delta_1]M_1, \ldots, [\Delta_n]M_n \end{aligned}$$

Let
$$(\!|[(\Delta_i)^*]M_i|\!)_\Sigma^\Gamma \equiv \mathrm{let}\, x_i \left[(\!|(\Delta_i)^*|\!)_\Sigma^\Gamma\right] = (\!|M_i|\!)_\Sigma^{\Gamma,(\!|(\Delta_i)^*|\!)} : (\!|K_i^*|\!)_\Sigma^{\Gamma,(\!|(\Delta_i)^*|\!)} \,\mathrm{in}\, x_i$$
Then
$$\left[\left(\!\left|\overrightarrow{M}\right|\!\right)_\Sigma^{\Gamma_1} / \Delta\right] (\!|X|\!)_\Sigma^{\Gamma_1,\Delta,\Gamma_2} \rhd \left(\!\left|\{\overrightarrow{M}/\Delta\}X\right|\!\right)_\Sigma^{\Gamma_1,[\left(\!\left|\overrightarrow{M}\right|\!\right)_\Sigma^{\Gamma_1}/\Delta]\Gamma_2}$$

for any term, kind or context X.

Proof The proof is by double induction, firstly on the arity α, secondly on the expression X. We deal with the case

$$X \equiv x_i[\overrightarrow{P}_i]$$

here. Let us write

$$\begin{array}{rcl} Z^* & \equiv & \{\overrightarrow{M}/\Delta\}Z \\ Z^\dagger & \equiv & \left[\left(\! \left| \overrightarrow{M} \right| \! \right)_\Sigma^{\Gamma_1} / \Delta \right] Z \end{array}$$

throughout this proof.

We must show that

$$\left(\left(\! \left| x_i[\overrightarrow{P}_i] \right| \! \right)_\Sigma^{\Gamma_1,\Delta,\Gamma_2} \right)^* \rhd \left(\! \left| X^\dagger \right| \! \right)_\Sigma^{\Gamma_1,\Gamma_2^*}$$

i.e. that

$$(\!|[(\Delta_i)^*]M_i|\!)_\Sigma^{\Gamma_1} \left[\left(\left(\! \left| \overrightarrow{P}_i \right| \! \right)_\Sigma^{\Gamma_1,\Delta,\Gamma_2} \right)^* \right] \rhd \left(\! \left| \{(\overrightarrow{P}_i)^\dagger/\Delta_i\}M_i \right| \! \right)_\Sigma^{\Gamma_1,\Gamma_2^*}$$

Well,

$$\begin{array}{rcl} (\!|[(\Delta_i)^*]M_i|\!)_\Sigma^{\Gamma_1} \left[\left(\left(\! \left| \overrightarrow{P}_i \right| \! \right)_\Sigma^{\Gamma_1,\Delta,\Gamma_2} \right)^* \right] & \rhd & \left[\left(\left(\! \left| \overrightarrow{P}_i \right| \! \right)_\Sigma^{\Gamma_1,\Delta,\Gamma_2} \right)^* / \Delta_i \right] (\!|M_i|\!)_\Sigma^{\Gamma_1,(\!|(\Delta_i)^*|\!)} \\ & \rhd & \left[\left(\! \left| (\overrightarrow{P}_i)^\dagger \right| \! \right)_\Sigma^{\Gamma_1,\Gamma_2^*} / \Delta_i \right] (\!|M_i|\!)_\Sigma^{\Gamma_1,(\!|(\Delta_i)^*|\!)} \\ & & \text{(i.h. on } X) \\ & \equiv & \left[\left(\! \left| (\overrightarrow{P}_i)^\dagger \right| \! \right)_\Sigma^{\Gamma_1,\Gamma_2^*} / \Delta_i \right] (\!|M_i|\!)_\Sigma^{\Gamma_1,\Gamma_2^*,(\!|(\Delta_i)^*|\!)} \\ & & \text{(previous lemma)} \\ & \equiv & \left(\! \left| \{(\overrightarrow{P}_i)^\dagger/\Delta_i\}M_i \right| \! \right)_\Sigma^{\Gamma_1,\Gamma_2^*} \qquad \text{(i.h. on } \alpha) \end{array}$$

□

Theorem 6.19 *If J is derivable in* **LPar** $(\omega)_c$*, then $(\!|J|\!)$ is derivable in PAL$^+$.*

Proof Induction on the derivation of J. We deal with one case here:

$$(\text{var}) \frac{\Gamma \Vdash_\Sigma \overrightarrow{M} :: \Delta}{\Gamma \vdash_\Sigma x[\overrightarrow{M}] : \{\overrightarrow{M}/\Delta\}K} \; (x : (\Delta)K \in \Gamma)$$

Let

$$\begin{array}{rcl} \Delta & \equiv & x_1 : (\Delta_1)K_1, \ldots, x_n : (\Delta_n)K_n \\ \overrightarrow{M} & \equiv & [\Delta_1]M_1, \ldots, [(\Delta_n)^*]M_n \end{array}$$

and, as usual, let us write

$$(\!|[(\Delta_i)^*]M_i|\!) \equiv \mathrm{let}\, x_i\, [(\!|(\Delta_i)^*|\!)] = (\!|M_i|\!) : (\!|K_i^*|\!)\, \mathrm{in}\, x_i$$

We must show that

$$[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} x \left[\left(\!\left|\overrightarrow{M}\right|\!\right)\right] : \left(\!\left|\{\overrightarrow{M}/\Delta\}K\right|\!\right) .$$

By the previous lemma and Subject Reduction, it is sufficient to show

$$[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} x \left[\left(\!\left|\overrightarrow{M}\right|\!\right)\right] : \left[\left(\!\left|\overrightarrow{M}\right|\!\right)/\Delta\right] (\!|K|\!)$$

The induction hypothesis gives us, for $i = 1, \ldots, n$, in PAL$^+$

$$[\![\Gamma]\!] , [\![(\Delta_i)^*]\!] \vdash_{[\![\Sigma]\!]} [\![M_i]\!] : [\![K_i^*]\!]$$

It can be deduced from this that

$$[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \mathrm{let}\, x_i\, [[\![(\Delta_i)^*]\!]] = [\![M_i]\!] : [\![K_i^*]\!]\, \mathrm{in}\, x_i : ([\![(\Delta_i)^*]\!])\, [\![K_i^*]\!]$$

We also have

$$[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} x_i : (x_1 : ([\![\Delta_1]\!])\, [\![K_1]\!]) \cdots (x_n : ([\![\Delta_n]\!])\, [\![K_n]\!])\, [\![K]\!]$$

The application rules now suffice to give us our conclusion. □

To prove conservativity, we first assign arities to the entities of PAL$^+$ as follows.

Kinds

$$\begin{aligned}
\mathrm{Ar}\,(\mathbf{Type}) &\equiv \mathbf{0} \\
\mathrm{Ar}\,(\mathrm{El}\,(A)) &\equiv \mathbf{0} \\
\mathrm{Ar}\,((\Delta)T) &\equiv \mathrm{Ar}\,(\Delta) \\
\mathrm{Ar}\,(\mathrm{let}\, v\, [\Delta] = t\, \mathrm{in}\, TK) &\equiv \mathrm{Ar}\,(K)
\end{aligned}$$

Contexts

$$\mathrm{Ar}\,(x_1 : K_1, \ldots, x_n : K_n) \equiv (\mathrm{Ar}\,(K_1)\,, \ldots, \mathrm{Ar}\,(K_n))$$

Note that every n-ary kind is assigned an arity of length n.

We shall assume that:

- Whenever a declaration $x^i : K^i$ is made in a context, if $\mathrm{Ar}\left(K^i\right) \equiv \alpha$, then x^i is an α-ary variable of **LPar** (ω).
- Whenever a declaration $x^i[\Delta] = k^0 : K^0$ is made in a context, if $\mathrm{Ar}\,(\Delta) \equiv \alpha$, then x^i is an α-ary variable of **LPar** (ω).
- Whenever a declaration $c^i : K^i$ is made in a signature, if $\mathrm{Ar}\left(K^i\right) \equiv \alpha$, then c^i is an α-ary constant of **LPar** (ω).

Objects

- If x^i is an α-ary variable, $\mathrm{Ar}\left(x^i\right) \equiv \alpha$.
- If c^i is an α-ary constant, $\mathrm{Ar}\left(c^i\right) \equiv \alpha$.
- $\mathrm{Ar}\left(\mathrm{let}\, x^i\,[\Delta] = k^0 : K^0 \,\mathrm{in}\, k^i\right) \equiv \mathrm{Ar}\left(k^i\right)$
- $\mathrm{Ar}\left(k^i[k^0, \ldots, k^0]\right) \equiv \mathbf{0}$

Note that every n-ary object is assigned an arity of length n.

We now define, for every n-ary object k^n of PAL$^+$ such that

$$\mathrm{Ar}\,(k^n)^\Gamma_\Sigma \equiv \alpha\,,$$

and every α-ary abstraction sequence $\overrightarrow{M}$ in $\mathbf{LPar}\,(\omega)$, the object

$$\mathrm{APP}(k^n, \overrightarrow{M})$$

of $\mathbf{LPar}\,(\omega)$ as follows.

$$\begin{array}{rcl}
\mathrm{APP}(x^i, \overrightarrow{M}) & \equiv & x^i[\overrightarrow{M}] \\
\mathrm{APP}(c^i, \overrightarrow{M}) & \equiv & c^i[\overrightarrow{M}] \\
\mathrm{APP}(\mathrm{let}\, x^i\,[\Delta] = k^0 : K^0 \,\mathrm{in}\, k^i, \overrightarrow{M}) & \equiv & \{[\mathrm{dom}\,\Delta]\mathrm{APP}(k^0)/x^i\}\mathrm{APP}(k^i, \overrightarrow{M}) \\
\mathrm{APP}(k^i[k^0_1, \ldots, k^0_i]) & \equiv & \mathrm{APP}(k^i, [\vec{x_1}]\mathrm{APP}(k^0, \vec{x_1}), \ldots, [\vec{x_i}]\mathrm{APP}(k^0_i, \vec{x_i}))
\end{array}$$

where, in the last clause, $\vec{x_1}$, …, $\vec{x_i}$ are appropriate variable sequences, whose arities are given by $\mathrm{Ar}\left(k^i\right)$.

We further define the domain $\mathrm{DOM}\,K$, an α-ary context of TF, and the codomain $\mathrm{COD}\,K$, a kind of TF, where $\mathrm{Ar}\,(K) \equiv \alpha$ as follows:

$$\begin{array}{rcl}
\mathrm{DOM}\,\mathbf{Type} & \equiv & \langle\rangle \\
\mathrm{COD}\,\mathbf{Type} & \equiv & \mathbf{Type} \\
\mathrm{DOM}\,\mathrm{El}\,(a) & \equiv & \langle\rangle \\
\mathrm{COD}\,\mathrm{El}\,(a) & \equiv & \mathrm{APP}(a) \\
\mathrm{DOM}(x_1 : K_1, \ldots, x_n : K_n)T & \equiv & x_1 : (\mathrm{DOM}\,K_1)\,\mathrm{COD}\,K_1, \ldots, x_n : (\mathrm{DOM}\,K_n)\,\mathrm{COD}\,K_n \\
\mathrm{COD}(x_1 : K_1, \ldots, x_n : K_n)T & \equiv & \mathrm{COD}\,T \\
\mathrm{DOM}\,\mathrm{let}\,v\,[\Delta] = t : T \,\mathrm{in}\, K & \equiv & \{[\mathrm{dom}\,\Delta]\mathrm{APP}(t)/v\}\,\mathrm{DOM}\,K \\
\mathrm{COD}\,\mathrm{let}\,v\,[\Delta] = t : T \,\mathrm{in}\, K & \equiv & \{[\mathrm{dom}\,\Delta]\mathrm{APP}(t)/v\}\,\mathrm{COD}\,K
\end{array}$$

Theorem 6.20 *1. If* $(\!|\Sigma|\!)$ sig *is derivable in PAL$^+$, then* Σ sig *is derivable in* $\mathbf{LPar}\,(\omega)$.

2. If $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)}$ valid *is derivable in PAL$^+$, then* $\Gamma \vdash_\Sigma$ valid *is derivable in* $\mathbf{LPar}\,(\omega)$.

3. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} k : K$ *is derivable in PAL*$^+$*, and*

$$\Gamma \Vdash_\Sigma \vec{M} :: \mathrm{DOM}\, K$$

is derivable in **LPar** (ω)*, then*

$$\Gamma \vdash_\Sigma \mathrm{APP}(k, \vec{M}) : \{\vec{M}/\, \mathrm{DOM}\, K\}\, \mathrm{COD}\, K$$

is derivable in **LPar** (ω)*.*

4. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} k : K$ *is derivable in PAL*$^+$*, and*

$$\Gamma \Vdash_\Sigma \vec{M} = \vec{N} :: \mathrm{DOM}\, K$$

is derivable in **LPar** (ω)*, then*

$$\Gamma \vdash_\Sigma \mathrm{APP}(k, \vec{M}) = \mathrm{APP}(k, \vec{N}) : \{\vec{M}/\, \mathrm{DOM}\, K\}\, \mathrm{COD}\, K$$

is derivable in **LPar** (ω)*.*

5. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} k = k' : K$ *is derivable in PAL*$^+$*, and*

$$\Gamma \Vdash_\Sigma \vec{M} :: \mathrm{DOM}\, K$$

is derivable in **LPar** (ω)*, then*

$$\Gamma \vdash_\Sigma \mathrm{APP}(k, \vec{M}) = \mathrm{APP}(k', \vec{M}) : \{\vec{M}/\, \mathrm{DOM}\, K\}\, \mathrm{COD}\, K$$

is derivable in **LPar** (ω)*.*

6. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} K$ kind *is derivable in PAL*$^+$*, then*

$$\Gamma, \mathrm{DOM}\, K \Vdash_\Sigma \mathrm{COD}\, K \ \text{kind}$$

is derivable in **LPar** (ω)*.*

7. *If* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} K = K'$ *is derivable in PAL*$^+$*, then*

$$\Gamma \Vdash_\Sigma \mathrm{DOM}\, K = \mathrm{DOM}\, K'$$

and

$$\Gamma, \mathrm{DOM}\, K \Vdash_\Sigma \mathrm{COD}\, K = \mathrm{COD}\, K'$$

are derivable in **LPar** (ω)*.*

Proof The proof is by induction on the derivation of the PAL$^+$ judgement. Again, all cases are straightforward. □

Corollary 6.21 *For any judgement J of* $\mathbf{LPar}\,(\omega)_c$*, if* $(\!|J|\!)$ *is derivable in PAL*$^+$*, then J is derivable in* $\mathbf{LPar}\,(\omega)_c$*.*

Proof This follows from the case $\alpha \equiv \mathbf{0}$ of the previous theorem, as well as the correspondence between $\mathbf{LPar}\,(\omega)$ and $\mathbf{LPar}\,(\omega)_c$ given in Section 5.4, together with the observations that

$$\begin{array}{rcl} \mathrm{APP}((\!|M|\!)) & \equiv & M \\ \mathrm{COD}((\!|K|\!)) & \equiv & K \end{array}$$

Each of these is easily proven by induction on M or K. $\square$

This proof used the fact that APP is a left inverse to $(\!|\ |\!)$. We can also show that APP is a right inverse:

Lemma 6.22 *Suppose* $\Gamma \vdash_\Sigma M : K$ *in PAL*$^+$*. Then*

$$\Gamma \vdash_\Sigma (\!|[\mathrm{DOM}\,K]\mathrm{APP}(M, \mathrm{DOM}\,K)|\!) = M : K$$

Proof An induction on M. We omit the details. $\square$

6.5 Adequacy Theorems

One of the most striking consequences of this work is that it enables us to prove adequacy theorems for a given object theory with respect to a number of logical frameworks at once. Specifically:

- The adequacy theorem for propositional logic in Chapter 4 entails that the translations of the signature Σ_{prop} given there yields an adequate representation of propositional logic in ELF, PAL, AUT-68 and PAL$^+$.
- Likewise, the adequacy theorem for predicate logic gives an adequate representation of predicate logic in ELF, AUT-68 and PAL$^+$.
- The adequacy theorem for Schroeder-Heister's extension of natural deduction gives an adequate representation of this system in ELF, AUT-68 and PAL$^+$.
- The adequacy theorem for Martin-Löf Type Theory gives an adequate representation of this system in PAL$^+$.
- The adequacy theorem for the Calculus of Constructions gives an adequate representation of this system in PAL$^+$.

We shall only prove one of these statements here; the proofs for the others are similar. We shall prove the adequacy theorem for the Calculus of Constructions in PAL$^+$.

Let

$$\Sigma'_{CoC} \equiv (\!|\Sigma_{CoC}|\!)$$

where Σ_{CoC} was given in Chapter 4, and $(\!|\ |\!)$ is the translation from TF into PAL$^+$ given above. Let $[\![\ ]\!]$ be the translation from the Calculus of Constructions into TF given in Chapter 4. Then:

Theorem 6.23

1. Σ'_{CoC} sig *in PAL$^+$.*
2. $\Gamma \vdash$ valid *in CoC iff* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}}$ valid *in PAL$^+$.*
3. *If* $\Gamma \vdash B$ type *in CoC, then* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} (\![\![B]\!]\!) : \mathbf{Type}$ *in PAL$^+$. Conversely, if* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} M : \mathbf{Type}$ *in PAL$^+$, there exists a type B, unique up to convertibility in CoC, such that* $\Gamma \vdash B$ type *in CoC and* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} M = (\![\![B]\!]\!) : \mathbf{Type}$ *in PAL$^+$.*
4. $\Gamma \vdash B = B'$ *in CoC iff* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} (\![\![B]\!]\!) = (\![\![B']\!]\!) : \mathbf{Type}$ *in PAL$^+$.*
5. *If* $\Gamma \vdash M : B$ *in CoC, then* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} (\![\![M]\!]\!) : (\![\![B]\!]\!)$ *in PAL$^+$. Conversely, if* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} N : (\![\![B]\!]\!)$ *in PAL$^+$, there exists a term M, unique up to convertibility in CoC, such that* $\Gamma \vdash M : B$ *in CoC and* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} N = (\![\![M]\!]\!) : (\![\![B]\!]\!)$ *in PAL$^+$.*
6. $\Gamma \vdash M = N : B$ *in CoC iff* $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} (\![\![M]\!]\!) = (\![\![N]\!]\!) : (\![\![B]\!]\!)$ *in PAL$^+$.*

Proof 1, 2, 4 and 6 are immediate applications of Theorems 6.19 and 6.20 to Theorem 4.15. 3 and 5 are proven similarly; we shall prove 4 here.

Suppose $\Gamma \vdash M : B$ in CoC. By Theorem 4.15,

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![M]\!] : [\![B]\!]$$

in TF. By Theorem 6.19,

$$(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} (\![\![M]\!]\!) : (\![\![B]\!]\!)$$

Conversely, suppose $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} N : (\![\![B]\!]\!)$. By Theorem 6.20,

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} \mathrm{APP}(N) : [\![B]\!]$$

in TF. Therefore, by Theorem 4.15, there is a term M of CoC such that

$$\Gamma \vdash M : B$$

in CoC, and $\mathrm{APP}(N) \equiv [\![M]\!]$. Hence, by Lemma 6.22,

$$(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} N = (\![\![M]\!]\!) : (\![\![B]\!]\!)$$

as $(\![\![M]\!]\!) \equiv (\!\mathrm{APP}(N)\!)$.

If, in addition, $\Gamma \vdash M' : B$ and $(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} N = (\![\![M']\!]\!) : (\![\![B]\!]\!)$, then

$$(\![\![\Gamma]\!]\!) \vdash_{\Sigma'_{CoC}} (\![\![M]\!]\!) = (\![\![M']\!]\!) : (\![\![B]\!]\!)$$

By Theorem 6.20,

$$[\![\Gamma]\!] \vdash_{\Sigma_{CoC}} [\![M]\!] = [\![M']\!] : [\![B]\!]$$

in TF. Hence, by Theorem 4.15,

$$\Gamma \vdash M = M' : B$$

in CoC. □

Part II

Frameworks Built from Features

Chapter 7

Features

In the first part of this thesis, we introduced an infinite family of logical frameworks, and showed how they extend one another conservatively. We further showed how several existing logical frameworks could be fitted into this hierarchy.

Notice also that every inclusion that we have proved can be characterized as the larger system being the result of adding one or more 'features' to the smaller. For each of our weak frameworks S, S is the result of adding the ability to declare equalities to S^{-}. $\mathbf{LPar}\,(n)$ is the result of adding the ability to declare nth-order constants with parameters of large type to $\mathbf{SPar}\,(n)$; likewise $\mathbf{LPar}\,(n)^{-}$ and $\mathbf{SPar}\,(n)^{-}$. LF is the result of adding to $\mathbf{LPar}\,(\omega)$ the ability to speak of functions as first-class objects; and so forth.

This suggests the following view of logical frameworks. We may consider a framework to be a set of *features*. We may hope that we can define a whole swathe of frameworks at once as follows: we give a set of features, defined independently of one another, then we allow any combination of them to comprise a framework. (We anticipate that we may have to restrict which combinations are allowed, by some *dependency* relation, say.) We would hope that the framework defined by a set of features is a conservative extension of the framework defined by any subset; that is, that adding a feature never interferes with the behaviour of those features already present. If, in the future, we then wished to extend one of these frameworks, with local definitions or coercive subtyping, say, we would be able to define the appropriate feature; it would then be clear what it meant to extend any of these frameworks in the same manner, and the resulting frameworks would form a neat hierarchy of the form we see in Figure 1.1, just as they did before.

In this second part, we reconstruct the results of the first part within just such a system. In this chapter, we shall show how the frameworks introduced in Chapters 3 and 5 can be built out of a set of features. In chapter 8, we shall show once again how several existing frameworks extend our weak frameworks, this time by showing in each case how a weak framework can be extended, by the addition of features, to form a system equivalent to the existing framework. We shall conclude by discussing several other features with which our frameworks could be extended.

In order to carry out this scheme, we shall need some general definition of a logical framework. We shall adopt a semi-formal approach to this task. From the features we give, it shall be straightforward to write down a formal definition of the logical framework defined by a given set of features. Each of our theorems shall thus be precisely stated, for each given framework. However, we shall not attempt a definition of the class of 'logical frameworks' or 'features'. Thus, each of our theorems that begins 'for any logical framework F in the hierarchy' should be seen as being a theorem *schema*, giving a set of theorems, one for each set of features. The proofs should likewise be read as proof schemas, showing how the proof of each theorem proceeds, which shall be along similar lines for each theorem in the schema.

Grammar We shall consider a *grammar* to consist of a number of *syntactic classes*; a number of sets of *variables*; a number of sets of *constants*; a set of *(expression) constructors*; each of which takes a number of expressions from other syntactic classes, possibly binding some variables in each, and forms an expression of some syntactic class; and a set of *inclusions*, specifying that one syntactic class is to be a subclass of another. Each inclusion may also specify some *identifications* to hold; that is, if C_1 is declared to be a subclass of C_2, we may identify the expressions of C_1 produced by some constructor with those of C_2 produced by some other constructor. These grammars will be presented in a BNF style.

As an example, the grammar of the simply-typed lambda calculus might be presented as follows:

$$\begin{array}{lccl} \text{Type} & \tau & ::= & \alpha \mid \tau \to \tau \\ \text{Term} & t & ::= & c \mid x \mid tt \mid \lambda x t \end{array}$$

This shorthand is intended to indicate the following:

- There are two syntactic classes, the class of *types* and the class of *terms*.
- There are two sets of variables, one of *type variables* and one of *term variables*.
- There is one set of constants, that of *term constants*.
- There are five expression constructors:
  - An (unnamed) constructor which takes a type variable and returns a type.
  - $\to$, which takes two types and returns a type.
  - An (unnamed) constructor which takes a term variable and returns a term.
  - Application, which takes two terms and returns a term.
  - λ, which takes a term, binds a term variable, and returns a term.
- We shall henceforth use α to denote an arbitrary type variable, x for an arbitrary term variable, τ for a type and t for a term.

Notice that not all this information is explicit in the presentation we use (α could well have stood for a constant, for example). Our BNF-style grammars will therefore usually have to be accompanied by some commentary.

An inclusion shall be indicated as in the following example, giving the grammar of an impredicative dependent type theory with a single universe U à la Russell:

$$\begin{array}{llll} \text{Type} & T & ::= & U \mid M \mid \Pi x : T.T \\ \text{Term} & M & ::= & x \mid \lambda x : T.M \mid MM \mid \pi x : T.M \end{array}$$

With this grammar, we would make the side comment that we identify the two Π-types produced from T and M — that is, the one formed by considering M as a type and applying Π to get $\Pi x : T.M$, the other formed by constructing $\pi x : T.M$ then considering this term as a type.

The *grammar of a logical framework* consists of a grammar in which three of the syntactic classes are distinguished as:

- The set of *signature declarations.*
- The set of *context declarations.*
- The set of *judgement bodies.*

Each context declaration *declares* a variable; we say that it is a declaration *of* that variable. Likewise, each signature declaration may declare a constant, or it may declare none. We shall write "of x", "of c" or "of none" after the appropriate constructor in the grammar to indicate which of these cases hold. For example, in a framework that allowed constant declarations and equations in its signatures, we may write

$$\text{Signature declaration } \gamma ::= x : K \text{ of } x \mid (\Delta)(M = M : K) \text{ of none} .$$

A *signature* is a sequence of signature declarations, no two of which are of the same constant. It is *consistent* iff it has no free variables, and every constant that occurs has been declared earlier in the signature. Its *domain* is the sequence of constants declared by the signature, in the order in which they are declared.

A *context* is a sequence of context declarations, no two of which are of the same variable. It is *consistent* with respect to a consistent signature Σ iff every constant that occurs is declared in Σ, and every variable that occurs free has been declared earlier in the context. Its *domain* is the sequence of variables declared by the context, in the order in which they are declared.

A *judgement* is an expression of one of the two forms:

$$\Sigma \text{ sig}$$

or

$$\Gamma \vdash_\Sigma J$$

where Σ is a signature, Γ a context, and J a judgement body. A *logical framework* is given by an LF-grammar, together with a set of *rules of deduction.* A rule of deduction is a set of $n + 1$-tuples of judgements, written

$$\frac{J_1 \quad \cdots \quad J_n}{J_{n+1}}$$

Each member of a rule of deduction is called an *instance* of that rule. $J_1, \ldots, J_n$ are called the *premises* of that instance, and J_{n+1} the *conclusion*.

The set of *derivable* judgements of the framework is defined inductively as follows:

- If
$$\frac{J_1 \quad \cdots \quad J_n}{J}$$
is an instance of a rule of deduction, and $J_1, \ldots, J_n$ are derivable, then J is derivable.

The base case is, of course, the case $n = 0$, when the rule of deduction is called an *axiom*. In practice, the only axiom in any of our examples is

$$\frac{}{\langle\rangle \text{ sig}}$$

We shall also include in the formal definition of a framework a number of *defined operations* on the expressions of our syntactic classes; these shall include instantiation, substitution, and any defined judgement forms.

A Hierarchy Defined by Features We wish a *feature* to be a way of extending a logical framework. Given the grammatical definitions above, we shall take a feature to consist of:

- A set of *new syntactic classes*, *new variables sets* and *new constant sets* to be added to the framework.
- A set of *new expression constructors*, that may take arguments from either the new syntactic classes or the old ones (i.e. the ones already existing in the framework), and return an expression either in a new class or an old one.
- A set of *new rules of deduction*, which may refer to both the old and the new expressions.
- A set of *new defined operations*.
- A set of *clauses* that are to be added to the definitions of old defined operations to state how they behave when applied to the expressions formed by the new constructors.
- A set of *redundancies*. A redundancy consists of a declaration that an old expression constructor is to be removed from the grammar; we specify an expression that is to replace it wherever it occurred in a rule of deduction or defined operation. (For example, if we are adding first-class untyped lambda-abstractions to one of our weak frameworks, we might declare that the constructor for terms of the form
$$c^{\mathbf{1}}[[x^{\mathbf{0}}]M]$$
is to be removed, each term of this form being replaced with
$$c^{\mathbf{1}} \lambda x^{\mathbf{0}}.M\ .)$$

We shall say that a feature *depends on* those old syntactic classes, expression constructors and defined operations that its new components refer to. A feature can only be added to a framework that contains all those components on which the feature depends.

We assume that our rules of deduction and defined operations extend themselves automatically whenever a feature is added to the framework. For example, if the following rule of deduction is in the framework:

$$\frac{\Gamma \vdash_\Sigma K \text{ kind}}{\Gamma, x : K \vdash_\Sigma \text{valid}} \; (x \notin \operatorname{dom}\Gamma)$$

then, when a feature is added to the framework, the rule of deduction may be applied with K either an old or a new expression of the class of kinds.

Similarly, we assume that, for each defined operation, the clauses in the old definiton are still valid over the new, wider set of expressions, and new clauses are added for each new expression constructor; if the feature does not state explicitly what these clauses are to be, they shall indicate that the operation is passed directly to the subterms. For example, if we add lambda-abstractions to a weak framework of which instantiation is a defined operation, we assume that the old defining clause

$$\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}y^{\mathbf{1}}[[z^{\mathbf{0}}]N] \equiv \{[z^{\mathbf{0}}]\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}N/x^{\mathbf{0}}\}M$$

still holds, that is

$$\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}(y^{\mathbf{1}}\lambda z^{\mathbf{0}}.N) \equiv \{[z^{\mathbf{0}}]\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}N/x^{\mathbf{0}}\}M \tag{7.1}$$

for all terms M and N of the new framework; and that we add clauses to handle lambda-abstraction and application:

$$\begin{array}{rcl} \{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}\lambda z^{\mathbf{0}}.N & \equiv & \lambda z^{\mathbf{0}}.\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}N \\ \{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}(NP) & \equiv & (\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}N)(\{[x^{\mathbf{0}}]M/y^{\mathbf{1}}\}P) \end{array}$$

provided NP is not of the form given in (7.1) above.

(It is this extension of the rules of deduction and defined operations that is difficult to define correctly in a formal account. Presumably, it would involve making the rules and definitions involve some form of meta-expression, containing meta-variables for expressions of each syntactic class. Producing such an account is beyond the scope of this thesis.)

So much for features taken individually; here is how we can create a hierarchy of frameworks from a set of features. We begin by presenting a system that we shall call the *basic framework*, **BF**. (This is the same as the system $\mathbf{SPar}\,(0)^-$ with which we are familiar.) We then give a set of features, together with a well-founded *dependency relation*, $\prec$. This relation must be such that, if F depends on a component (syntactic class, constructor or defined operation), then either that component is in **BF**, or it is in some feature F' such that

$$F' \prec^* F$$

where $\prec^*$ is the transitive closure of $\prec$.

A set S of features is *$\prec$-closed* iff the following condition holds:

For all $F \in S$, if $F' \prec F$ then $F' \in S$.

The framework *defined* by a $\prec$-closed set of features S is just the result of adding all these features to **BF**, in order of dependence. These frameworks shall then comprise our modular hierarchy.

In this chapter, we shall give features

$$\begin{array}{rl}
\mathbf{SDec}\,(1)\,,\mathbf{SDec}\,(2)\,,\ldots & \text{allowing definition of constants with parameters of small type} \\
\mathbf{LDec}\,(1)\,,\mathbf{LDec}\,(2)\,,\ldots & \text{allowing definition of constants with parameters of large type} \\
\mathbf{SEq}\,(m,n)\,,\mathbf{LEq}\,(m,n) & \text{allowing declaration of equations}
\end{array}$$

We shall see that the frameworks we gave in the first part can be rebuilt as follows:

$$\begin{array}{rcl}
\mathbf{SPar}\,(n)^- & = & \mathbf{BF} + \mathbf{SDec}\,(1) + \mathbf{SDec}\,(2) + \cdots + \mathbf{SDec}\,(n) \\
\mathbf{SPar}\,(\omega) & = & \mathbf{BF} + \mathbf{SDec}\,(1) + \mathbf{SDec}\,(2) + \cdots \\
\mathbf{SPar}\,(n) & = & \mathbf{BF} + \mathbf{SEq}\,(0,0) + \mathbf{SDec}\,(1) + \mathbf{SEq}\,(0,1) + \cdots + \mathbf{SDec}\,(n) + \mathbf{SEq}\,(0,n) \\
\mathbf{SPar}\,(\omega) & = & \mathbf{BF} + \mathbf{SEq}\,(0,0) + \mathbf{SDec}\,(1) + \mathbf{SEq}\,(0,1) + \cdots \\
\mathbf{LPar}\,(n)^- & = & \mathbf{BF} + \mathbf{LDec}\,(0) + \mathbf{SDec}\,(1) + \mathbf{LDec}\,(1) + \cdots + \mathbf{SDec}\,(n) + \mathbf{LDec}\,(n) \\
\mathbf{LPar}\,(\omega)^- & = & \mathbf{BF} + \mathbf{LDec}\,(0) + \mathbf{SDec}\,(1) + \mathbf{LDec}\,(1) + \cdots \\
\mathbf{LPar}\,(n) & = & \mathbf{BF} + \mathbf{LDec}\,(0) + \mathbf{SEq}\,(0,0) + \mathbf{LEq}\,(0,0) \\
 & & +\ \mathbf{SDec}\,(1) + \mathbf{LDec}\,(1) + \mathbf{SEq}\,(0,1) + \mathbf{LEq}\,(0,1) \\
 & & +\cdots \\
 & & +\ \mathbf{SDec}\,(n) + \mathbf{LDec}\,(n) + \mathbf{SEq}\,(0,n) + \mathbf{LEq}\,(0,n) \\
\mathbf{LPar}\,(\omega) & = & \mathbf{BF} + \mathbf{LDec}\,(0) + \mathbf{SEq}\,(0,0) + \mathbf{LEq}\,(0,0) \\
 & & +\ \mathbf{SDec}\,(1) + \mathbf{LDec}\,(1) + \mathbf{SEq}\,(0,1) + \mathbf{LEq}\,(0,1) \\
 & & +\cdots
\end{array}$$

7.1 The Basic Framework

BF is the following framework. This is essentially the same system as $\mathbf{SPar}\,(0)^-$ as defined in Part I. We make the following inessential changes:

- We shall distinguish formally between types and kinds. These shall be two syntactic classes, one a subclass of the other. This shall make it easier to extend systems in a uniform manner when we define features for lambda-abstractions, local definitions, and so forth.
- We shall introduce judgement forms for kinds being well-formed, and for two kinds being equal. Again, this is for convenience when extending the system.

- We allow the declaration of base variables. This is simply for aesthetic reasons, to avoid the framework being too trivial; no harm would result if, as in **SPar** $(0)^-$, we left them out.

The definition is as follows:

Grammar We have one set of variables, the *base variables*, denoted $x^{\mathbf{0}}$, $y^{\mathbf{0}}$, ..., and one set of constants, the *base constants*, denoted $c^{\mathbf{0}}$, $a^{\mathbf{0}}$, $b^{\mathbf{0}}$, ... We shall use $z^{\mathbf{0}}$, possibly decorated, to denote either a base constant or a base variable.

Base Term	$M^{\mathbf{0}}, N^{\mathbf{0}}, P^{\mathbf{0}}$	$::=$	$x^{\mathbf{0}} \mid c^{\mathbf{0}}$
Base Type	$A^{\mathbf{0}}, B^{\mathbf{0}}$	$::=$	$M^{\mathbf{0}}$
Base Kind	$K^{\mathbf{0}}$	$::=$	$A^{\mathbf{0}} \mid \mathbf{Type}$
Signature Declaration	γ	$::=$	$c^{\mathbf{0}} : K^{\mathbf{0}}$ of $c^{\mathbf{0}}$
Context Declaration	δ	$::=$	$x^{\mathbf{0}} : A^{\mathbf{0}}$ of x
Judgement Body	J	$::=$	valid $\mid K^{\mathbf{0}}$ kind $\mid K^{\mathbf{0}} = K^{\mathbf{0}} \mid M^{\mathbf{0}} : K^{\mathbf{0}} \mid$ $M^{\mathbf{0}} = M^{\mathbf{0}} : K^{\mathbf{0}}$

Rules of Deduction

$$(\text{emp\_sig})\ \frac{}{\langle\rangle\ \text{sig}} \qquad (\text{sig}_{\mathbf{0}})\ \frac{\vdash_\Sigma K^{\mathbf{0}}\ \text{kind}}{\Sigma, c^{\mathbf{0}} : K^{\mathbf{0}}\ \text{sig}}\ (c^{\mathbf{0}} \notin \text{dom}\,\Sigma)$$

$$(\text{emp\_ctxt})\ \frac{\Sigma\ \text{sig}}{\vdash_\Sigma \text{valid}} \qquad (\text{ctxt}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma A^{\mathbf{0}}\ \text{kind}}{\Gamma, x^{\mathbf{0}} : A^{\mathbf{0}} \vdash_\Sigma \text{valid}}\ (x^{\mathbf{0}} \notin \text{dom}\,\Gamma)$$

$$(\text{const}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma c^{\mathbf{0}} : K^{\mathbf{0}}}\ (c^{\mathbf{0}} : K^{\mathbf{0}} \in \Sigma) \quad (\text{var}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma x^{\mathbf{0}} : A^{\mathbf{0}}}\ (x^{\mathbf{0}} : A^{\mathbf{0}} \in \Gamma)$$

$$(\text{ref}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} : K^{\mathbf{0}}}{\Gamma \vdash_\Sigma M^{\mathbf{0}} = M^{\mathbf{0}} : K^{\mathbf{0}}} \qquad (\text{sym}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} = N^{\mathbf{0}} : K^{\mathbf{0}}}{\Gamma \vdash_\Sigma N^{\mathbf{0}} = M^{\mathbf{0}} : K^{\mathbf{0}}}$$

$$(\text{trans}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} = N^{\mathbf{0}} : K^{\mathbf{0}} \quad \Gamma \vdash_\Sigma N^{\mathbf{0}} = P^{\mathbf{0}} : K^{\mathbf{0}}}{\Gamma \vdash_\Sigma M^{\mathbf{0}} = P^{\mathbf{0}} : K^{\mathbf{0}}}$$

$$(\text{Kref}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma K^{\mathbf{0}}\ \text{kind}}{\Gamma \vdash_\Sigma K^{\mathbf{0}} = K^{\mathbf{0}}} \qquad (\text{Ksym}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma K_1^{\mathbf{0}} = K_2^{\mathbf{0}}}{\Gamma \vdash_\Sigma K_2^{\mathbf{0}} = K_1^{\mathbf{0}}}$$

$$(\text{Ktrans}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma K_1^{\mathbf{0}} = K_2^{\mathbf{0}} \quad \Gamma \vdash_\Sigma K_2^{\mathbf{0}} = K_3^{\mathbf{0}}}{\Gamma \vdash_\Sigma K_1^{\mathbf{0}} = K_3^{\mathbf{0}}}$$

$$(\text{conv}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} : K_1^{\mathbf{0}} \quad \Gamma \vdash_\Sigma K_1^{\mathbf{0}} = K_2^{\mathbf{0}}}{\Gamma \vdash_\Sigma M^{\mathbf{0}} : K_2^{\mathbf{0}}} \qquad (\text{conv\_eq}_{\mathbf{0}})\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} = N^{\mathbf{0}} : K_1^{\mathbf{0}} \quad \Gamma \vdash_\Sigma K_1^{\mathbf{0}} = K_2^{\mathbf{0}}}{\Gamma \vdash_\Sigma M^{\mathbf{0}} = N^{\mathbf{0}} : K_2^{\mathbf{0}}}$$

$$(\mathbf{Type})\ \frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma \mathbf{Type}\ \text{kind}}$$

$$\text{(lift)}\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} : \mathbf{Type}}{\Gamma \vdash_\Sigma M^{\mathbf{0}}\ \text{kind}} \qquad \text{(lift\_eq)}\ \frac{\Gamma \vdash_\Sigma M^{\mathbf{0}} = N^{\mathbf{0}} : \mathbf{Type}}{\Gamma \vdash_\Sigma M^{\mathbf{0}} = N^{\mathbf{0}}}$$

7.2 Parametrization

7.2.1 Parameters of Small Kind

We introduce the family of features **SDec** (n) for $n \geqslant 1$ which extend one another conservatively. We shall write **SDec** (ω) for the union of all these features.

SDec (n) depends on **SDec** $(n-1)$ if $n > 1$.

Grammar We introduce sets of *α-ary constants* for each nth-order arity α, and *α-ary variables* for each $n-1$st-order arity α. We shall write c^α, a^α, ... for an arbitrary α-ary constant, x^α, y^α, ... for an arbitrary α-ary variable, and z^α for either an α-ary constant or an α-ary variable.

We define *α-ary variable sequences*, *α-ary contexts*, *α-ary abstractions*, *α-ary abstraction sequences* and *instantiation* for every nth-order arity α exactly as we did for the system **SPar** (n) in Part I. We also define the set of *α-ary small parametric contexts*; these shall be the same as those expressions we called simply 'α-ary contexts' in Chapter 5. We introduce the new name to distinguish between these expressions and the contexts that may appear on the left-hand side of a judgement; a future extension to the system may mean that these two classes of expression no longer coincide. Likewise, we shall define the three judgement forms

$$\begin{array}{lll} \Gamma & \Vdash_\Sigma & \Delta_1^\alpha = \Delta_2^\alpha \\ \Gamma & \Vdash_\Sigma & \overrightarrow{M}^\alpha :: \Delta^\alpha \\ \Gamma & \Vdash_\Sigma & \overrightarrow{M}^\alpha = \overrightarrow{N}^\alpha :: \Delta^\alpha \end{array}$$

as we did in Chapter 5.

For every appropriate arity $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, we add the following constructors to the grammar for every appropriate arity $\alpha \equiv (\alpha_1, \ldots, \alpha_m)$:

$$\begin{array}{llll} \text{Base Term} & M^{\mathbf{0}} & ::= & \cdots \mid c^\alpha[[\vec{x}^{\alpha_1}]M^{\mathbf{0}}, \ldots, [\vec{x}^{\alpha_m}]M^{\mathbf{0}}] \\ & & & \mid x^\alpha[[\vec{x}^{\alpha_1}]M^{\mathbf{0}}, \ldots, [\vec{x}^{\alpha_m}]M^{\mathbf{0}}] \\ \text{Signature Declaration} & \gamma & ::= & \cdots \mid c^\alpha : (\Delta^\alpha)K^{\mathbf{0}} \\ \text{Context Declaration} & \delta & ::= & \cdots \mid x^\alpha : (\Delta^\alpha)A^{\mathbf{0}} \end{array}$$

Defined Operations We make nth-order instantiation a defined operation that is formally added to the system when **SDec** (n) is added; thus, it is automatically extended to be a congruence with respect to any constructor that may be introduced in the future. Note that it is defined in terms of $n-1$st-order instantiation, which must already be present thanks to the feature **SDec** $(n-1)$ (if $n > 1$).

We shall also define the *argument list* of every variable declared in the context and every constant declared in the signature. The argument list of an α-ary variable or constant shall always be an α-ary small pure context. If

$$x^{\alpha} : (\Delta^{\alpha})A^{\mathbf{0}}$$

is in the context, then Δ^{α} is the argument list of x^{α}. If

$$c^{\alpha} : (\Delta^{\alpha})K^{\mathbf{0}}$$

is in the signature, then Δ^{α} is the argument list of c^{α}. This operation shall be useful when adding typed lambda-abstractions; it shall give us the types that are to be

Rules of Deduction For each arity α of the appropriate order, we add the following rules of deduction:

$$(\mathrm{sig}_{\alpha})\ \frac{\Delta^{\alpha} \vdash_{\Sigma} K^{\mathbf{0}} \text{ kind}}{\Sigma, c^{\alpha} : (\Delta^{\alpha})K^{\mathbf{0}} \text{ sig}}\ (c^{\alpha} \notin \operatorname{dom}\Sigma) \qquad (\mathrm{ctxt}_{\alpha})\ \frac{\Gamma, \Delta^{\alpha} \vdash_{\Sigma} A^{\mathbf{0}} \text{ kind}}{\Gamma, x^{\alpha} : (\Delta^{\alpha})A^{\mathbf{0}} \vdash_{\Sigma} \text{valid}}\ (x^{\alpha} \notin \operatorname{dom}\Sigma)$$

$$(\mathrm{const}_{\alpha})\ \frac{\Gamma \Vdash_{\Sigma} \overrightarrow{\hat{M}^{\alpha}} :: \Delta^{\alpha}}{\Gamma \vdash_{\Sigma} c^{\alpha}[\overrightarrow{\hat{M}^{\alpha}}] : \{\overrightarrow{\hat{M}^{\alpha}}/\Delta^{\alpha}\}K^{\mathbf{0}}}\ (c^{\alpha} : (\Delta^{\alpha})K^{\mathbf{0}} \in \Sigma)$$

$$(\mathrm{const\_eq}_{\alpha})\ \frac{\Gamma \Vdash_{\Sigma} \overrightarrow{\hat{M}^{\alpha}} = \overrightarrow{\hat{N}^{\alpha}} :: \Delta^{\alpha}}{\Gamma \vdash_{\Sigma} c^{\alpha}[\overrightarrow{\hat{M}^{\alpha}}] = c^{\alpha}[\overrightarrow{\hat{N}^{\alpha}}] : \{\overrightarrow{\hat{M}^{\alpha}}/\Delta^{\alpha}\}K^{\mathbf{0}}}\ (c^{\alpha} : (\Delta^{\alpha})K^{\mathbf{0}} \in \Sigma)$$

$$(\mathrm{var}_{\alpha})\ \frac{\Gamma \Vdash_{\Sigma} \overrightarrow{\hat{M}^{\alpha}} :: \Delta^{\alpha}}{\Gamma \vdash_{\Sigma} x^{\alpha}[\overrightarrow{\hat{M}^{\alpha}}/\Delta^{\alpha}] : \{\overrightarrow{\hat{M}^{\alpha}}/\operatorname{dom}\Delta^{\alpha}\}A^{\mathbf{0}}}\ (x^{\alpha} : (\Delta^{\alpha})A^{\mathbf{0}} \in \Gamma)$$

$$(\mathrm{var\_eq}_{\alpha})\ \frac{\Gamma \Vdash_{\Sigma} \overrightarrow{\hat{M}^{\alpha}} = \overrightarrow{\hat{N}^{\alpha}} :: \Delta^{\alpha}}{\Gamma \vdash_{\Sigma} x^{\alpha}[\overrightarrow{\hat{M}^{\alpha}}] = x^{\alpha}[\overrightarrow{\hat{N}^{\alpha}}] : \{\overrightarrow{;\hat{M}^{\alpha}}/\Delta^{\alpha}\}A^{\mathbf{0}}}\ (x^{\alpha} : (\Delta^{\alpha})A^{\mathbf{0}} \in \Gamma)$$

Conservativity Result

Theorem 7.1 *Let L be any of our frameworks that includes* **SDec** $(n-1)$ *if $n > 1$, but does not include* **SDec** (n)*. Then $L +$* **SDec** (n) *is a conservative extension of* **SDec** $(n-1)$.

Proof A simple inspection of the rules of deduction in $L +$ **SDec** (n) reveals that, if any of the new term constructors appears in the body of a judgement, one of the new signature or context declaration constructors must occur in the signature or context respectively. Further, if any of the new constructors occurs in the signature or context at some point in a derivation, it occurs in every judgement below that point.

Thus, if there is a derivation Δ of J in $L +$ **SDec** (n) and no new constructor occurs in the signature or context of J, no such constructor can have occurred anywhere in the derivation. Hence none of the new term constructors can have occurred in the derivation either; and so the whole derivation is a derivation of J in L. $\square$

7.2.2 Parameters of Large Kind

We introduce the features **LDec** (n) for $n \geqslant 0$. **LDec** (n) depends on **SDec** (n) and **LDec** $(n-1)$ if $n > 0$.

Grammar We introduce new sets of *large α-ary variables* (that is, α-ary variables of large kind) for each $n-1$st-order arity α (if $n > 0$), and *large α-ary constants* (that is, α-ary constants of large kind) for each nth-order arity α We shall refer to the old variables and constants as *small*. (This formal division of the variables and constants into large and small shall prove necessary later, when we add features for first-class lambda abstractions. We shall have separate features for lambda abstractions with small domains and those with large domains; to ensure that the first of these does not interfere with the feature **LDec** (n), the two must use different sets of variables and constants.)

We extend the notion of α-ary small parametric context to the class of *α-ary parametric contexts*, which are divided into large and small contexts, as follows:

If $\alpha \equiv (\alpha_1, \ldots, \alpha_m)$ is an nth-order arity, an *α-ary parametric context* is an expression of the form

$$x^{\alpha_1} : (\Theta^{\alpha_1})K_1^{\mathbf{0}}, \ldots, x^{\alpha_m} : (\Theta^{\alpha_m})K_m^{\mathbf{0}} ,$$

where each x^{α_i} is a small α_i-ary variable if Θ^{α_i} is small and $K_i^{\mathbf{0}}$ is a type; otherwise, x^{α_i} is a large α_i-ary variable. This context is *small* iff each Θ^{α_i} is small and each $K_i^{\mathbf{0}}$ is a type (iff each x^{α_i} is small); otherwise, it is *large*.

LDec (n) adds the following constructors to the grammar:

Signature Declaration	γ	$::=$	$\cdots \mid c^\alpha : (\Theta^\alpha)K^{\mathbf{0}}$
Context Declaration	δ	$::=$	$\cdots \mid x^\alpha : (\Theta^\alpha)K^{\mathbf{0}}$

where c^α (x^α) is a large α-ary context (variable), and *either* Θ^α is a large α-ary context, or $K^{\mathbf{0}}$ is a kind that is not a type. (We preserve the signature and context declaration constructors from **SDec** (n) that handle the case that Θ^α is small and $K^{\mathbf{0}}$ is a type; in this case, of course, c^α or x^α must be small.)

Defined Operations We introduce defined judgement forms for large parametric contexts:

$$\begin{array}{lll} \Gamma & \Vdash_\Sigma & \Theta_1^\alpha = \Theta_2^\alpha , \\ \Gamma & \Vdash_\Sigma & \hat{\overrightarrow{M}}^\alpha :: \Theta^\alpha , \end{array}$$

and

$$\Gamma \Vdash_\Sigma \hat{\overrightarrow{M}}^\alpha = \hat{\overrightarrow{N}}^\alpha :: \Theta^\alpha ,$$

in a similar manner to that of **SDec** (n).

We define the argument list for a large α-ary constant or variable z^α as follows:

$$z^\alpha : (\Theta^\alpha)K$$

is in the context or signature, then Θ^α is the argument list of z^α. Thus, the argument list of any α-ary constant or variable will be an α-ary parametric context. If the constant or variable is small, the context will be small; if large, the context may be either small or large.

Rules of Deduction For each nth-order arity α, we add the following rules of deduction:

$$(\mathrm{Lsig}_\alpha)\ \frac{\Theta^\alpha \vdash_\Sigma K^{\mathbf{0}}\ \mathrm{kind}}{\Sigma, c^\alpha : (\Theta^\alpha)K^{\mathbf{0}}\ \mathrm{sig}}\ (c^\alpha \notin \mathrm{dom}\,\Sigma) \qquad (\mathrm{Lctxt}_\alpha)\ \frac{\Gamma, \Theta^\alpha \vdash_\Sigma K^{\mathbf{0}}\ \mathrm{kind}}{\Gamma, x^\alpha : (\Theta^\alpha)K^{\mathbf{0}} \vdash_\Sigma \mathrm{valid}}\ (x^\alpha \notin \mathrm{dom}\,\Gamma)$$

$$(\mathrm{Lconst}_\alpha)\ \frac{\Gamma \Vdash_\Sigma \overrightarrow{\hat{M}}^\alpha :: \Theta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{\hat{M}}^\alpha] : \{\overrightarrow{\hat{M}}^\alpha/\Theta^\alpha\}K^{\mathbf{0}}}\ (c^\alpha : (\Theta^\alpha)K^{\mathbf{0}} \in \Sigma)$$

$$(\mathrm{Lconst\_eq}_\alpha)\ \frac{\Gamma \Vdash_\Sigma \overrightarrow{\hat{M}}^\alpha = \overrightarrow{\hat{N}}^\alpha :: \Theta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{\hat{M}}^\alpha = c^\alpha[\overrightarrow{\hat{N}}^\alpha] : \{\overrightarrow{\hat{M}}^\alpha/\Delta^\alpha\}K^{\mathbf{0}}}\ (c^\alpha : (\Theta^\alpha)K^{\mathbf{0}} \in \Sigma)$$

$$(\mathrm{Lvar}_\alpha)\ \frac{\Gamma \Vdash_\Sigma \overrightarrow{\hat{M}}^\alpha :: \Theta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{\hat{M}}^\alpha] : \{\overrightarrow{\hat{M}}^\alpha/\Theta^\alpha\}K^{\mathbf{0}}}\ (x^\alpha : (\Theta^\alpha)K^{\mathbf{0}} \in \Gamma)$$

$$(\mathrm{Lvar\_eq}_\alpha)\ \frac{\Gamma \Vdash_\Sigma \overrightarrow{\hat{M}}^\alpha = \overrightarrow{\hat{N}}^\alpha :: \Theta^\alpha}{\Gamma \vdash_\Sigma x^\alpha[\overrightarrow{\hat{M}}^\alpha] = x^\alpha[\overrightarrow{\hat{N}}^\alpha] : \{\overrightarrow{\hat{M}}^\alpha/\Delta^\alpha\}K^{\mathbf{0}}}\ (x^\alpha : (\Theta^\alpha)K^{\mathbf{0}} \in \Gamma)$$

Lemma 7.2 *In a framework containing* $\mathbf{LDec}\,(n)$*, the rules (sig$_\alpha$), (ctxt$_\alpha$), (const$_\alpha$), (const_eq$_\alpha$), (var$_\alpha$) and (var_eq$_\alpha$) are redundant; that is, if we remove these rules from the system, the same set of judgements are still derivable.*

Proof It is easy to see that every instance of (sig$_\alpha$) is an instance of (Lsig$_\alpha$), every instance of (ctxt$_\alpha$) an instance of (Lctxt$_\alpha$), and so forth. □

Conservativity Result

Theorem 7.3 *Let L be any of our frameworks that includes* $\mathbf{SDec}\,(n)$*, and* $\mathbf{LDec}\,(n-1)$ *if $n > 1$, but not* $\mathbf{LDec}\,(n)$*. Then $L + \mathbf{LDec}\,(n)$ is a conservative extension of L.*

Proof It is simple to show that, if any of the new signature or context declaration constructors occurs in a derivation in $L+\mathbf{LDec}\,(n)$, it occurs in every judgement below that point. Therefore, if it does not occur in the conclusion, it does not occur anywhere in the derivation. In short, if Δ is a derivation of J in $L + \mathbf{LDec}\,(n)$, and J is a judgement of L, then Δ is a derivation of J in L. □

7.3 Declaring Equations

We define the features $\mathbf{SEq}\,(0, n)$ and $\mathbf{LEq}\,(0, n)$, which allows us to declare equalities between objects in the same kind, given an nth-order list of parameters, which must be of small kind in $\mathbf{SEq}\,(0, n)$, or may be of large kind in $\mathbf{LEq}\,(0, n)$. (The meaning of the parameter 0 shall become clear later.) $\mathbf{XEq}\,(0, n)$ depends on the feature $\mathbf{XPar}(n-1)$ if $n > 0$.

For the rest of this section, let $\mathbf{X}$ be either $\mathbf{S}$ or $\mathbf{L}$. The following is the definition of the feature $\mathbf{XEq}\,(0,n)$.

Grammar If $\mathbf{X} \equiv \mathbf{S}$, we add the following constructor to the grammar for every nth-order arity α and mth-order arity β:

$$\text{Signature Declaration } \gamma \in \mathcal{C} ::= \cdots \mid (\Delta^\alpha)(M^\mathbf{0} = M^\mathbf{0} : K^\mathbf{0}) \text{ of none}$$

where Δ^α is an α-ary small pure context.

If $\mathbf{X} \equiv \mathbf{L}$, we add the following constructor to the grammar for every nth-order arity α and mth-order arity β:

$$\text{Signature Declaration } \gamma \in \mathcal{C} ::= \cdots \mid (\Theta^\alpha)(M^\mathbf{0} = M^\mathbf{0} : K^\mathbf{0}) \text{ of none}$$

where Θ^α is an α-ary large pure context.

Rules of Deduction If $\mathbf{X} \equiv \mathbf{S}$, we add the following rules of deduction for every nth-order arity α:

$$(\text{eqn\_sig}_{\alpha,\mathbf{0}})\,\frac{\Delta^\alpha \vdash_\Sigma M^\mathbf{0} : P^\mathbf{0} \quad \Delta^\alpha \vdash_\Sigma N^\mathbf{0} : P^\mathbf{0}}{\Sigma, (\Delta^\alpha)(M^\mathbf{0} = N^\mathbf{0} : P^\mathbf{0})}$$

$$(\text{eqn\_use}_{\alpha,\mathbf{0}})\,\frac{\Gamma \Vdash_\Sigma \overrightarrow{Q}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma \{\overrightarrow{Q}^\alpha/\Delta^\alpha\}M^\mathbf{0} = \{\overrightarrow{Q}^\alpha/\Delta^\alpha\}N^\mathbf{0} : \{\overrightarrow{Q}^\alpha/\Delta^\alpha\}P^\mathbf{0}}\,((\Delta^\alpha)(M^\mathbf{0} = N^\mathbf{0} : P^\mathbf{0}) \in \Sigma)$$

If $\mathbf{X} \equiv \mathbf{L}$, we add the following rules of deduction for every nth-order arity α:

$$(\text{eqn\_sig}_{\alpha,\mathbf{0}})\,\frac{\Delta^\alpha \vdash_\Sigma M^\mathbf{0} : K^\mathbf{0} \quad \Delta^\alpha \vdash_\Sigma N^\mathbf{0} : K^\mathbf{0}}{\Sigma, (\Delta^\alpha)(M^\mathbf{0} = N^\mathbf{0} : K^\mathbf{0})}$$

$$(\text{eqn\_use}_{\alpha,\mathbf{0}})\,\frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma \{\overrightarrow{M}^\alpha/\Delta^\alpha\}M^\mathbf{0} = \{\overrightarrow{M}^\alpha/\Delta^\alpha\}N^\mathbf{0} : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}K^\mathbf{0}}\,((\Delta^\alpha)(M^\mathbf{0} = N^\mathbf{0} : K^\mathbf{0}) \in \Sigma)$$

7.4 Subsystems of TF revisited

These features are sufficient to build all the weak frameworks we introduced in Part I. Specifically:

$$\begin{aligned}
\mathbf{SPar}\,(n)^- &= \mathbf{BF} + \mathbf{SDec}\,(n)\\
\mathbf{SPar}\,(n) &= \mathbf{BF} + \mathbf{SDec}\,(n) + \mathbf{SEq}\,(0,n)\\
\mathbf{LPar}\,(n)^- &= \mathbf{BF} + \mathbf{LDec}\,(n)\\
\mathbf{LPar}\,(n) &= \mathbf{BF} + \mathbf{LDec}\,(n) + \mathbf{LEq}\,(0,n)
\end{aligned}$$

for $n \in \{0, 1, \ldots, \omega\}$.

More precisely:

Theorem 7.4 *For each pair (F, F') in the list*

$$(\mathbf{SPar}\,(n)^{-}\,, \mathbf{BF} + \mathbf{SDec}\,(n)),$$
$$(\mathbf{SPar}\,(n)\,, \mathbf{BF} + \mathbf{SDec}\,(n) + \mathbf{SEq}\,(0, n)),$$
$$(\mathbf{LPar}\,(n)^{-}\,, \mathbf{BF} + \mathbf{LDec}\,(n)),$$
$$(\mathbf{LPar}\,(n)\,, \mathbf{BF} + \mathbf{LDec}\,(n) + \mathbf{LEq}\,(0, n))$$

where $n \in \mathbb{N} \cup \{\omega\}$, we have:

1. *A judgement of F is derivable in F iff it is derivable in F'.*
2. *A judgement of the form*

$$\Gamma \vdash_\Sigma K \text{ kind } \textit{or } \Gamma \vdash_\Sigma K = K'$$

is derivable in F' iff the corresponding defined judgement in derivable in F.

Proof A simple induction on derivations in either direction. □

7.5 Lambda Abstraction

We are now able to extend our systems with first-class lambda abstractions. The method we use is as follows: we introduce new syntactic classes for *α-ary objects* for arities α other than $\mathbf{0}$, and *α-ary types* and *α-ary kinds* for them to live in.

There are three choices to be made when adding lambda abstractions to a system:

- What order of object to allow — do we extend the system to include α-ary objects for all arities α, or only those of order $\leq n$ for some fixed n?
- Whether we allow the formation of *large* product kinds (we allow $(x : K_1)K_2$ for *any* kinds K_1, K_2 of the appropriate arities) or only *small* product kinds (we allow $(x : A_1)K_2$ for *types* A_1 only).
- Whether the new terms are to be identified up to $\beta\eta$-equality, or only up to β-equality.

To this end, we shall define eight infinite families of features:

$$\mathbf{X}\lambda^R\,(n)$$

where:

- n is a natural number, indicating the largest order of object that may be formed.
- $\mathbf{X}$ is either $\mathbf{S}$, indicating that only small product kinds may be formed, or $\mathbf{L}$, indicating that large product kinds may be formed.
- R is either β, indicating terms are to be identified up to β-equality, or $\beta\eta$, indicating they are to be identified up to $\beta\eta$-equality.

As with the features $\mathbf{XPar}\,(n)$, we write $\mathbf{X}\lambda^R\,(\omega)$ for the union of all the features $\mathbf{X}\lambda^T\,(R)\,n$ as n varies; including all the features in this manner gives a system in which objects of any order may be formed.

As we define it, $\mathbf{X}\lambda^R\,(n)$ shall depend on the feature $\mathbf{XPar}\,(n)$. We shall identify every term from $\mathbf{XPar}\,(n)$ with a new term; this shall be done using the the *argument lists* provided by $\mathbf{XPar}\,(n)$. We shall also take $\mathbf{L}\lambda^R\,(n)$ to depend on $\mathbf{S}\lambda^R\,(n)$.

Without further delay, we proceed with the definition of the feature $\mathbf{S}\lambda^R\,(n)$:

7.5.1 Lambda Abstractions with Small Domains

For each $n \geqslant 1$, we define the features $\mathbf{S}\lambda^R\,(n)$, where R is either β or $\beta\eta$, indicating whether we are identifying our terms up to β-convertibility or $\beta\eta$-convertibility. $\mathbf{S}\lambda^R\,(n)$ depends on $\mathbf{SPar}(n)$, and on $\mathbf{S}\lambda^R\,(n-1)$ if $n > 1$.

Grammar We introduce classes of *α-ary terms* $M^\alpha, N^\alpha, P^\alpha \in \mathcal{M}^\alpha$, *α-ary types* $A^\alpha \in \mathcal{T}^\alpha$ and *α-ary kinds* $K^\alpha \in \mathcal{K}^\alpha$ for every nth-order arity α. We take the terms, types and kinds of **BF** to be the base terms, types and kinds.

We add the following constructors for every nth-order arity $\alpha \equiv (\alpha_1)\hat{}\alpha'$. (Note that, as $n \geqslant 1$, every nth-order arity α is of this form.)

- Every α-ary constant and variable is an α-ary term.
- If x^{α_1} is an α_1-ary variable, A^{α_1} an α_1-ary type, and $N^{\alpha'}$ an α'-ary term, then
$$[x^{\alpha_1} : A^{\alpha_1}]N^{\alpha'}$$
is an $(\alpha_1)\hat{}\alpha'$-ary term.
- If $M^{(\alpha_1)\hat{}\alpha'}$ is an $(\alpha_1)\hat{}\alpha'$-ary term, and N^{α_1} an α_1-ary term, then
$$M^{(\alpha_1)\hat{}\alpha'}[N^{\alpha_1}]$$
is an α'-ary term.
- If x^{α_1} is an α_1-ary variable, A^{α_1} an α_1-ary type, and $A^{\alpha'}$ an α'-ary type, then
$$(x^{\alpha_1} : A^{\alpha_1})A^{\alpha'}$$
is an $(\alpha_1)\hat{}\alpha'$-ary type.
- Every α-ary type is an α-ary kind.
- If x^{α_1} is an α_1-ary variable, A^{α_1} an α_1-ary small kind, and $K^{\alpha'}$ an α'-ary kind, then
$$(x^{\alpha_1} : A^{\alpha_1})K^{\alpha'}$$
is an $(\alpha_1)\hat{}\alpha'$-ary kind.

- If c^α is a small α-ary constant, and K^α an α-ary kind, then

$$c^\alpha : K^\alpha$$

is a clause of c^α.

- If x^α is a small α-ary variable, and A^α an α-ary type, then

$$x^\alpha : A^\alpha$$

is a declaration of x^α.

- If M^α and N^α are α-ary terms, and K_1^α and K_2^α are α-ary kinds, then

$$M^\alpha : K^\alpha, \quad M^\alpha = N^\alpha : K_1^\alpha, \quad K_1^\alpha \text{ kind}, \quad K_1^\alpha = K_2^\alpha$$

are judgement bodies.

We identify the $(\alpha_1)\hat{}\alpha'$-ary type $(x^{\alpha_1} : A^{\alpha_1})A^{\alpha'}$ with the $(\alpha_1)\hat{}\alpha'$-ary kind $(x^{\alpha_1} : A^{\alpha_1})A^{\alpha'}$.

Redundancies $\mathbf{S}\lambda^R\,(n)$ makes redundant the clause and context declaration constructors of $\mathbf{SDec}\,(n)$. It is fairly obvious what form the redundancies must take, but we shall spell out the formal details nonetheless.

We first find, for each expression of the form $(\Delta^\alpha)A^{\mathbf{0}}$, where Δ^α is an α-ary small parametric context, an α-ary type with which to identify the expression. We shall write this α-ary type as

$$[\![(\Delta^\alpha)A^{\mathbf{0}}]\!]$$

The definition is as follows:

$$\begin{aligned} &[\![(x^{\alpha_1} : (\Delta^{\alpha_1})A_1^{\mathbf{0}}, \ldots, x^{\alpha_m} : (\Delta^{\alpha_m})A_m^{\mathbf{0}})A^{\mathbf{0}}]\!] \\ \equiv \quad &(x^{\alpha_1} : [\![(\Delta^{\alpha_1})A_1^{\mathbf{0}}]\!]) \cdots (x^{\alpha_m} : [\![(\Delta^{\alpha_m})A_m^{\mathbf{0}}]\!])A^{\mathbf{0}} \end{aligned}$$

Similarly, we define an α-ary kind

$$[\![(\Delta^\alpha)K^{\mathbf{0}}]\!]$$

for each expression of the form $(\Delta^\alpha)K^{\mathbf{0}}$.

We now identify the $\mathbf{SDec}\,(n)$ signature declaration

$$c^\alpha : (\Delta^\alpha)K^{\mathbf{0}}$$

with the signature declaration

$$c^\alpha : [\![(\Delta^\alpha)K^{\mathbf{0}}]\!]$$

and similarly context declarations.

We also make redundant the term constructors of $\mathbf{SDec}\,(n)$, using the defined operation of argument list. Let z^α be a small α-ary variable or constant, and suppose the argument list of

z^α is

$$\Delta \equiv x^{\alpha_1} : (\Delta^{\alpha_1})A^{\mathbf{0}}_1, \ldots, x^{\alpha_m} : (\Delta^{\alpha_m})A^{\mathbf{0}}_m$$

where

$$\Delta^{\alpha_i} \equiv x^{\alpha_{i1}} : (\Delta^{\alpha_{i1}})A^{\mathbf{0}}_{i1}, \ldots, x^{\alpha_{ir_i}} : (\Delta^{\alpha_{ir_i}})A^{\mathbf{0}}_{ir_i} .$$

(Recall that the argument list of a *small* variable or constant must be a small parametric context.) Let $\overrightarrow{M}^\alpha$ be an α-ary series of abstractions. We define λM^{α_i} to be the following α_i-ary term: if $T \equiv \mathrm{ut}$,

$$\lambda M^{\alpha_i} \equiv [x^{\alpha_{i1}}] \cdots [x^{\alpha_{ir_i}}]M^{\mathbf{0}}_i ;$$

if $T \equiv \mathrm{t}$,

$$\lambda M^{\alpha_i} \equiv [x^{\alpha_{i1}} : [\![(\Delta^{\alpha_{i1}})A^{\mathbf{0}}_{i1}]\!]^\dagger] \cdots [x^{\alpha_{ir_i}} : [\![(\Delta^{\alpha_{ir_i}})A^{\mathbf{0}}_{ir_i}]\!]^\dagger]M^{\mathbf{0}}_i ,$$

where

$$(A^\beta)^\dagger \equiv [\lambda M^{\alpha_1}/x^{\alpha_1}, \ldots, \lambda M^{\alpha_{i-1}}/x^{\alpha_{i-1}}]A^\beta .$$

Then we identify the old term

$$z^\alpha[\overrightarrow{M}^\alpha]$$

with the term

$$z^\alpha[\lambda M^{\alpha_1}] \cdots [\lambda M^{\alpha_m}] .$$

Defined Operations We define an *α-ary small pure context* thus: if $\alpha \equiv (\alpha_1, \ldots, \alpha_m)$, an α-ary pure context is an expression of the form

$$x^{\alpha_1} : A^{\alpha_1}, \ldots, x^{\alpha_m} : A^{\alpha_m}$$

We define the *domain list* of the α-ary kind K^α, which is an α-ary small pure context, as follows. The domain list of any base kind is the empty list. The domain list of $(x^{\alpha_1} : A^{\alpha_1})K^{(\alpha_2,\ldots,\alpha_m)}$ is $x^{\alpha_1} : A^{\alpha_1}$ followed by the domain list of $K^{(\alpha_2,\ldots,\alpha_m)}$. We define the domain list of an α-ary type similarly.

We now extend the notion of argument list by defining the argument list of c^α to be the domain list of K^α when $c^\alpha : K^\alpha$ is in the signature, and that of x^α to be the domain list of A^α when $x^\alpha : A^\alpha$ is in the context. This is a conservative extension of the definition if we identify the α-ary small parametric context

$$x^{\alpha_1} : (\Delta^{\alpha_1})A^{\mathbf{0}}_1, \ldots, x^{\alpha_m} : (\Delta^{\alpha_m})A^{\mathbf{0}}_m$$

with the α-ary small pure context

$$x^{\alpha_1} : [\![(\Delta^{\alpha_1})A^{\mathbf{0}}_1]\!] , \ldots, x^{\alpha_m} : [\![(\Delta^{\alpha_m})A^{\mathbf{0}}_m]\!]$$

Rules of Deduction For each nth-order arity $\alpha \equiv (\alpha_1)\hat{}\alpha'$, we add the following rules of deduction:

Constants

$$(\mathbf{S}\mathrm{sig}_{\alpha})\,\frac{\vdash_{\Sigma} K^{\alpha}\ \mathrm{kind}}{\Sigma, c^{\alpha} : K^{\alpha}\ \mathrm{sig}}\,(c^{\alpha} \notin \mathrm{dom}\,\Sigma) \qquad (\mathbf{S}\mathrm{const}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} \mathrm{valid}}{\Gamma \vdash_{\Sigma} c^{\alpha} : K^{\alpha}}\,(c^{\alpha} : K^{\alpha} \in \Sigma)$$

Variables

$$(\mathbf{S}\mathrm{ctxt}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} A^{\alpha}\ \mathrm{kind}}{\Gamma, x^{\alpha} : A^{\alpha} \vdash_{\Sigma} \mathrm{valid}}\,(x^{\alpha} \notin \mathrm{dom}\,\Gamma) \qquad (\mathbf{S}\mathrm{var}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} \mathrm{valid}}{\Gamma \vdash_{\Sigma} x^{\alpha} : A^{\alpha}}\,(x^{\alpha} : A^{\alpha} \in \Gamma)$$

Abstractions

$$(\mathbf{S}\mathrm{abs}_{(\alpha_1)\widehat{\ }\alpha'})\,\frac{\Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_{\Sigma} N^{\alpha'} : K^{\alpha'}}{\Gamma \vdash_{\Sigma} [x^{\alpha_1} : A^{\alpha_1}]N^{\alpha'} : (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'}}$$

$$(\mathbf{S}\mathrm{abs\_eq}_{(\alpha_1)\widehat{\ }\alpha'})\,\frac{\Gamma \vdash_{\Sigma} A^{\alpha_1} = B^{\alpha_1} \quad \Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_{\Sigma} M^{\alpha'} = N^{\alpha'} : K^{\alpha'}}{\Gamma \vdash_{\Sigma} [x^{\alpha_1} : A^{\alpha_1}]M^{\alpha'} = [x^{\alpha_1} : B^{\alpha_1}]N^{\alpha'} : (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'}}$$

Applications

$$(\mathbf{S}\mathrm{app}_{(\alpha_1)\widehat{\ }\alpha'})\,\frac{\Gamma \vdash_{\Sigma} M^{(\alpha_1)\widehat{\ }\alpha'} : (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'} \quad \Gamma \vdash_{\Sigma} N^{\alpha_1} : A^{\alpha_1}}{\Gamma \vdash_{\Sigma} M^{(\alpha_1)\widehat{\ }\alpha'}[N^{\alpha_1}] : [N^{\alpha_1}/x^{\alpha_1}]K^{\alpha'}}$$

$$(\mathbf{S}\mathrm{app\_eq}_{(\alpha_1)\widehat{\ }\alpha'})\,\frac{\begin{array}{c}\Gamma \vdash_{\Sigma} M_1^{(\alpha_1)\widehat{\ }\alpha'} = M_2^{(\alpha_1)\widehat{\ }\alpha'} : (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'} \\ \Gamma \vdash_{\Sigma} N_1^{\alpha_1} = N_2^{\alpha_1} : A^{\alpha_1}\end{array}}{\Gamma \vdash_{\Sigma} M_1^{(\alpha_1)\widehat{\ }\alpha'}[N_1^{\alpha_1}] = M_2^{(\alpha_1)\widehat{\ }\alpha'}[N_2^{\alpha_1}] : [N_1^{\alpha_1}/x^{\alpha_1}]K^{\alpha'}}$$

Product Kinds

$$(\mathbf{S}\mathrm{prod}_{(\alpha_1)\widehat{\ }\alpha'})\,\frac{\Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_{\Sigma} K^{\alpha'}\ \mathrm{kind}}{\Gamma \vdash_{\Sigma} (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'}\ \mathrm{kind}}$$

$$(\mathbf{S}\mathrm{prod\_eq}_{(\alpha_1)\widehat{\ }\alpha'})\,\frac{\begin{array}{c}\Gamma \vdash_{\Sigma} A^{\alpha_1} = B^{\alpha_1} \\ \Gamma, x^{\alpha_1} : A_1^{\alpha_1} \vdash_{\Sigma} K_1^{\alpha'} = K_2^{\alpha'}\end{array}}{\Gamma \vdash_{\Sigma} (x^{\alpha_1} : A^{\alpha_1})K_1^{\alpha'} = (x^{\alpha_1} : B^{\alpha_1})K_2^{\alpha'}}$$

Equality

$$(\mathrm{ref}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} M^{\alpha} : K^{\alpha}}{\Gamma \vdash_{\Sigma} M^{\alpha} = M^{\alpha} : K^{\alpha}} \qquad (\mathrm{sym}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} M^{\alpha} = N^{\alpha} : K^{\alpha}}{\Gamma \vdash_{\Sigma} N^{\alpha} = M^{\alpha} : K^{\alpha}}$$

$$(\mathrm{trans}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} M^{\alpha} = N^{\alpha} : K^{\alpha} \quad \Gamma \vdash_{\Sigma} N^{\alpha} = P^{\alpha} : K^{\alpha}}{\Gamma \vdash_{\Sigma} M^{\alpha} = P^{\alpha} : K^{\alpha}}$$

$$(\mathrm{Kref}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} K^{\alpha}\ \mathrm{kind}}{\Gamma \vdash_{\Sigma} K^{\alpha} = K^{\alpha}} \qquad (\mathrm{Ksym}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} K_1^{\alpha} = K_2^{\alpha}}{\Gamma \vdash_{\Sigma} K_2^{\alpha} = K_1^{\alpha}}$$

$$(\mathrm{Ktrans}_{\alpha})\,\frac{\Gamma \vdash_{\Sigma} K_1^{\alpha} = K_2^{\alpha} \quad \Gamma \vdash_{\Sigma} K_2^{\alpha} = K_3^{\alpha}}{\Gamma \vdash_{\Sigma} K_1^{\alpha} = K_3^{\alpha}}$$

$$(\mathrm{conv}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha : K_1^\alpha \quad \Gamma \vdash_\Sigma K_1^\alpha = K_2^\alpha}{\Gamma \vdash_\Sigma M^\alpha : K_2^\alpha}$$

$$(\mathrm{conv\_eq}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha = N^\alpha : K_1^\alpha \quad \Gamma \vdash_\Sigma K_1^\alpha = K_2^\alpha}{\Gamma \vdash_\Sigma M^\alpha = N^\alpha : K_2^\alpha}$$

$$(\mathrm{beta}_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_\Sigma M^{\alpha'} : K^{\alpha'} \quad \Gamma \vdash_\Sigma N^{\alpha_1} : A^{\alpha_1}}{\Gamma \vdash_\Sigma ([x^{\alpha_1} : A^{\alpha_1}]M^{\alpha'})[N^{\alpha_1}] = [N^{\alpha_1}/x^{\alpha_1}]M^{\alpha'} : [N^{\alpha_1}/x^{\alpha_1}]K^{\alpha'}}$$

If $R \equiv \beta\eta$:

$$(\mathrm{eta}_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma \vdash_\Sigma M^{(\alpha_1)\hat{}\alpha'} : (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'}}{\begin{array}{rcl}\Gamma & \vdash_\Sigma & [x^{\alpha_1} : A^{\alpha_1}]M^{(\alpha_1)\hat{}\alpha'}[x^{\alpha_1}] \\ & = & M^{(\alpha_1)\hat{}\alpha'} : (x^{\alpha_1} : A^{\alpha_1})K^{\alpha'}\end{array}}\ \left(x^{\alpha_1} \notin \mathrm{FV}\left(M^{(\alpha_1)\hat{}\alpha'}\right)\right)$$

Metatheory It is provable that this feature makes redundant the six rules of $\mathbf{SDec}\,(n)$: (sig_α), (ctxt_α), (const_α), $(\mathrm{const\_eq}_\alpha)$, (var_α), and $(\mathrm{var\_eq}_\alpha)$; and that, for any framework L including $\mathbf{SDec}\,(n)$ and $\mathbf{S}\lambda^R\,(n-1)$, $L + \mathbf{S}\lambda^R\,(n)$ is a conservative extension of L. However, in order to prove this, we must establish the connection between substitution and instantiation; and the only tool available for this purpose would appear to be untyped reduction.

To this end, we define the notion of *β-reduction* on the entities of $L + \mathbf{S}\lambda^R\,(n)$ via the following reduction:

$$([x^\alpha : A^\alpha]M^\beta)[N^\alpha] \rightsquigarrow [N^\alpha/x^\alpha]M^\beta$$

for α an arity of order $< n$, β an arity of order $\leq n$.

Let L_0 be the system $L + \mathbf{S}\lambda^R\,(n)$ with the rules of deduction (sig_α), (ctxt_α), (const_α), $(\mathrm{const\_eq}_\alpha)$, (var_α), and $(\mathrm{var\_eq}_\alpha)$ from $\mathbf{SDec}\,(n)$ removed. We shall prove shortly that this system is equivalent to $L + \mathbf{S}\lambda^R\,(n)$.

We shall need the following result:

Theorem 7.5 (Subject Reduction) *In L_0:*

1. *If $\Gamma \vdash_\Sigma M : K$ and $M \twoheadrightarrow N$, then $\Gamma \vdash_\Sigma M = N : K$.*

2. *If $\Gamma \vdash_\Sigma K$ kind and $K \twoheadrightarrow K'$, then $\Gamma \vdash_\Sigma K = K'$.*

This result is notoriously difficult to prove for a system based on judgemental equality. One method of proving such a result is given in Appendix B; this method can be adapted to any of the systems in our hierarchy. Alternatively, as all our frameworks are normalising systems, a method similar to that in [Coq91] can be used.

We also need:

Lemma 7.6 *Let α be an nth-order arity, Δ^α a small parametric context, $\overrightarrow{M}^\alpha$ an α-ary abstraction sequence, and $\overrightarrow{\lambda M}^\alpha$ the corresponding sequence of terms. For any entity X of $L + \mathbf{S}\lambda^R\,(n)$,*

$$[\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]X \twoheadrightarrow \{\overrightarrow{M}^\alpha/\Delta^\alpha\}X$$

Proof A double induction, firstly on the arity α, secondly on X. The case where X is an application of one of the variables of $\mathrm{dom}\,\Delta^\alpha$ is handled as follows. Let

$$\begin{array}{rcl} \Delta^\alpha & \equiv & x^{\alpha_1} : (\Delta^{\alpha_1})A_1^{\mathbf{0}}, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})A_n^{\mathbf{0}} \\ \overrightarrow{M}^\alpha & \equiv & [\mathrm{dom}\,\Delta^{\alpha_1}]M_1^{\mathbf{0}}, \ldots, [\mathrm{dom}\,\Delta^{\alpha_n}]M_n^{\mathbf{0}} \\ X & \equiv & x^{\alpha_i}[\overrightarrow{\lambda P}^{\alpha_i}] \end{array}$$

Then

$$\begin{array}{rcll} [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]X & \equiv & ([\Delta^{\alpha_i}]M_i^{\mathbf{0}})[[\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]\overrightarrow{\lambda P}^{\alpha_i}] & \\ & \twoheadrightarrow & ([\Delta^{\alpha_i}]M_i^{\mathbf{0}})[\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\overrightarrow{\lambda P}^{\alpha_i}] & \text{(i.h. on } X) \\ & \twoheadrightarrow & [\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\overrightarrow{\lambda P}^{\alpha_i}/\Delta^{\alpha_i}]M_i^{\mathbf{0}} & \\ & \twoheadrightarrow & \{\{\overrightarrow{M}^\alpha/\Delta^\alpha\}\overrightarrow{P}^{\alpha_i}/\Delta^{\alpha_i}\}M_i^{\mathbf{0}} & \\ & \equiv & \{\overrightarrow{M}^\alpha/\Delta^\alpha\}x^{\alpha_i}[\overrightarrow{P}^{\alpha_i}] & \end{array}$$

□

We are now able to prove:

Theorem 7.7 $\mathbf{S}\lambda^R(n)$ *makes redundant the rules of deduction (sig$_\alpha$), (ctxt$_\alpha$), (const$_\alpha$), (const_eq$_\alpha$), (var$_\alpha$), (var_eq$_\alpha$) for each nth-order arity α from* $\mathbf{SDec}\,(n)$.

Proof We shall deal with (sig$_\alpha$) and (const$_\alpha$) here. The proof for (ctxt$_\alpha$) is very similar to (sig$_\alpha$), and the proof for the other three rules is similar to (const$_\alpha$).

$$(\mathrm{sig}_\alpha)\,\frac{\Delta^\alpha \vdash_\Sigma K^{\mathbf{0}}\ \mathrm{kind}}{\Sigma, c^\alpha : (\Delta^\alpha)K^{\mathbf{0}}\ \mathrm{kind}}\,(c^\alpha \notin \mathrm{dom}\,\Sigma)$$

Suppose $\Delta^\alpha \vdash_\Sigma K^{\mathbf{0}}$ kind. By the ($\mathbf{S}$prod) rules,

$$\vdash_\Sigma (\Delta^\alpha)K^{\mathbf{0}}\ \mathrm{kind}$$

Hence, by ($\mathbf{S}$const$_\alpha$),

$$\Sigma, c^\alpha : (\Delta^\alpha)K^{\mathbf{0}}\ \mathrm{sig}\,.$$

$$(\mathrm{const}_\alpha)\,\frac{\Gamma \Vdash_\Sigma \overrightarrow{M}^\alpha :: \Delta^\alpha}{\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{M}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}K^{\mathbf{0}}}\,(c^\alpha : (\Delta^\alpha)K^{\mathbf{0}} \in \Sigma)$$

Let

$$\begin{array}{rcl} \Delta^\alpha & \equiv & x^{\alpha_1} : (\Delta^{\alpha_1})A_1^{\mathbf{0}}, \ldots, x^{\alpha_n} : (\Delta^{\alpha_n})A_n^{\mathbf{0}} \\ \overrightarrow{M}^\alpha & \equiv & [\mathrm{dom}\,\Delta^{\alpha_1}]M_1^{\mathbf{0}}, \ldots, [\mathrm{dom}\,\Delta^{\alpha_n}]M_n^{\mathbf{0}} \end{array}$$

so that

$$\overrightarrow{\lambda M}^\alpha \equiv [\Delta^{\alpha_1}]M_1^{\mathbf{0}}, [(\Delta^{\alpha_2})^*]M_2^{\mathbf{0}}, \ldots, [(\Delta^{\alpha_n})^*]M_n^{\mathbf{0}}$$

where

$$(\Delta^{\alpha_i})^* \equiv [\lambda M^{\alpha_1}/x^{\alpha_1}, \ldots, \lambda M^{\alpha_{i-1}}/x^{\alpha_{i-1}}]\Delta^{\alpha_i} .$$

We must show that

$$\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{\lambda M}^\alpha] : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}K^{\mathbf{0}}$$

Using Subject Reduction and Lemma 7.6, it is sufficient to show

$$\Gamma \vdash_\Sigma c^\alpha[\overrightarrow{\lambda M}^\alpha] : [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]K^{\mathbf{0}}$$

Now, ($\mathbf{S}$const$_\alpha$) gives us

$$\Gamma \vdash_\Sigma c^\alpha : (\Delta^\alpha)K^{\mathbf{0}}$$

Hence, by the ($\mathbf{S}$app) rules, it is sufficient to show that

$$\Gamma \vdash_\Sigma [(\Delta^{\alpha_i})^*]M^{\alpha_i} : ((\Delta^{\alpha_i})^*)(A_i^{\mathbf{0}})^*$$

and so, by the ($\mathbf{S}$abs) rules, it is sufficient to show that

$$\Gamma, (\Delta^{\alpha_i})^* \vdash_\Sigma M^{\alpha_i} : (A_i^{\mathbf{0}})^*$$

i.e.

$$\Gamma, [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]\Delta^{\alpha_i} \vdash_\Sigma M^{\alpha_i} : [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]A_i^{\mathbf{0}} .$$

The hypothesis is that

$$\Gamma, \{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i} \vdash_\Sigma M^{\alpha_i} : \{\overrightarrow{M}^\alpha/\Delta^\alpha\}A_i^{\mathbf{0}}$$

If we can show

$$\Gamma, [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]\Delta^{\alpha_i} \vdash_\Sigma [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]A_i^{\mathbf{0}} \text{ kind} \tag{7.2}$$

then Subject Reduction will give us

$$\Gamma, \Delta^\alpha \Vdash_\Sigma [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]\Delta^{\alpha_i} = \{\overrightarrow{M}^\alpha/\Delta^\alpha\}\Delta^{\alpha_i}$$

and

$$\Gamma, \Delta^\alpha, [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]\Delta^{\alpha_i} \vdash_\Sigma [\overrightarrow{\lambda M}^\alpha/\Delta^\alpha]A_i^{\mathbf{0}} = \{\overrightarrow{\lambda M}^\alpha/\Delta^\alpha\}A_i^{\mathbf{0}}$$

(7.2) will follow by the Substitution Lemma (which is provable for L_0) provided that

$$\Gamma, \Delta^\alpha, \Delta^{\alpha_i} \vdash_\Sigma A_i^{\mathbf{0}} \text{ kind}$$

This follows from Weakening and the fact that Σ is a valid signature. □

Conservativity Result We shall prove that $L + \mathbf{S}\lambda^R(n)$ is a conservative extension of L as follows. Note that every nth-order arity α is of the form

$$\alpha \equiv (\alpha_1, \ldots, \alpha_m)\hat{\ }\beta$$

where each α_i and β is an arity of order $< n$. (This decomposition is not unique.) For every object M of this arity in $L + \mathbf{S}\lambda^R(n)$, and every α_i-ary object N_i of L, we shall define the β-ary object

$$APP(M, N_1, \ldots, N_m)$$

of L. We shall prove that this object behaves in L as $M[N_1]\cdots[N_m]$ behaves in $L + \mathbf{S}\lambda^R(n)$. Taking the case $n = 0$ shall give us the result we need.

The definition of the functions APP is as follows. To handle variables and constants, we make use of the operation of domain list; therefore, APP is in fact defined with respect to a given signature and context (of $L + \mathbf{S}\lambda^R(n)$).

Definition 7.8 (APP) *For every constructor of L, we simply pass APP to the subterms. We need only give the definition for the new constructors of $\mathbf{S}\lambda^R(n)$.*

Variables are handled as follows:

$$APP(x^\alpha, N_1, \ldots, N_m) \equiv [y^{\beta_1} : (A^{\beta_1})^*] \cdots [y^{\beta_k} : (A^{\beta_k})^*] x[\overline{N_1}, \ldots, \overline{N_m}, \overline{y^{\beta_1}}, \ldots, \overline{y^{\beta_k}}]$$

where

$$x_1 : A_1, \ldots, x_m : A_m, y^{\beta_1} : A^{\beta_1}, \ldots, y^{\beta_k} : A^{\beta_k}$$

is the domain list of x^α,

$$X^* \equiv [N_1/x_1, \ldots, N_m/x_m]X\,,$$

and $\overline{X}$ denotes the η-long form of X:

$$\begin{array}{rcl} \overline{N_i} & \equiv & [z_1, \ldots, z_{r_i}]N_i[z_1]\cdots[z_{r_i}] \\ \overline{y^{\beta_j}} & \equiv & [z_1, \ldots, z_{s_j}]y^{\beta_j}[z_1]\cdots[z_{s_j}] \end{array}$$

for an appropriate variable sequence $\vec{z}$.

Constants are handled similarly.

For abstractions:

$$APP([x^{\alpha_1} : A^{\alpha_1}]M, N_1, \ldots, N_m) \equiv APP_\alpha([N_1/x]M, N_2, \ldots, N_m)$$

Applications are handled thus:

$$APP(M[N], N_1, \ldots, N_m) \equiv APP(M, N, N_1, \ldots, N_m)$$

Theorem 7.9 *Let Δ^α, Θ^β be pure contexts, α and β arities of order $< n$. Let $\vec{M}^\alpha$, $\vec{N}^\beta$ be sequences of terms of L of the appropriate arities. Let Γ be a context of L, and Σ a signature of L. Suppose*

$$\begin{array}{lll} \Gamma & \Vdash^L_\Sigma & \vec{M}^\alpha :: \Delta^\alpha \\ \Gamma & \Vdash^L_\Sigma & \vec{N}^\beta :: \Theta^\beta \end{array}$$

1. *If Σ sig is derivable in $L + \mathbf{S}\lambda^R(n)$, it is derivable in L.*

2. *If* $\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)}$ valid *then* $\Gamma \vdash_\Sigma^L$ valid.

3. *If* $\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)} P : (\Theta^\beta)K$*, then*

$$\Gamma \vdash_\Sigma^L APP([\vec{M}^\alpha/\Delta^\alpha]P, \vec{N}^\beta) : [\vec{M}^\alpha/\Delta^\alpha, \vec{N}^\beta/\Theta^\beta]K$$

4. *If* $\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)} P = Q : (\Theta^\beta)K$*, then*

$$\Gamma \vdash_\Sigma^L APP([\vec{M}^\alpha/\Delta^\alpha]P, \vec{N}^\beta) = APP([\vec{M}^\alpha/\Delta^\alpha]Q, \vec{N}^\beta) : [\vec{M}^\alpha/\Delta^\alpha, \vec{N}^\beta/\Theta^\beta]K$$

5. *If* $\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)} (\Theta^\beta)K$ kind*, then*

$$\Gamma \vdash_\Sigma^L [\vec{M}^\alpha/\Delta^\alpha, \vec{N}^\beta/\Theta^\beta]K \text{ kind}$$

6. *If* $\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)} (\Theta^\beta)K = (\Theta^\beta)K'$*, then*

$$\Gamma \vdash_\Sigma^L [\vec{M}^\alpha/\Delta^\alpha, \vec{N}^\beta/\Theta^\beta]K = [\vec{M}^\alpha/\Delta^\alpha, \vec{N}^\beta/\Theta^\beta]K'$$

Proof Induction on the derivation of the premise; by Theorem 7.7, we need not deal with the cases of the rules of **SDec** (n). The following are the most difficult cases:

$$(\mathbf{S}\text{var}) \frac{\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)} \text{valid}}{\Gamma, \Delta^\alpha \vdash_\Sigma^{L+\mathbf{S}\lambda^R(n)} x^{\beta\gamma} : (\Theta^\beta)(\Phi^\gamma)K} \ (x^{\beta\gamma} : (\Theta^\beta, \Phi^\gamma)K \in \Gamma)$$

We must show that

$$\Gamma \vdash_\Sigma^L [[\vec{N}^\beta/\Theta^\beta]\Phi^\gamma]x^{\beta\gamma}[\overrightarrow{\overline{N}}^\beta, \overline{\Phi}^\gamma] : [\vec{N}^\beta/\Theta^\beta](\Phi^\gamma)K$$

By the ($\mathbf{S}$abs) rules, it is sufficient to show

$$\Gamma, [\vec{N}^\beta/\Theta^\beta]\Phi^\gamma \vdash_\Sigma^L x^{\beta\gamma}[\overrightarrow{\overline{N}}^\beta, \overline{\Phi}^\gamma] : [\vec{N}^\beta/\Theta^\beta]K$$

By Substitution, it is sufficient to show

$$\Gamma, \Theta^\beta, \Phi^\gamma \vdash_\Sigma^L x^{\beta\gamma}[\overline{\Theta}^\beta, \overline{\Phi}^\gamma] : K$$

which is easy to do.

For the rules ($\mathbf{S}$abs) and ($\mathbf{S}$app), the result is immediate from the induction hypothesis.

The cases for ($\mathbf{S}$beta) and ($\mathbf{S}$eta) follow from the fact that

$$\begin{aligned}
APP(([x:A]M)[N], P_1, \ldots, P_n) &\equiv APP([x:A]M, N, P_1, \ldots, P_n) \\
&\equiv APP([N/x]M, P_1, \ldots, P_n) \\
APP([x:A]M[x], P_1, \ldots, P_n) &\equiv APP(M[P_1], P_2, \ldots, P_n) \\
&\equiv APP(M, P_1, \ldots, P_n)
\end{aligned}$$

□

Corollary 7.10 (Conservativity) *If a judgement of L is derivable in $L+\mathbf{S}\lambda^R(n)$, it is derivable in L.*

Proof Simply take $\alpha \equiv \beta \equiv \mathbf{0}$ in the above theorem. □

7.5.2 Lambda Abstractions with Large Domains

For $n \geqslant 1$, we introduce the features $\mathbf{L}\lambda^R(n)$, where again R is either β or $\beta\eta$. $\mathbf{L}\lambda^R(n)$ depends on $\mathbf{LDec}(n)$, on $\mathbf{S}\lambda^R(n)$, and on $\mathbf{L}\lambda^R(n-1)$ if $n > 1$.

Grammar For every nth-order arity $\alpha \equiv (\alpha_1, \ldots, \alpha_m)$, we introduce the following constructors to the grammar:

- If x^{α_1} is an α_1-ary variable, K^{α_1} an α_1-ary kind, and $M^{\alpha'}$ an α'-ary term, then

$$[x^{\alpha_1} : K^{\alpha_1}]M^{\alpha'}$$

  is an $(\alpha_1)\hat{\ }\alpha'$-ary term.

- If x^{α_1} is an α_1-ary variable, K^{α_1} an α_1-ary kind, and $K^{\alpha'}$ an α'-ary kind, then

$$(x^{\alpha_1} : K^{\alpha_1})K^{\alpha'}$$

  is an $(\alpha_1)\hat{\ }\alpha'$-ary kind.

- If x^{α} is an α-ary variable, and K^{α} an α-ary kind, then

$$x^{\alpha} : K^{\alpha}$$

  is a declaration of x^{α}.

We make the constructors of $\mathbf{S}\lambda^R(n)$ redundant in the obvious manner. We also make the constructors of $\mathbf{LDec}(n)$ redundant in exactly the same manner as *lamSRn* made the constructors of $\mathbf{SDec}(n)$ redundant. We first define an α-ary kind

$$[\![(\Theta^{\alpha})K^{\mathbf{0}}]\!]$$

with which to associate every expression of the form

$$(\Theta^{\alpha})K^{\mathbf{0}}.$$

The definition is as follows:

$$\begin{array}{l} [\![(x^{\alpha_1} : (\Theta^{\alpha_1})K_1^{\mathbf{0}}, \ldots, x^{\alpha_m} : (\Theta^{\alpha_m})K_m^{\mathbf{0}})K^{\mathbf{0}}]\!] \\ \quad \equiv \quad (x^{\alpha_1} : [\![(\Theta^{\alpha_1})K_1^{\mathbf{0}}]\!]) \cdots (x^{\alpha_m} : [\![(\Theta^{\alpha_m})K_m^{\mathbf{0}}]\!])K^{\mathbf{0}} \end{array}$$

We now make the signature and context declaration constructors of **LDec** (n) redundant by identifying the signature declaration

$$c^\alpha : (\Theta^\alpha)K^{\mathbf{0}}$$

with the signature declaration

$$c^\alpha : [\![(\Theta^\alpha)K^{\mathbf{0}}]\!]$$

and similarly context declarations.

We also make the term constructors of **LDec** (n) (inherited from **SDec** (n)) redundant. If the argument list of z^α is

$$x^{\alpha_1} : (\Theta^{\alpha_1})K_1^{\mathbf{0}}, \ldots, x^{\alpha_m} : (\Theta^{\alpha_m})K_m^{\mathbf{0}}$$

where

$$\Theta^{\alpha_i} \equiv x^{\alpha_{i1}} : (\Theta^{\alpha_{i1}})K_{i1}^{\mathbf{0}}, \ldots, x^{\alpha_{ir_1}} : (\Theta^{\alpha_{ir_i}})K_{ir_i}^{\mathbf{0}},$$

then we identify the term

$$z^\alpha[[x^{\alpha_{11}}, \ldots, x^{\alpha_{1r_1}}]M_1^{\mathbf{0}}, \ldots, [x^{\alpha_{m1}}, \ldots, x^{\alpha_{mr_m}}]M_m^{\mathbf{0}}]$$

with the term

$$\begin{aligned} z^\alpha[&[x^{\alpha_{11}} : [\![(\Theta^{\alpha_{11}})K_{11}^{\mathbf{0}}]\!]] \cdots [x^{\alpha_{1r_1}} : [\![(\Theta^{\alpha_{1r_1}})K_{1r_1}^{\mathbf{0}}]\!]]M_1^{\mathbf{0}}] \\ &\cdots \\ &[[x^{\alpha_{m1}} : [\![(\Theta^{\alpha_{m1}})K_{m1}^{\mathbf{0}}]\!]] \cdots [x^{\alpha_{mr_m}} : [\![(\Theta^{\alpha_{mr_m}})K_{mr_m}^{\mathbf{0}}]\!]]M_m^{\mathbf{0}}] \end{aligned}$$

Defined Operations We define an *α-ary large pure context* to be an expression of the form

$$x^{\alpha_1} : K^{\alpha_1}, \ldots, x^{\alpha_m} : K^{\alpha_m}$$

where $\alpha \equiv (\alpha_1, \ldots, \alpha_m)$, x^{α_i} is an α_i-ary variable, and K^{α_i} an α_i-ary kind.

We define the *domain list* of every α-ary kind K^α, which is an α-ary large pure context, as follows. The domain list of any base kind is the empty list. The domain list of $(x^{\alpha_1} : K^{\alpha_1})K^{\alpha'}$ is $x^{\alpha_1} : K^{\alpha_1}$ followed by the domain list of $K^{\alpha'}$.

We now extend the notion of argument list by defining the argument list of z^α to be the domain list of K^α when $z^\alpha : K^\alpha$ is in the context or signature.

Rules of Deduction

Constants

$$(\mathbf{L}\mathrm{sig}_\alpha)\,\frac{\vdash_\Sigma K^\alpha \text{ kind}}{\Sigma, c^\alpha : K^\alpha \text{ sig}}\,(c^\alpha \notin \mathrm{dom}\,\Sigma) \qquad (\mathbf{L}\mathrm{const}_\alpha)\,\frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma c^\alpha : K^\alpha}\,(c^\alpha : K^\alpha \in \Sigma)$$

Variables

$$(\mathbf{L}\text{ctxt}_\alpha)\ \frac{\Gamma \vdash_\Sigma K^\alpha \text{ kind}}{\Gamma, x^\alpha : K^\alpha \vdash_\Sigma \text{valid}}\ (x^\alpha \notin \mathrm{dom}\,\Gamma) \qquad (\mathbf{L}\text{var}_\alpha)\ \frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma x^\alpha : K^\alpha}\ (x^\alpha : K^\alpha \in \Gamma)$$

Abstractions

$$(\mathbf{L}\text{abs}^{\text{t}}_{(\alpha_1)\widehat{\ }\alpha'})\ \frac{\Gamma, x^{\alpha_1} : K^{\alpha_1} \vdash_\Sigma M^{\alpha'} : K^{\alpha'}}{\Gamma \vdash_\Sigma [x^{\alpha_1} : K^{\alpha_1}]M^{\alpha'} : (x^{\alpha_1} : K^{\alpha_1})K^{\alpha'}}$$

$$(\mathbf{L}\text{abs\_eq}^{\text{t}}_{(\alpha_1)\widehat{\ }\alpha'})\ \frac{\Gamma \vdash_\Sigma K_1^{\alpha_1} = K_2^{\alpha_1} \quad \Gamma, x^{\alpha_1} : K_1^{\alpha_1} \vdash_\Sigma M^{\alpha'} = N^{\alpha'} : K^{\alpha'}}{\Gamma \vdash_\Sigma [x^{\alpha_1} : K_1^{\alpha_1}]M^{\alpha'} = [x^{\alpha_1} : K_2^{\alpha_1}]N^{\alpha'} : (x^{\alpha_1} : K_1^{\alpha_1})K^{\alpha'}}$$

Applications

$$(\mathbf{L}\text{app}_{(\alpha_1)\widehat{\ }\alpha'})\ \frac{\Gamma \vdash_\Sigma M^{(\alpha_1)\widehat{\ }\alpha'} : (x^{\alpha_1} : K^{\alpha_1})K^{\alpha'} \quad \Gamma \vdash_\Sigma N^{\alpha_1} : K^{\alpha_1}}{\Gamma \vdash_\Sigma M^{(\alpha_1)\widehat{\ }\alpha'}[N^{\alpha_1}] : [N^{\alpha_1}/x^{\alpha_1}]K^{\alpha'}}$$

$$(\mathbf{L}\text{app\_eq}_{(\alpha_1)\widehat{\ }\alpha'})\ \frac{\begin{array}{c}\Gamma \vdash_\Sigma M_1^{(\alpha_1)\widehat{\ }\alpha'} = M_2^{(\alpha_1)\widehat{\ }\alpha'} : (x^{\alpha_1} : K^{\alpha_1})K^{\alpha'} \\ \Gamma \vdash_\Sigma N_1^{\alpha_1} = N_2^{\alpha_1} : K^{\alpha_1}\end{array}}{\Gamma \vdash_\Sigma M_1^{(\alpha_1)\widehat{\ }\alpha'}[N_1^{\alpha_1}] = M_2^{(\alpha_1)\widehat{\ }\alpha'}[N_2^{\alpha_1}] : [N_1^{\alpha_1}/x^{\alpha_1}]K^{\alpha'}}$$

Product Kinds

$$(\mathbf{L}\text{prod}_{(\alpha_1)\widehat{\ }\alpha'})\ \frac{\Gamma, x^{\alpha_1} : K^{\alpha_1} \vdash_\Sigma K^{\alpha'} \text{ kind}}{\Gamma \vdash_\Sigma (x^{\alpha_1} : K^{\alpha_1})K^{\alpha'} \text{ kind}}$$

$$(\mathbf{L}\text{prod\_eq}_{(\alpha_1)\widehat{\ }\alpha'})\ \frac{\Gamma \vdash_\Sigma K_1^{\alpha_1} = K_2^{\alpha_1} \quad \Gamma x^{\alpha_1} : K_1^{\alpha_1} \vdash_\Sigma K_1^{\alpha'} = K^{\alpha'}}{\Gamma \vdash_\Sigma (x^{\alpha_1} : K_1^{\alpha_1})K_1^{\alpha'} = (x^{\alpha_1} : K_2^{\alpha_1})K_2^{\alpha'}}$$

Equality

$$(\text{ref}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha : K^\alpha}{\Gamma \vdash_\Sigma M^\alpha = M^\alpha : K^\alpha} \qquad (\text{sym}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha = N^\alpha : K^\alpha}{\Gamma \vdash_\Sigma N^\alpha = M^\alpha : K^\alpha}$$

$$(\text{trans}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha = N^\alpha : K^\alpha \quad \Gamma \vdash_\Sigma N^\alpha = P^\alpha : K^\alpha}{\Gamma \vdash_\Sigma M^\alpha = P^\alpha : K^\alpha}$$

$$(\text{Kref}_\alpha)\ \frac{\Gamma \vdash_\Sigma K^\alpha \text{ kind}}{\Gamma \vdash_\Sigma K^\alpha = K^\alpha} \qquad (\text{Ksym}_\alpha)\ \frac{\Gamma \vdash_\Sigma K_1^\alpha = K_2^\alpha}{\Gamma \vdash_\Sigma K_2^\alpha = K_1^\alpha}$$

$$(\text{Ktrans}_\alpha)\ \frac{\Gamma \vdash_\Sigma K_1^\alpha = K_2^\alpha \quad \Gamma \vdash_\Sigma K_2^\alpha = K_3^\alpha}{\Gamma \vdash_\Sigma K_1^\alpha = K_3^\alpha}$$

$$(\text{conv}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha : K_1^\alpha \quad \Gamma \vdash_\Sigma K_1^\alpha = K_2^\alpha}{\Gamma \vdash_\Sigma M^\alpha : K_2^\alpha}$$

$$(\text{conv\_eq}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^\alpha = N^\alpha : K_1^\alpha \quad \Gamma \vdash_\Sigma K_1^\alpha = K_2^\alpha}{\Gamma \vdash_\Sigma M^\alpha = N^\alpha : K_2^\alpha}$$

$$(\text{beta}_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma, x^{\alpha_1} : K^{\alpha_1} \vdash_\Sigma M^{\alpha'} : K^{\alpha'} \quad \Gamma \vdash_\Sigma N^{\alpha_1} : K^{\alpha_1}}{\Gamma \vdash_\Sigma ([x^{\alpha_1} : K^{\alpha_1}]M^{\alpha'})[N^{\alpha_1}] = [N^{\alpha_1}/x^{\alpha_1}]M^{\alpha'} : [N^{\alpha_1}/x^{\alpha_1}]K^{\alpha'}}$$

If $R \equiv \beta\eta$:

$$(\text{eta}_\alpha)\ \frac{\Gamma \vdash_\Sigma M^{(\alpha_1)\hat{}\alpha'} : (x^{\alpha_1} : K^{\alpha_1})K^{\alpha'}}{\begin{array}{lll}\Gamma & \vdash_\Sigma & [x^{\alpha_1} : K^{\alpha_1}]M^{(\alpha_1)\hat{}\alpha'}[x^{\alpha_1}] = M^{(\alpha_1)\hat{}\alpha'} \\ & : & (x^{\alpha_1} : K^{\alpha_1})K^{\alpha'}\end{array}}\ \left(x^{\alpha_1} \notin \text{FV}\left(M^{(\alpha_1)\hat{}\alpha'}\right)\right)$$

These rules make redundant the rules (Lsig$_\alpha$), (Lctxt$_\alpha$), (Lconst$_\alpha$), (Lconst_eq$_\alpha$), (Lvar$_\alpha$) and (Lvar_eq$_\alpha$). This fact, and the proof that $L + \mathbf{L}\lambda^R(n)$ is conservative over L for each appropriate framework L, is proven similarly to the corresponding results for $\mathbf{S}\lambda^R(n)$.

7.6 Declaring Equations

We define the feature $\mathbf{XEq}(m, n)$, which allows us to declare equalities between β-ary objects in the same kind, where β is an mth-order arity, given an nth-order list of parameters, which must be of small kind if $\mathbf{X} \equiv \mathbf{S}$, or may be of large kind if $\mathbf{X} \equiv \mathbf{L}$. $\mathbf{XEq}(m, n)$ requires one of the features $\mathbf{X}\lambda^R(n)$ to be included if $n > 0$, as well as $\mathbf{XPar}(m-1)$ if $m > 0$.

Grammar If $\mathbf{X} \equiv \mathbf{S}$, we add the following constructor to the grammar for every nth-order arity α and mth-order arity β:

$$\text{Signature Declaration } \gamma \in \mathcal{C} ::= \cdots \mid (\Delta^\alpha)(M^\beta = M^\beta : K^\beta) \text{ of none}$$

where Δ^α is an α-ary small pure context.

If $\mathbf{X} \equiv \mathbf{L}$, we add the following constructor to the grammar for every nth-order arity α and mth-order arity β:

$$\text{Signature Declaration } \gamma \in \mathcal{C} ::= \cdots \mid (\Theta^\alpha)(M^\beta = M^\beta : K^\beta) \text{ of none}$$

where Θ^α is an α-ary large pure context.

Rules of Deduction If $\mathbf{X} \equiv \mathbf{S}$, we add the following rules of deduction for every nth-order arity α and mth-order arity β:

$$(\text{eqn\_sig}_{\alpha,\beta})\ \frac{\Delta^\alpha \vdash_\Sigma M^\beta : K^\beta \quad \Delta^\alpha \vdash_\Sigma N^\beta : K^\beta}{\Sigma, (\Delta^\alpha)(M^\beta = N^\beta : K^\beta)}$$

$$(\text{eqn\_use}_{\alpha,\beta})\ \frac{\Gamma \Vdash_\Sigma \hat{\overrightarrow{M}}^\alpha :: \Delta^\alpha}{\begin{array}{lll}\Gamma & \vdash_\Sigma & \{\hat{\overrightarrow{M}}^\alpha/\Delta^\alpha\}M^\beta = \{\hat{\overrightarrow{M}}^\alpha/\Delta^\alpha\}N^\beta \\ & : & \{\hat{\overrightarrow{M}}^\alpha/\Delta^\alpha\}K^\beta\end{array}}\ ((\Delta^\alpha)(M^\beta = N^\beta : K^\beta) \in \Sigma)$$

It is quite trivial to prove $L + \mathbf{XEq}(m, n)$ conservative over L; any derivation whose conclusion does not contain any equation in its signature must not contain an equation anywhere, and so is a derivation of L.

Chapter 8

Existing Logical Frameworks Revisited

We are now able to revisit the results of Chapter 6 from a new perspective. Rather than embedding a weak framework in a strong one, we shall show how features could be added to the weak framework to create a system equivalent to the strong one. We shall only do this for ELF and Martin-Löf's Logical Framework here; we discuss the form that the features for the other systems of Chapter 6 should take in the concluding chapter 9.

8.1 The Edinburgh Logical Framework

8.1.1 Families of Kinds

We introduce the feature **FKnd** to handle ELF's inclusion of families of types. **FKnd** introduces to the framework functions whose values are types. It depends on **SPar** (ω). We could write versions that extend the finite orders of **SPar**, or the **LPar** features, but we shall have no need of such. This feature uses typed lambda-abstractions and β-reduction only; again, other versions could easily be written.

The feature, as we give it, depends on $\mathbf{S}\lambda^{T}\,(R)\,\omega$.

Grammar Introduce classes of *α-ary families of β-ary types* $F^{\beta}_{\alpha}, G^{\beta}_{\alpha}, H^{\beta}_{\alpha} \in \mathcal{F}^{\beta}_{\alpha}$ for every arity β and every non-$\mathbf{0}$ arity α. we take the class of β-ary types to be the class of base families of β-ary types. We also introduce classes of *α-ary superkinds* $S^{\alpha} \in \mathcal{S}^{\alpha}$. We add the following constructors to the grammar:

- If x^{α_1} is an α_1-ary variable, A^{α_1} an α_1-ary type, and $F^{\alpha'}_{\beta}$ an α'-ary family of β-ary types, then
$$[x^{\alpha_1} : A^{\alpha_1}]F^{\alpha'}_{\beta}$$
is an $(\alpha_1)\hat{\ }\alpha'$-ary family of β-ary types.

- If $F_\beta^{(\alpha_1)\hat{}\alpha'}$ is an $(\alpha_1)\hat{}\alpha'$-ary family of β-ary types, and M^{α_1} an α_1-ary term, then
$$F_\beta^{(\alpha_1)\hat{}\alpha'}[M^{\alpha_1}]$$
is an α'-ary family of β-ary types.
- **Kind** is a base superkind.
- If x^{α_1} is an α_1-ary variable, A^{α_1} an α_1-ary type, and S an α'-ary superkind, then
$$(x^{\alpha_1} : A^{\alpha_1})S^{\alpha'}$$
is an $(\alpha_1)\hat{}\alpha'$-ary superkind.
- If F_β^α is an α-ary family of β-ary types, and S^α an α-ary superkind. then $F_\beta^\alpha : S^\alpha$ is a judgement body.
- If F_β^α and G_β^α are α-ary families of β-ary types, and S^α an α-ary superkind, then $F_\beta^\alpha = G_\beta^\alpha : S^\alpha$ is a judgement body.
- If S^α is an α-ary superkind, then S^α skind is a judgement body.
- If S_1^α and S_2^α are α-ary superkinds, then $S_1^\alpha = S_2^\alpha$ is a judgement body.

Rules of Deduction

$$(\text{kind\_fam}_\beta)\ \frac{\Gamma \vdash_\Sigma A^\beta \text{ kind}}{\Gamma \vdash_\Sigma A^\beta : \mathbf{Kind}} \qquad (\text{kind\_fam\_eq}_\beta)\ \frac{\Gamma \vdash_\Sigma A^\beta = B^\beta}{\Gamma \vdash_\Sigma A^\beta = B^\beta : \mathbf{Kind}}$$

$$(\text{fam\_kind}_\beta)\ \frac{\Gamma \vdash_\Sigma A^\beta : \mathbf{Kind}}{\Gamma \vdash_\Sigma A^\beta \text{ kind}} \qquad (\text{fam\_kind\_eq}_\beta)\ \frac{\Gamma \vdash_\Sigma A^\beta = B^\beta : \mathbf{Kind}}{\Gamma \vdash_\Sigma A^\beta = B^\beta}$$

$$(\text{kind})\ \frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma \mathbf{Kind} \text{ skind}}$$

$$(\text{prod\_skind}_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_\Sigma S^{\alpha'} \text{ skind}}{\Gamma \vdash_\Sigma (x^{\alpha_1} : A^{\alpha_1})S^{\alpha'} \text{ skind}}$$

$$(\text{prod\_skind\_eq}_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma \vdash_\Sigma A^{\alpha_1} = B^{\alpha_1} \quad \Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_\Sigma S_1^{\alpha'} = S_2^{\alpha'}}{\Gamma \vdash_\Sigma (x^{\alpha_1} : A^{\alpha_1})S_1^{\alpha'} = (x^{\alpha_1} : B^{\alpha_1})S_2^{\alpha'}}$$

$$(\text{lambda\_fam}^\beta_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_\Sigma F^\beta_{\alpha'} : S^{\alpha'}}{\Gamma \vdash_\Sigma [x^{\alpha_1} : A^{\alpha_1}]F^\beta_{\alpha'} : (x^{\alpha_1} : A^{\alpha_1})S^{\alpha'}}$$

$$(\text{lambda\_fam\_eq}^\beta_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma \vdash_\Sigma A^{\alpha_1} = B^{\alpha_1} \quad \Gamma, x^{\alpha_1} : A^{\alpha_1} \vdash_\Sigma F^\beta_{\alpha'} = G^\beta_{\alpha'} : S^{\alpha'}}{\Gamma \vdash_\Sigma [x^{\alpha_1} : A^{\alpha_1}]F^\beta_{\alpha'} = [x^{\alpha_1} : B^{\alpha_1}]G^\beta_{\alpha'} : (x^{\alpha_1} : A^{\alpha_1})S^{\alpha'}}$$

$$(\text{app\_fam}^\beta_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma \vdash_\Sigma F^\beta_{(\alpha_1)\hat{}\alpha'} : (x^{\alpha_1} : A^{\alpha_1})S^{\alpha'} \quad \Gamma \vdash_\Sigma M^{\alpha_1} : A^{\alpha_1}}{\Gamma \vdash_\Sigma F^\beta_{(\alpha_1)\hat{}\alpha'}[M^{\alpha_1}] : [M^{\alpha_1}/x^{\alpha_1}]S^{\alpha'}}$$

$$(\mathrm{app\_fam\_eq}^{\beta}_{(\alpha_1)\hat{}\alpha'})\ \frac{\Gamma \vdash_\Sigma F^{\beta}_{(\alpha_1)\hat{}\alpha'} = G^{\beta}_{(\alpha_1)\hat{}\alpha'} : (x^{\alpha_1} : A^{\alpha_1})S^{\alpha'} \quad \Gamma \vdash_\Sigma M^{\alpha_1} = N^{\alpha_1} : A^{\alpha_1}}{\Gamma \vdash_\Sigma F^{\beta}_{(\alpha_1)\hat{}\alpha'}[M^{\alpha_1}] = G^{\beta}_{(\alpha_1)\hat{}\alpha'}[N^{\alpha_1}] : [M^{\alpha_1}/x^{\alpha_1}]S^{\alpha'}}$$

$$(\mathrm{fam\_ref}^{\beta}_{\alpha})\ \frac{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} : S^{\alpha}}{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} = F^{\beta}_{\alpha} : S^{\alpha}} \qquad (\mathrm{fam\_sym}^{\beta}_{\alpha})\ \frac{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} = G^{\beta}_{\alpha} : S^{\alpha}}{\Gamma \vdash_\Sigma G^{\beta}_{\alpha} = F^{\beta}_{\alpha} : S^{\alpha}}$$

$$(\mathrm{fam\_trans}^{\beta}_{\alpha})\ \frac{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} = G^{\beta}_{\alpha} : S^{\alpha} \quad \Gamma \vdash_\Sigma G^{\beta}_{\alpha} = H^{\beta}_{\alpha} : S^{\alpha}}{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} = H^{\beta}_{\alpha} : S^{\alpha}}$$

$$(\mathrm{skind\_ref}_{\alpha})\ \frac{\Gamma \vdash_\Sigma S^{\alpha}\ \mathrm{skind}}{\Gamma \vdash_\Sigma S^{\alpha} = S^{\alpha}} \qquad (\mathrm{skind\_sym}_{\alpha})\ \frac{\Gamma \vdash_\Sigma S^{\alpha}_1 = S^{\alpha}_2}{\Gamma \vdash_\Sigma S^{\alpha}_2 = S^{\alpha}_1}$$

$$(\mathrm{skind\_trans}_{\alpha})\ \frac{\Gamma \vdash_\Sigma S^{\alpha}_1 = S^{\alpha}_2 \quad \Gamma \vdash_\Sigma S^{\alpha}_2 = S^{\alpha}_3}{\Gamma \vdash_\Sigma S^{\alpha}_1 = S^{\alpha}_3}$$

$$(\mathrm{conv\_fam}^{\beta}_{\alpha})\ \frac{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} : S^{\alpha}_1 \quad \Gamma \vdash_\Sigma S^{\alpha}_1 = S^{\alpha}_2}{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} : S^{\alpha}_2}$$

$$(\mathrm{conv\_fam\_eq}^{\beta}_{\alpha})\ \frac{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} = G^{\beta}_{\alpha} : S^{\alpha}_1 \quad \Gamma \vdash_\Sigma S^{\alpha}_1 = S^{\alpha}_2}{\Gamma \vdash_\Sigma F^{\beta}_{\alpha} = G^{\beta}_{\alpha} : S^{\alpha}_2}$$

We can now give the framework in our hierarchy that we claim is equivalent to ELF:

$$F_{ELF} = \mathbf{BF} + \mathbf{SPar}\,(\omega) + \mathbf{S}\lambda^{\beta}\,(\omega) + \mathbf{FKnd}\,.$$

We firstly divide the families of ELF that are not of the form $\Pi x : A.B$ into the *abstraction-like* (those that will be translated to families of types) and the *object-like* (those that will be translated to types). The definition is as follows:

- a is object-like.
- $\lambda x : A.B$ is abstraction-like.
- AM is of the same sort as A.

We also assign arities to every kind, family and object of ELF within a given signature and context. We assign a *content arity* $\mathrm{CAr}\,(K)^{\Sigma}_{\Gamma}$ to each kind K, an *argument arity* $\mathrm{AAr}\,(M)^{\Sigma}_{\Gamma}$ to each object M, and a content arity and argument arity $\mathrm{CAr}\,(A)^{\Sigma}_{\Gamma}$ and $\mathrm{AAr}\,(A)^{\Sigma}_{\Gamma}$ to each family A, provided:

- Every constant in K, M or A is in $\mathrm{dom}\,\Sigma$, and every free variable is in $\mathrm{dom}\,\Gamma$.
- Every constant in Γ is in $\mathrm{dom}\,\Sigma$.
- Σ and Γ are consistent.

Intuitively, the meanings of these operations are as follows:

- An object with argument arity α is an α-ary function that returns an entity of the object logic.

- A family with argument arity α and content arity β is an α-ary function that returns a type whose members are objects of argument arity β.
- A kind with content arity α is a kind whose members are families of argument arity α and content arity $\mathbf{0}$.

The definition is by triple recursion, first on Σ, second on Γ, thirdly on K, M or A itself. The definition is as follows:

$$\begin{array}{rcl}
\mathrm{CAr}\,(\mathbf{Type})^{\Sigma}_{\Gamma} & \equiv & \mathbf{0} \\
\mathrm{CAr}\,(\Pi x : A.K)^{\Sigma}_{\Gamma} & \equiv & (\mathrm{CAr}\,(A)^{\Sigma}_{\Gamma})\hat{}\,\mathrm{AAr}\,(K)^{\Sigma}_{\Gamma,x:A} \\
\\
\mathrm{CAr}\,(a)^{\Sigma}_{\Gamma} & \equiv & \mathbf{0} \\
\mathrm{CAr}\,(\Pi x : A.B)^{\Sigma}_{\Gamma} & \equiv & (\mathrm{CAr}\,(A)^{\Sigma}_{\Gamma})\hat{}\,\mathrm{CAr}\,(B)^{\Sigma}_{\Gamma,x:A} \\
\mathrm{CAr}\,(\lambda x : A.B)^{\Sigma}_{\Gamma} & \equiv & \mathrm{CAr}\,(B)^{\Sigma}_{\Gamma,x:A} \\
\mathrm{CAr}\,(AM)^{\Sigma}_{\Gamma} & \equiv & \mathrm{CAr}\,(A)^{\Sigma}_{\Gamma} \\
\\
\mathrm{AAr}\,(a)^{\Sigma_0,a:K,\Sigma_1}_{\Gamma} & \equiv & \mathrm{CAr}\,(K)^{\Sigma_0}_{\langle\rangle} \\
\mathrm{AAr}\,(\Pi x : A.B)^{\Sigma}_{\Gamma} & \equiv & \mathbf{0} \\
\mathrm{AAr}\,(\lambda x : A.B)^{\Sigma}_{\Gamma} & \simeq & (\mathrm{CAr}\,(A)^{\Sigma}_{\Gamma})\hat{}\,\mathrm{AAr}\,(B)^{\Sigma}_{\Gamma,x:A} \\
\mathrm{AAr}\,(AM)^{\Sigma}_{\Gamma} & \simeq & \mathrm{AAr}\,(A)^{\Sigma}_{\Gamma} \text{ with its first entry removed} \\
\\
\mathrm{AAr}\,(c)^{\Sigma_0,c:A,\Sigma_1}_{\Gamma} & \equiv & \mathrm{CAr}\,(A)^{\Sigma_0}_{\langle\rangle} \\
\mathrm{AAr}\,(x)^{\Sigma}_{\Gamma_0,x:A,\Gamma_1} & \equiv & \mathrm{CAr}\,(A)^{\Sigma}_{\Gamma_0} \\
\mathrm{AAr}\,(\lambda x : A.M)^{\Sigma}_{\Gamma} & \simeq & (\mathrm{CAr}\,(A)^{\Sigma}_{\Gamma})\hat{}\,\mathrm{AAr}\,(M)^{\Sigma}_{\Gamma,x:A} \\
\mathrm{AAr}\,(MN)^{\Sigma}_{\Gamma} & \simeq & \mathrm{AAr}\,(M)^{\Sigma}_{\Gamma} \text{ with its first entry removed}
\end{array}$$

We assume that:

- If a declaration $a : K$ is made, and K has content arity α, then a is an α-ary constant of F_{ELF}.
- If a declaration $c : A$ is made, and A has content arity α, then c is an α-ary constant of F_{ELF}.
- If a declaration $x : A$ is made, and A has content arity α, then x is an α-ary variable of F_{ELF}.

Define the translation Φ from ELF to F_{ELF} as follows. We define:

- For kinds K of content arity α, the α-ary kind $[\![K]\!]$ and the α-ary superkind $\langle\!|K|\!\rangle$ of F_{ELF}.
- For families A of argument arity α and content arity $\mathbf{0}$, the α-ary term $[\![A]\!]$ of F_{ELF}.

- For families A of argument arity α and content arity β, the α-ary family of β-ary types $\llparenthesis A \rrparenthesis$ of F_{ELF}.
- For objects M of argument arity α, the α-ary term $\llbracket M \rrbracket$ of F_{ELF}.

The definition is as follows:

$$
\begin{array}{rcl}
\llbracket \mathbf{Type} \rrbracket & \equiv & \mathbf{Type} \\
\llbracket \Pi x : A.K \rrbracket & \simeq & (x : \llparenthesis A \rrparenthesis)\, \llbracket K \rrbracket \\
\\
\llparenthesis \mathbf{Type} \rrparenthesis & \equiv & \mathbf{Kind} \\
\llparenthesis \Pi x : A.K \rrparenthesis & \simeq & (x : \llparenthesis A \rrparenthesis)\, \llparenthesis K \rrparenthesis \\
\\
\llbracket a \rrbracket & \equiv & a \\
\llbracket \lambda x : A.B \rrbracket & \simeq & [x : \llparenthesis A \rrparenthesis]\, \llbracket B \rrbracket \\
\llbracket AM \rrbracket & \simeq & \llbracket A \rrbracket\, [\llbracket M \rrbracket] \\
\\
\llparenthesis A \rrparenthesis & \simeq & \llbracket A \rrbracket \qquad \text{(if } A \text{ is an object-like family of content and argument arity } \mathbf{0}\text{)} \\
\llparenthesis \Pi x : A.B \rrparenthesis & \simeq & (x : \llparenthesis A \rrparenthesis)\, \llparenthesis B \rrparenthesis \\
\llparenthesis \lambda x : A.B \rrparenthesis & \simeq & [x : \llparenthesis A \rrparenthesis]\, \llparenthesis M \rrparenthesis \\
\llparenthesis AM \rrparenthesis & \simeq & \llparenthesis A \rrparenthesis\, [\llbracket M \rrbracket] \\
\\
\llbracket c \rrbracket & \equiv & c \\
\llbracket x \rrbracket & \equiv & x \\
\llbracket \lambda x : A.M \rrbracket & \simeq & [x : \llparenthesis A \rrparenthesis]\, \llbracket M \rrbracket \\
\llbracket MN \rrbracket & \simeq & \llbracket M \rrbracket\, [\llbracket N \rrbracket]
\end{array}
$$

We now map each context Γ of ELF to a context $\llbracket \Gamma \rrbracket$ of F_{ELF} thus:

$$
\begin{array}{rcl}
\llbracket \langle\rangle \rrbracket & \equiv & \langle\rangle \\
\llbracket \Gamma, x : A \rrbracket & \simeq & \llbracket \Gamma \rrbracket, x : \llparenthesis A \rrparenthesis
\end{array}
$$

and each signature Σ of ELF to a signature $\llbracket \Sigma \rrbracket$ of F_{ELF} thus:

$$
\begin{array}{rcl}
\llbracket \langle\rangle \rrbracket & \equiv & \langle\rangle \\
\llbracket \Sigma, a : K \rrbracket & \equiv & \llbracket \Sigma \rrbracket, a : \llbracket K \rrbracket \\
\llbracket \Sigma, c : A \rrbracket & \equiv & \llbracket \Sigma \rrbracket, c : \llbracket A \rrbracket
\end{array}
$$

The translation from F_{ELF} to ELF is defined as follows. We define, relative to a given signature Σ:

- For terms M, every constant in which is delared in Σ, the expression $(\!|M|\!)$, which is either an object or a family of ELF.
- For families of types F, every constant in which is delared in Σ, a family $(\!|F|\!)$ of ELF.
- For large kinds K, every constant in which is delared in Σ, the kind $(\!|K|\!)$ of ELF.
- For superkinds S, every constant in which is delared in Σ,the kind $(\!|S|\!)$ of ELF.

We assume that, when $c^\alpha : K^\alpha$ occurs in an F_{ELF} signature, c^α is an object-level constant of ELF if K^α is a type, and a family-level constant of ELF if K^α is a large kind.

The definition is as follows:

$$\begin{array}{rcl}
(\!|x|\!) & \equiv & x \\
(\!|c|\!) & \equiv & c \\
(\!|[x:A]M|\!) & \equiv & \lambda x : (\!|A|\!) \,.\, (\!|M|\!)\,\text{, which is an object iff } (\!|M|\!) \text{ is an object} \\
(\!|M[N]|\!) & \simeq & (\!|M|\!)\,(\!|N|\!)\,\text{, which is defined iff } (\!|N|\!) \text{ is an object,} \\
 & & \quad \text{in which case it is an object iff } (\!|M|\!) \text{ is an object}
\end{array}$$

$$\begin{array}{rcl}
(\!|(x:A)B|\!) & \equiv & \Pi x : (\!|A|\!) \,.\, (\!|B|\!) \\
(\!|[x:A]F|\!) & \equiv & \lambda x : (\!|A|\!) \,.\, (\!|F|\!) \\
(\!|F[M]|\!) & \equiv & (\!|F|\!)\,(\!|M|\!)\,\text{, which is defined iff } (\!|M|\!) \text{ is an object}
\end{array}$$

$$\begin{array}{rcl}
(\!|\mathbf{Type}|\!) & \equiv & \mathbf{Type} \\
(\!|(x:A)K|\!) & \equiv & \Pi x : (\!|A|\!) \,.\, (\!|K|\!)
\end{array}$$

$$\begin{array}{rcl}
(\!|\mathbf{Kind}|\!) & \equiv & \mathbf{Type} \\
(\!|(x:A)S|\!) & \equiv & \Pi x : (\!|A|\!) \,.\, (\!|S|\!)
\end{array}$$

We now define, for each context Γ of F_{ELF}, the context $(\!|\Gamma|\!)$ of ELF thus:

$$\begin{array}{rcl}
(\!|\langle\rangle|\!) & \equiv & \langle\rangle \\
(\!|\Gamma, x:A|\!) & \simeq & (\!|\Gamma|\!)\,, x : (\!|A|\!)
\end{array}$$

and, for each signature Σ of F_{ELF}, the signature $(\!|\Sigma|\!)$ of ELF thus:

$$\begin{array}{rcl}
(\!|\langle\rangle|\!) & \equiv & \langle\rangle \\
(\!|\Sigma, c:K|\!) & \simeq & (\!|\Sigma|\!)\,, c : (\!|K|\!)
\end{array}$$

It is easy to show that this latter translation 'preserves arity' in the following sense:

Lemma 8.1 *1. For every α-ary term M of F_{ELF}, if $(\!|M|\!)$ is defined, then either $(\!|M|\!)$ is an object of argument arity α, or $(\!|M|\!)$ is an object-like family of argument arity α and* *content arity* $\mathbf{0}$.

2. *For every* α*-ary family of* β*-ary types* F *of* F_{ELF}*, if* $(\!|F|\!)$ *is defined, then* $(\!|F|\!)$ *is a family of argument arity* α *and content arity* β.

3. *For every* α*-ary large kind* K *of* F_{ELF}*, if* $(\!|K|\!)$ *is defined, then* $(\!|K|\!)$ *is a kind of content arity* α.

4. *For every* α*-ary superkind* S *of* F_{ELF}*, if* $(\!|S|\!)$ *is defined, then* $(\!|S|\!)$ *is a kind of content arity* α.

Proof The four parts are proved simultaneously by induction on the F_{ELF} expression. No case poses any difficulty. $\square$

These translations are inverses to one another, as is proven in the following two theorems:

Theorem 8.2 *1. For every object* M *of ELF, if* $[\![M]\!]$ *is defined, then* $(\!|[\![M]\!]|\!)$ *is defined and is an object, and*

$$(\!|[\![M]\!]|\!) \equiv M$$

2. *For every family* A *of ELF, if* $[\![A]\!]$ *is defined, then* $(\!|[\![A]\!]|\!)$ *is defined and is a family, and*

$$(\!|[\![A]\!]|\!) \equiv A$$

3. *For every family* A *of ELF, if* $\langle\!|A|\!\rangle$ *is defined, then* $(\!|\langle\!|A|\!\rangle|\!)$ *is defined, and*

$$(\!|\langle\!|A|\!\rangle|\!) \equiv A$$

4. *For every kind* K *of ELF, if* $[\![K]\!]$ *is defined, then* $(\!|[\![K]\!]|\!)$ *is defined, and*

$$(\!|[\![K]\!]|\!) \equiv K$$

5. *For every kind* K *of ELF, if* $\langle\!|K|\!\rangle$ *is defined, then* $(\!|\langle\!|K|\!\rangle|\!)$ *is defined, and*

$$(\!|\langle\!|K|\!\rangle|\!) \equiv K$$

Proof The five parts are proven simultaneously by induction on the ELF expression. No case poses much difficulty; we give the details of two here.

- Suppose $[\![\lambda x : A.M]\!]$ is defined. Then

$$[\![\lambda x : A.M]\!] \equiv [x : \langle\!|A|\!\rangle]\,[\![M]\!]$$

By the induction hypothesis,

$$\begin{array}{rcl} (\!|\langle\!|A|\!\rangle|\!) & \equiv & A \\ (\!|[\![M]\!]|\!) & \equiv & M \\ \therefore (\!|[x:\langle\!|A|\!\rangle]\,[\![M]\!]|\!) & \equiv & \lambda x : (\!|\langle\!|A|\!\rangle|\!) \,.\, (\!|[\![M]\!]|\!) \\ & \equiv & \lambda x : A.M \end{array}$$

- Suppose A is an object-like family of content and argument arity $\mathbf{0}$, and $\langle\!|A|\!\rangle$ is defined; i.e. $[\![A]\!]$ is defined. By the induction hypothesis, $(\!|[\![A]\!]|\!)$ is defined, is a family, and

$$(\!|[\![A]\!]|\!) \equiv A$$

as required.

□

Theorem 8.3 *1. For every term M of F_{ELF}, if $(\!|M|\!)$ is defined, then $[\![(\!|M|\!)]\!]$ is defined, and*

$$[\![(\!|M|\!)]\!] \equiv M$$

2. For every family F of F_{ELF}, if $(\!|F|\!)$ is defined and is a family, then $\langle\!|(\!|F|\!)|\!\rangle$ is defined, and

$$\langle\!|(\!|F|\!)|\!\rangle \equiv F$$

3. For every large kind K of F_{ELF}, if $(\!|K|\!)$ is defined, then $[\![(\!|K|\!)]\!]$ is defined, and

$$\langle\!|(\!|K|\!)|\!\rangle \equiv K$$

4. For every superkind S of F_{ELF}, if $(\!|S|\!)$ is defined, then $\langle\!|(\!|S|\!)|\!\rangle$ is defined, and

$$\langle\!|(\!|S|\!)|\!\rangle \equiv S$$

Proof Again, the proof is by simultaneous induction on the F_{ELF} expressions. We give the details of two cases:

- Suppose $(\!|[x:A]M|\!)$ is defined. Then

$$(\!|[x:A]M|\!) \equiv \lambda x : (\!|A|\!) \,.\, (\!|M|\!)$$

$(\!|A|\!)$ must therefore be a family; $(\!|M|\!)$ may be an object or a family. In either case, the induction hypothesis gives us

$$\begin{array}{rcl} [\![(\!|A|\!)]\!] & \equiv & A \\ [\![(\!|M|\!)]\!] & \equiv & M \\ \therefore [\![\lambda x : (\!|A|\!) \,.\, (\!|M|\!)]\!] & \equiv & \lambda x : A.M \end{array}$$

- We consider the case 2 in which F is a $\mathbf{0}$-ary term being considered as a $\mathbf{0}$-ary type. Suppose A is a term, and $(\!|A|\!)$ is defined and is a family. By Lemma 8.1, $(\!|A|\!)$ is a family of content and argument arity $\mathbf{0}$. By the induction hypothesis,

$$[\![(\!|A|\!)]\!] \equiv A$$

Therefore,

$$\begin{aligned} \langle\!|(\!|A|\!)|\!\rangle &\equiv [\![(\!|A|\!)]\!] \\ &\equiv A \end{aligned}$$

$\Box$

Corollary 8.4 *1. For every context Γ of ELF, if $[\![\Gamma]\!]$ is defined, then $(\!|[\![\Gamma]\!]|\!)$ is defined, and*

$$(\!|[\![\Gamma]\!]|\!) \equiv \Gamma$$

2. For every signature Σ of ELF, if $[\![\Sigma]\!]$ is defined, then $(\!|[\![\Sigma]\!]|\!)$ is defined, and

$$(\!|[\![\Sigma]\!]|\!) \equiv \Sigma$$

3. For every context Γ of F_{ELF}, if $(\!|\Gamma|\!)$ is defined, then $[\![(\!|\Gamma|\!)]\!]$ is defined, and

$$[\![(\!|\Gamma|\!)]\!] \equiv \Gamma$$

4. For every signature Σ of F_{ELF}, if $(\!|\Sigma|\!)$ is defined, then $[\![(\!|\Sigma|\!)]\!]$ is defined, and

$$[\![(\!|\Sigma|\!)]\!] \equiv \Sigma$$

Proof A simple induction on Γ or Σ in each case. $\Box$

The following two theorems therefore comprise the proof of the equivalence of ELF and F_{ELF}:

Theorem 8.5 *1. If Σ sig is derivable in ELF, then $[\![\Sigma]\!]$ sig is derivable in F_{ELF}.*

2. If $\Gamma \vdash_\Sigma \cdot$ is derivable in ELF, then $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]}$ valid is derivable in F_{ELF}.

3. If $\Gamma \vdash_\Sigma K$ is derivable in ELF, then $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![K]\!]$ kind and $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \langle\!|K|\!\rangle$ skind.

4. If $\Gamma \vdash_\Sigma K_1 = K_2$ is derivable in ELF, then $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![K_1]\!] = [\![K_2]\!]$ and $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \langle\!|K_1|\!\rangle = \langle\!|K_2|\!\rangle$ are derivable in F_{ELF}.

5. If $\Gamma \vdash_\Sigma A : K$ and A has content arity $\mathbf{0}$, then $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![A]\!] : [\![K]\!]$.

6. If $\Gamma \vdash_\Sigma A = B : K$ and either A or B has content arity $\mathbf{0}$, then A and B both have content arity $\mathbf{0}$, and $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![A]\!] = [\![B]\!] : [\![K]\!]$.

7. *If* $\Gamma \vdash_\Sigma A : K$*, then* $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \langle\!| A |\!\rangle : \langle\!| K |\!\rangle$.

8. *If* $\Gamma \vdash_\Sigma A = B : K$*, then* $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \langle\!| A |\!\rangle = \langle\!| B |\!\rangle : \langle\!| K |\!\rangle$.

9. *If* $\Gamma \vdash_\Sigma M : A$*, then* $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![M]\!] : \langle\!| A |\!\rangle$.

10. *If* $\Gamma \vdash_\Sigma M = N : A$*, then* $[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![M]\!] = [\![N]\!] : \langle\!| A |\!\rangle$.

Proof The proof is by induction on the derivation in ELF. All cases are straightforward. □

Theorem 8.6 1. *If* Σ sig *is derivable in* F_{ELF}*, then* $(\!|\Sigma|\!)$ sig *is derivable in ELF.*

2. *If* $\Gamma \vdash_\Sigma$ valid *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \cdot$ *is derivable in ELF.*

3. *If* $\Gamma \vdash_\Sigma K$ kind *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|K|\!)$ *is derivable in ELF.*

4. *If* $\Gamma \vdash_\Sigma K_1 = K_2$ *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|K_1|\!) = (\!|K_2|\!)$.

5. *If* $\Gamma \vdash_\Sigma S$ skind *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|S|\!)$ *is derivable in ELF.*

6. *If* $\Gamma \vdash_\Sigma S_1 = S_2$ *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|S_1|\!) = (\!|S_2|\!)$.

7. *If* $\Gamma \vdash_\Sigma M : K$ *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|M|\!) : (\!|K|\!)$ *is derivable in ELF.*

8. *If* $\Gamma \vdash_\Sigma M = N : K$ *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|M|\!) = (\!|N|\!) : (\!|K|\!)$ *is derivable in ELF.*

9. *If* $\Gamma \vdash_\Sigma F : S$ *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|F|\!) : (\!|S|\!)$ *is derivable in ELF.*

10. *If* $\Gamma \vdash_\Sigma F_1 = F_2 : S$ *is derivable in* F_{ELF}*, then* $(\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|F_1|\!) = (\!|F_2|\!) : (\!|S|\!)$ *is derivable in ELF.*

Proof The proof is by induction on the derivation in F_{ELF}. Again, all cases are straightforward. □

8.1.2 Discussion — The Reduced Edinburgh Logical Framework

If we remove the feature **FKnd**, the framework we obtain is isomorphic to a variant of the Edinburgh Logical Framework, in which we do not allow lambda-abstraction at the level of families. We may call this theory the *Reduced Edinburgh Logical Framework*, ELF$^-$. ELF$^-$ is obtained from ELF by removing the clause

$$\text{Family } A ::= \cdots \mid \lambda x : A.A \mid \cdots$$

from the grammar, and the rule of deduction ($\mathtt{B-ABS-FAM}$).

That ELF is a conservative extension of ELF$^-$ was first observed by Geuvers and Barenden in [GB99]. This result can be regarded as one of the conservative inclusions of our hierachy. It is not quite the inclusion of $\mathbf{BF} + \mathbf{S}\lambda^\beta(\omega)$ in $\mathbf{BF} + \mathbf{S}\lambda^\beta(\omega) + \mathbf{FKnd}$; rather, we must use a feature $\mathbf{S}\lambda_0^\beta(\omega)$, which permits the formation of term $[x : A]M$ only in types, never in large kinds. Its definition is obtained from that of $\mathbf{S}\lambda^{\beta}(\omega)$ by removing the constructor for kinds $(x : A)K$, but retaining the constructor for types $(x : A)B$, and restricting the rules of deduction accordingly. The conservativity result of $\mathbf{S}\lambda^{\beta}(\omega)$ over $\mathbf{S}\lambda_0^{\beta}(\omega)$ follows the same pattern as that of $F + \mathbf{X}\lambda^{R}(n)$ over F.

Theorem 8.7 *ELF⁻ is isomorphic to*

$$F_{ELF^-} = \mathbf{BF} + \mathbf{SPar}(\omega) + \mathbf{S}\lambda_0^{\beta}(\omega)$$

Proof The restrictions of the translations given above form an isomorphism between ELF$^-$ and F_{ELF^-}. □

Corollary 8.8 *ELF is a conservative extension of ELF$^-$.*

Proof Suppose J is a judgement of ELF$^-$ derivable in ELF. Then $\Psi[\![J]\!]$ is a judgement of F_{ELF^-} derivable in F_{ELF}. As F_{ELF} is conservative over F_{ELF^-}, $\Psi[\![J]\!]$ is derivable in F_{ELF^-}; hence J is derivable in ELF$^-$. □

8.2 Martin-Löf's Logical Framework

While it is not one of the systems we dealt with in Chapter 8, we can incorporate Martin-Löf's Logical Framework into our hierarchy. We deal here with the typed version of Martin-Löf's LF, as presented in [Luo94].

Grammar

Term	k	$\in$	$\mathcal{M}$	$::=$	$x \mid c \mid [x : K]k \mid k(k)$
Kind	K	$\in$	$\mathcal{K}$	$::=$	$\mathbf{Type} \mid \mathrm{El}(k) \mid (x : K)K$
Signature Declaration	γ	$\in$	$\mathcal{C}$	$::=$	$c : K$ of $c \mid (x : K, \ldots, x : K)k = k : K$ of none
Context Declaration	δ	$\in$	$\mathcal{D}$	$::=$	$x : K$ of x
Judgement	J	$\in$	$\mathcal{J}$	$::=$	valid $\mid K$ kind $\mid k : K \mid k = k : K \mid K = K$

Reduction Define $\beta\eta$-conversion on the set of terms via the contractions:

$$\begin{array}{rcll} ([x : K]k)(k') & \rightsquigarrow & [k'/x]k & \\ [x : K]k(x) & \rightsquigarrow & k & (x \notin FV(k)) \end{array}$$

Rules of Deduction

$$\frac{}{\langle\rangle \text{ sig}} \qquad \frac{\vdash_\Sigma K \text{ kind}}{\Sigma, c : K \text{ sig}}\,(c \notin \operatorname{dom}\Sigma) \qquad \frac{\Gamma \vdash_\Sigma k : K \quad \Gamma \vdash_\Sigma k' : K}{\Sigma, (\Gamma)k = k' : K \text{ sig}}$$

$$\frac{\Sigma \text{ sig}}{\vdash_\Sigma \text{ valid}} \qquad \frac{\Gamma \vdash_\Sigma K \text{ kind}}{\Gamma, x : K \vdash_\Sigma \text{ valid}}\,(x \notin \operatorname{dom}\Gamma)$$

$$\frac{\Gamma \vdash_\Sigma \text{ valid}}{\Gamma \vdash_\Sigma x : K}\,(x : K \in \Gamma) \qquad \frac{\Gamma \vdash_\Sigma \text{ valid}}{\Gamma \vdash_\Sigma c : K}\,(c : K \in \Sigma)$$

$$\frac{\Gamma \Vdash_\Sigma \vec{k} :: \Delta}{\Gamma \vdash_\Sigma [\vec{k}/\operatorname{dom}\Delta]k = [\vec{k}/\operatorname{dom}\Delta]k' : [\vec{k}/\operatorname{dom}\Delta]K}\,((\Delta)k = k' : K)$$

$$\frac{\Gamma \vdash_\Sigma K \text{ kind}}{\Gamma \vdash_\Sigma K = K} \qquad \frac{\Gamma \vdash_\Sigma K_1 = K_2}{\Gamma \vdash_\Sigma K_2 = K_1} \qquad \frac{\Gamma \vdash_\Sigma K_1 = K_2 \quad \Gamma \vdash_\Sigma K_2 = K_3}{\Gamma \vdash_\Sigma K_1 = K_3}$$

$$\frac{\Gamma \vdash_\Sigma k : K}{\Gamma \vdash_\Sigma k = k : K} \qquad \frac{\Gamma \vdash_\Sigma k_1 = k_2 : K}{\Gamma \vdash_\Sigma k_2 = k_1 : K} \qquad \frac{\Gamma \vdash_\Sigma k_1 = k_2 : K \quad \Gamma \vdash_\Sigma k_2 = k_3 : K}{\Gamma \vdash_\Sigma k_1 = k_3 : K}$$

$$\frac{\Gamma \vdash_\Sigma k : K \quad \Gamma \vdash_\Sigma K = K'}{\Gamma \vdash_\Sigma k : K'} \qquad \frac{\Gamma \vdash_\Sigma k = k' : K \quad \Gamma \vdash_\Sigma K = K'}{\Gamma \vdash_\Sigma k = k' : K'}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash_\Sigma \text{ valid} \quad \Gamma \vdash_\Sigma k : K}{\Gamma, [k/x]\Gamma' \vdash_\Sigma \text{ valid}}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash_\Sigma K' \text{ kind} \quad \Gamma \vdash_\Sigma k : K}{\Gamma, [k/x]\Gamma' \vdash_\Sigma [k/x]K' \text{ kind}} \qquad \frac{\Gamma, x : K, \Gamma' \vdash_\Sigma K' \text{ kind} \quad \Gamma \vdash_\Sigma k = k' : K}{\Gamma, [k/x]\Gamma' \vdash_\Sigma [k/x]K' = [k'/x]K'}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash_\Sigma k' : K' \quad \Gamma \vdash_\Sigma k : K}{\Gamma, [k/x]\Gamma' \vdash_\Sigma [k/x]k' : [k/x]K'} \qquad \frac{\Gamma, x : K, \Gamma' \vdash_\Sigma k' : K' \quad \Gamma \vdash_\Sigma k_1 = k_2 : K}{\Gamma, [k_1/x]\Gamma' \vdash_\Sigma [k_1/x]k' = [k_2/x]k' : [k_1/x]K'}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash_\Sigma K' = K'' \quad \Gamma \vdash_\Sigma k : K}{\Gamma, [k/x]\Gamma' \vdash_\Sigma [k/x]K' = [k/x]K''} \qquad \frac{\Gamma, x : K, \Gamma' \vdash_\Sigma k' = k'' : K' \quad \Gamma \vdash_\Sigma k : K}{\Gamma, [k/x]\Gamma' \vdash_\Sigma [k/x]k' = [k/x]k'' : [k/x]K'}$$

$$\frac{\Gamma \vdash_\Sigma K \text{ kind} \quad \Gamma, x : K \vdash_\Sigma K' \text{ kind}}{\Gamma \vdash_\Sigma (x : K)K' \text{ kind}} \qquad \frac{\Gamma \vdash_\Sigma K_1 = K_2 \quad \Gamma, x : K_1 \vdash_\Sigma K_1' = K_2'}{\Gamma \vdash_\Sigma (x : K_1)K_1' = (x : K_2)K_2'}$$

$$\frac{\Gamma, x : K \vdash_\Sigma k : K'}{\Gamma \vdash_\Sigma [x : K]k : (x : K)K'} \qquad \frac{\Gamma \vdash_\Sigma K_1 = K_2 \quad \Gamma, x : K_1 \vdash_\Sigma k_1 = k_2 : K}{\Gamma \vdash_\Sigma [x : K_1]k_1 = [x : K_2]k_2 : (x : K_1)K}$$

$$\frac{\Gamma \vdash_\Sigma f : (x : K)K' \quad \Gamma \vdash_\Sigma k : K}{\Gamma \vdash_\Sigma f(k) : [k/x]K'} \qquad \frac{\Gamma \vdash_\Sigma f = f' : (x : K)K' \quad \Gamma \vdash_\Sigma k_1 = k_2 : K}{\Gamma \vdash_\Sigma f(k_1) = f'(k_2) : [k_1/x]K'}$$

$$\frac{\Gamma, x : K \vdash_\Sigma k' : K' \quad \Gamma \vdash_\Sigma k : K}{\Gamma \vdash_\Sigma ([x : K]k')(k) = [k/x]k' : [k/x]K'} \qquad \frac{\Gamma \vdash_\Sigma f : (x : K)K'}{\Gamma \vdash_\Sigma [x : K]f(x) = f : (x : K)K'} \; (x \notin FV(f))$$

$$\frac{\Gamma \vdash_\Sigma \text{valid}}{\Gamma \vdash_\Sigma \mathbf{Type} \text{ kind}} \qquad \frac{\Gamma \vdash_\Sigma k : \mathbf{Type}}{\Gamma \vdash_\Sigma \mathrm{El}\,(k) \text{ kind}}$$

We claim that LF is equivalent to the logical framework

$$F_{LF} = \mathbf{BF} + \mathbf{LPar}\,(\omega) + \mathbf{L}\lambda^{\beta\eta}\,(\omega) + \mathbf{LEq}\,(\omega, \omega)$$

We first assign *arities* to the terms and kinds of LF as follows. For the kinds:

$$\begin{aligned} \mathrm{Ar}\,(\mathbf{Type}) &\equiv \mathbf{0} \\ \mathrm{Ar}\,(\mathrm{El}\,(k)) &\equiv \mathbf{0} \\ \mathrm{Ar}\,((x : K_1)K_2) &\equiv (\mathrm{Ar}\,(K_1))\hat{\ }\mathrm{Ar}\,(K_2) \end{aligned}$$

Recall that we have assumed that LF and F_{LF} use the same sets of constants and variables. We now make the further assumption that:

- Whenever the declaration $c : K$ is made in a signature in LF, if $\mathrm{Ar}\,(K) \equiv \alpha$, then c is an α-ary constant of F_{LF}.
- Whenever the declaration $x : K$ is made in a context in LF, if $\mathrm{Ar}\,(K) \equiv \alpha$, then x is an α-ary variable of F_{LF}.

We assign arities to the terms of LF thus:

- If x is an α-ary variable of F_{LF}, $\mathrm{Ar}\,(x) \equiv \alpha$.
- If c is an α-ary constant of F_{LF}, $\mathrm{Ar}\,(c) \equiv \alpha$.
- $\mathrm{Ar}\,([x : K]k) \equiv (\mathrm{Ar}\,(K))\hat{\ }\mathrm{Ar}\,(k)$

- If k_1 has arity $\mathbf{0}$, $\mathrm{Ar}\,(k_1(k_2))$ is undefined. Otherwise, if $\mathrm{Ar}\,(k_1) \equiv (\alpha_1, \ldots, \alpha_n)$, then

$$\mathrm{Ar}\,(k_1(k_2)) \equiv (\alpha_2, \ldots, \alpha_n)$$

Once we have this method for assigning arities to expressions, the task is more or less complete; the translations are almost trivial. We define, for each F_{LF} term M the LF term $[\![M]\!]$ as follows:

$$\begin{array}{rcl} [\![x]\!] & \equiv & x \\ [\![c]\!] & \equiv & c \\ [\![[x:K]M]\!] & \equiv & [x:[\![K]\!]]\,[\![M]\!] \\ [\![M[N]]\!] & \equiv & [\![M]\!]\,([\![N]\!]) \end{array}$$

We define simultaneously, for each F_{LF} kind K, the LF kind $[\![K]\!]$ thus:

$$\begin{array}{rcl} [\![M]\!] & \equiv & \mathrm{El}\,(M) \\ [\![\mathbf{Type}]\!] & \equiv & \mathbf{Type} \\ [\![(x:K_1)K_2]\!] & \equiv & (x:[\![K_1]\!])\,[\![K_2]\!] \end{array}$$

We define, for each F_{LF} context Γ, the LF context $[\![\Gamma]\!]$ thus:

$$\begin{array}{rcl} [\![\langle\rangle]\!] & \equiv & \langle\rangle \\ [\![\Gamma, x:K]\!] & \equiv & [\![\Gamma]\!]\,, x:[\![K]\!] \end{array}$$

We define, for each F_{LF} signature Σ, the LF signature $[\![\Sigma]\!]$ thus:

$$\begin{array}{rcl} [\![\langle\rangle]\!] & \equiv & \langle\rangle \\ [\![\Sigma, c:K]\!] & \equiv & [\![\Sigma]\!]\,, c:[\![K]\!] \\ [\![\Sigma, (\Gamma)(M=N:K)]\!] & \equiv & [\![\Sigma]\!]\,, ([\![\Gamma]\!])([\![M]\!] = [\![N]\!] : [\![K]\!]) \end{array}$$

Finally, we define, for each F_{LF} judgement J, the LF judgement $[\![J]\!]$ thus:

$$\begin{array}{rcl} [\![\Sigma \text{ sig}]\!] & \equiv & [\![\Sigma]\!] \text{ sig} \\ [\![\Gamma \vdash_\Sigma \text{valid}]\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} \text{valid} \\ [\![\Gamma \vdash_\Sigma M:K]\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![M]\!] : [\![K]\!] \\ [\![\Gamma \vdash_\Sigma M=N:K]\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![M]\!] = [\![N]\!] : [\![K]\!] \\ [\![\Gamma \vdash_\Sigma K \text{ kind}]\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![K]\!] \text{ kind} \\ [\![\Gamma \vdash_\Sigma K = K']\!] & \equiv & [\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![K]\!] = [\![K']\!] \end{array}$$

The translation in the opposite direction must be defined as a *partial* function. (For example, unless $\mathrm{Ar}\,(k_2)$ matches the first entry of $\mathrm{Ar}\,(k_1)$, $(\!|k_1(k_2)|\!)$ shall be undefined.) We define, for some terms k of LF such that $\mathrm{Ar}\,(k) \equiv \alpha$, the α-ary term $(\!|k|\!)$ of F_{LF} thus:

$$\begin{array}{rcl} (\!|x|\!) & \equiv & x \\ (\!|c|\!) & \equiv & c \\ (\!|[x:K]k|\!) & \simeq & [x:(\!|K|\!)]\,(\!|k|\!) \\ (\!|k(k')|\!) & \simeq & (\!|k|\!)\,[(\!|k'|\!)] \end{array}$$

We define simultaneously, for some kinds K of LF such that $\mathrm{Ar}\,(K) \equiv \alpha$, the α-ary kind $(\!|K|\!)$ of F_{LF} thus:

$$\begin{array}{rcl} (\!|\mathrm{El}\,(k)|\!) & \simeq & (\!|k|\!) \\ (\!|\mathbf{Type}|\!) & \equiv & \mathbf{Type} \\ (\!|(x:K)K'|\!) & \simeq & (x:(\!|K|\!))\,(\!|K'|\!) \end{array}$$

We define, for some contexts Γ, the context $(\!|\Gamma|\!)$ of F_{LF} thus:

$$\begin{array}{rcl} (\!|\langle\rangle|\!) & \equiv & \langle\rangle \\ (\!|\Gamma, x:K|\!) & \simeq & (\!|\Gamma|\!)\,, x:(\!|K|\!) \end{array}$$

We define, for some signatures Σ, the signature $(\!|\Sigma|\!)$ of F_{LF} thus:

$$\begin{array}{rcl} (\!|\langle\rangle|\!) & \equiv & \langle\rangle \\ (\!|\Sigma, c:K|\!) & \simeq & (\!|\Sigma|\!)\,, c:(\!|K|\!) \\ & & (\!|\Sigma, (x_1:K_1, \ldots, x_n:K_n)(k = k':K)|\!) \\ & & \simeq \quad (\!|\Sigma|\!)\,, (x_1:(\!|K_1|\!)\,, \ldots, x_n:(\!|K_n|\!))((\!|k|\!) = (\!|k'|\!) : (\!|K|\!)) \end{array}$$

And we define, for some judgements J of LF, the judgement $(\!|J|\!)$ of F_{LF} thus:

$$\begin{array}{rcl} (\!|\Sigma \text{ sig}|\!) & \simeq & (\!|\Sigma|\!) \text{ sig} \\ (\!|\Gamma \vdash_\Sigma \text{valid}|\!) & \simeq & (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} \text{valid} \\ (\!|\Gamma \vdash_\Sigma k : K|\!) & \simeq & (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|k|\!) : (\!|K|\!) \\ (\!|\Gamma \vdash_\Sigma k = k' : K|\!) & \simeq & (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|k|\!) = (\!|k'|\!) : (\!|K|\!) \\ (\!|\Gamma \vdash_\Sigma K \text{ kind}|\!) & \simeq & (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|K|\!) \text{ kind} \\ (\!|\Gamma \vdash_\Sigma K = K'|\!) & \simeq & (\!|\Gamma|\!) \vdash_{(\!|\Sigma|\!)} (\!|K|\!) = (\!|K'|\!) \end{array}$$

It is simple to prove that these two translations are inverses to one another:

Theorem 8.9 *1. For any term k of LF, if $(\!|k|\!)$ is defined, then*

$$[\![(\!|k|\!)]\!] \equiv k$$

Similarly for kinds, contexts, signatures and judgements.

2. For any α-ary term M of F_{LF},

$$\mathrm{Ar}\,(\llbracket M \rrbracket) \equiv \alpha\,,$$

$(\!|\llbracket M \rrbracket|\!)$ is defined, and

$$(\!|\llbracket M \rrbracket|\!) \equiv M\,.$$

Similarly for kinds, contexts, signatures and judgements.

Proof

1. A simple induction on k. No case is at all difficult; we give one as an illustration:

$$\begin{aligned} \llbracket (\!|[x:K]k|\!) \rrbracket &\equiv \llbracket [x:(\!|K|\!)]\,(\!|k|\!) \rrbracket \\ &\equiv [x:\llbracket (\!|K|\!) \rrbracket]\,\llbracket (\!|k|\!) \rrbracket \\ &\equiv [x:K]k \qquad \text{(i.h.)} \end{aligned}$$

2. A simple induction on M. Again, no case is difficult. As an example, the case of an application is as follows:

   Consider the β-ary term $M^{(\alpha)\hat{}\beta}[N^\alpha]$. By i.h.,

$$\begin{aligned} \mathrm{Ar}\,(\llbracket M \rrbracket) &\equiv (\alpha)\hat{}\beta & \qquad \mathrm{Ar}\,(\llbracket N \rrbracket) &\equiv \alpha \\ (\!|\llbracket M \rrbracket|\!) &\equiv M & (\!|\llbracket N \rrbracket|\!) &\equiv N \end{aligned}$$

   Therefore, $(\!|\llbracket M \rrbracket\,(\llbracket N \rrbracket)|\!)$ is defined, and

$$(\!|\llbracket M \rrbracket\,(\llbracket N \rrbracket)|\!) \equiv M[N]$$

□

We shall also need the fact that the two translations preserve substitution:

Lemma 8.10 *1. For any α-ary variable x, α-ary term M, and β-ary term N of F_{LF},*

$$\llbracket [M/x]N \rrbracket \equiv [\llbracket M \rrbracket/x]\,\llbracket N \rrbracket$$

Similarly for N replaced with a kind.

2. For any α-ary variable x and terms k, k' of LF such that $\mathrm{Ar}\,(k) \equiv x$, if $(\!|k|\!)$ and $(\!|k'|\!)$ are defined, then $(\!|[k/x]k'|\!)$ is defined and

$$(\!|[k/x]k'|\!) \equiv [(\!|k|\!)/x]\,(\!|k'|\!)$$

Similarly for k' replaced with a kind.

Proof A simple induction on N or k'. □

The following two theorems complete the proof of the equivalence of the two systems:

Theorem 8.11 *If J is derivable in F_{LF}, then $\llbracket J \rrbracket$ is derivable in LF.*

Proof Induction on the derivation in J, considering only the rules that are not redundant. No case is too difficult, so, again, we only give the details of one as an example.

$$\frac{\Gamma \vdash_\Sigma M : (x : K)K' \quad \Gamma \vdash_\Sigma N : K}{\Gamma \vdash_\Sigma M[N] : [N/x]K'}$$

By the induction hypothesis,

$$\begin{array}{lll} [\![\Gamma]\!] & \vdash_{[\![\Sigma]\!]} & [\![M]\!] : (x : [\![K]\!])\,[\![K']\!] \\ [\![\Gamma]\!] & \vdash_{[\![\Sigma]\!]} & [\![N]\!] : [\![K]\!] \end{array}$$

Therefore, by the application rule in LF,

$$[\![\Gamma]\!] \vdash_{[\![\Sigma]\!]} [\![M]\!]\,([\![N]\!]) : [[\![N]\!]\,/x]\,[\![K']\!]$$

By the previous lemma, this is the desired conclusion. $\square$

Theorem 8.12 *If J is derivable in LF, then $(\!|J|\!)$ is defined and is derivable in F_{LF}.*

Proof Induction on the derivation in F_{LF}. Again, no case is at all difficult. $\square$

Just as in Chapter 6, it follows immediately that the translation of the signatures for Martin-Löf Type Theory and the Calculus of Constructions under $(\!|\ |\!)$ gives adequate representations of these two systems in LF.

Chapter 9

Conclusion

We have now examined in some detail an infinite family of logical frameworks, and their relationships to one another and to several existing frameworks, from two directions, which may be described as 'top-down' and 'bottom-up'. In Part I, we presented the frameworks as completed systems; in Part II, we built the systems up brick-by-brick, ensuring that each enlargement of the framework was a conservative extension of what had gone before.

This approach in Part II is not quite as modular as we might wish. Consider the situation where we take an arbitrary framework L in the hierarchy, to which we wish to add the feature F. We often proved results by induction over the rules of deduction of L, or of $L + F$. This is only possible because we know the set of features from which L must be composed: there is no way we can perform induction over any rule of deduction in any feature that we may wish to add to the hierarchy in the future.

The ideal situation would be to create a general theory of typing systems, giving a general definition of a grammar and rules of deduction, and defining precisely what it means to add a new set of constructors and rules to a system. We would then hopefully be able to lay down conditions C_1, C_2, ... that a feature must satisfy, and prove a general result of the form:

> If the feature F satisfies conditions C_1, C_2, ..., and the framework L is composed of features all of which satisfy conditions C_1, C_2, ..., then $L + F$ is a conservative extension of L.

There are a few promising signs in this direction. Many of our metatheoretic results, which we stated as properties of entire systems, can be rephrased as properties of individual rules. For example, the Weakening Lemma:

> If $\Gamma \vdash_\Sigma J$, $\Gamma \subseteq \Delta$, and $\Delta \vdash_\Sigma$ valid.

holds because each rule of deduction either has conclusion of the form $\Gamma \vdash_\Sigma$ valid, or it has the form:

$$\frac{\Gamma, \Theta_1 \vdash_\Sigma J_1 \quad \cdots \quad \Gamma, \Theta_n \vdash_\Sigma J_n}{\Gamma \vdash_\Sigma J}$$

It is easy to see that the Weakening Lemma must hold for any system provided this is true for all of its rules.

When proving the conservativity results in Chapters 7 and 8 of the form "$L + F$ is a conservative extension of L", the only metatheoretic properties we used of L were of this form. Each proof was of one of two forms: either it involved showing that, once a new constructor appeared in a derivation, it could never be removed (which depends only on the form of the new rules); or it involved translating the new terms and types introduced by F into terms and types of L, using only constructors and operations that we knew were in L thanks to the features on which F depends. For example, when defining $\mathrm{APP}(M, \vec{N})$ in the conservativity proof for $\mathbf{X}\lambda^R\,(n)$, we only needed to give the definition for those cases in which M was the application of a new constructor. Thanks to the uniformity of these proofs, it is quite plausible that a general theory of logical frameworks of the form we have been considering could be produced.

9.1 Further Features

In the body of the thesis, we considered features of only a very few kinds. We do not consider it worthwhile going into great detail for a large variety of features; however, we shall show briefly here the shape that features for local and global definitions and derivability-as-primitive frameworks would take.

9.1.1 Local Definitions

A *local definition* is a term or kind of the form

$$\mathrm{let}\, x\, [\Delta] = M \,\mathrm{in}\, K X$$

indicating that, whenever the variable x appears in X, it is to be read as denoting the abstraction $[\Delta]M$. We follow PAL$^+$'s lead in that our mecanism for handling local definitions will be to introduce clauses of the form

$$x[\Delta] = M : K$$

into our contexts; a judgement of the form

$$\Gamma_1, x[\Delta] = M : K, \Gamma_2 \vdash_\Sigma J$$

indicates that x is to be taken to denote $[\Delta]M$ wherever it occurs in Γ_2 or J.

We can again define various versions of this feature depending on what arity context Δ may be, what arity object and kind M and K may be, whether Δ may use large kinds, and whether K may be of this large kind. The feature would then have these new grammatical constructors:

β-ary term	M	$::=$	$\cdots \mid \mathrm{let}\, x\, [\Delta] = N \,\mathrm{in}\, K M$
β-ary type	A	$::=$	$\cdots \mid \mathrm{let}\, x\, [\Delta] = M \,\mathrm{in}\, K A$
β-ary kind	K	$::=$	$\cdots \mid \mathrm{let}\, x\, [\Delta] = M \,\mathrm{in}\, K' K$
Context Declaration	δ	$::=$	$\cdots \mid x[\Delta] = M : K$ of x

identifying similar types and kinds as usual, and these new rules of deduction:

$$\text{(def\_var)}\ \frac{\Gamma, \Delta \vdash_\Sigma M : K}{\Gamma, x[\Delta] = M : K \vdash_\Sigma \text{valid}}\ (x \notin \operatorname{dom}\Gamma)$$

$$\text{(def\_var\_out)}\ \frac{\Gamma \Vdash_\Sigma \overrightarrow{\hat{M}} :: \Delta}{\Gamma \vdash_\Sigma x[\overrightarrow{\hat{M}}] : \{\overrightarrow{\hat{M}}/\Delta\}K}\ (x[\Delta] = M : K \in \Gamma)$$

$$\text{(def\_var\_use)}\ \frac{\Gamma \Vdash_\Sigma \overrightarrow{\hat{M}} :: \Delta}{\Gamma \vdash_\Sigma x[\overrightarrow{\hat{M}}] = \{\overrightarrow{\hat{M}}/\Delta\}M : \{\overrightarrow{\hat{M}}/\Delta\}K}\ (x[\Delta] = M : K \in \Gamma)$$

$$\text{(let)}\ \frac{\Gamma, x[\Delta] = M : K \vdash_\Sigma M : K}{\Gamma \vdash_\Sigma \operatorname{let} x\, [\Delta] = M \operatorname{in} K M : \operatorname{let} x\, [\Delta] = M \operatorname{in} K K}$$

$$\text{(let\_use)}\ \frac{\Gamma, x[\Delta] = M : K \vdash_\Sigma M : K}{\Gamma \vdash_\Sigma \operatorname{let} x\, [\Delta] = M \operatorname{in} K M = \{[\Delta]M/x\}M : \{[\Delta]M/x\}K}$$

as well as an equality rule (let_eq) for let-terms, and versions of (let), (let_use) and (let_eq) for kinds, whose form can be inferred from the rules above.

If we wish to build PAL$^+$ using this feature, we would also need a feature enabling α-ary variables to stand as terms on their own; that is, introducing α-ary terms without introducing lambda abstractions. Possibly a better way to construct PAL$^+$ would be to build a feature for let-definitions similar to that above that depends on the feature $\mathbf{L}\lambda^{\beta\eta}(\omega)$, identifying the abstraction $[\Delta]M$ with the term $\operatorname{let} x\, [\Delta] = M \operatorname{in} K x$ for some appropriate kind K.

If we introduce clauses of the form $c[\Delta] = M : K$ to the *signatures*, where c is a *constant*, with rules similar to (def_var), (def_var_out) and (def_var_use) above, we obtain a feature giving *global definitions* of the kind found in the AUTOMATH systems. With these, we would be able to build PAL and AUT-68.

9.1.2 Logic as Primitive

To handle logic in the way that a derivability-as-primitive framework such as Isabelle does — that is, recording which propositions are derivable, but without terms representing the derivations — we shall need a new judgement form, "M true", to indicate that the proposition represented by M is derivable. We shall need a new kind, **Prop**, whose objects are the terms that may form the body of this judgement.

We need a way to represent both axioms and rules of deduction. An axiom M, possibly parametrised by the context Δ, will be represented as

$$(\Rightarrow M)$$

The rule of deduction allowing the proposition N to be derived from the propositions M_1, …, M_n will be written

$$(M_1, \ldots, M_n \Rightarrow N)$$

The natural deduction rule

$$\dfrac{\begin{matrix}[P_{11}] \cdots [P_{1k_1}] \\ \vdots \\ M_1\end{matrix} \qquad \cdots \qquad \begin{matrix}[P_{n1}] \cdots [P_{nk_n}] \\ \vdots \\ M_n\end{matrix}}{N}$$

will be written

$$((P_{11}, \ldots, P_{1k_1} \Rightarrow M_1), \ldots, (P_{n1}, \ldots, P_{nk_n} \Rightarrow M_n) \Rightarrow N)$$

In general, an mth-order *rule of deduction* will be an expression of the form

$$(\Delta)(R_1, \ldots, R_n \Rightarrow M)$$

where $R_1, \ldots, R_n$ are $m-1$st-order rules. This rule indicates that, when the *parameters* Δ are filled, if the *premises*, the rules $R_1, \ldots, R_n$, are all derivable, then the *conclusion* M is true. Thus, for example, modus ponens would be the first-order rule

$$(p, q : \mathbf{Prop})(p, \rightarrow [p, q] \Rightarrow q)$$

and the introduction rule for $\rightarrow$ would be the second-order rule

$$(p, q : \mathbf{Prop})(()(p \Rightarrow q) \Rightarrow \rightarrow [p, q])$$

As usual, we may define various versions of this feature, depending on what order the parameters Δ in a rule may take, and which orders of rule may appear. The new grammatical clauses introduced would be:

Base kind	K	$::=$	$\cdots \mid \mathbf{Prop}$
mth-order rule	R^m	$::=$	$(\Delta^\alpha)(R^{m-1}, \ldots, R^{m-1} \Rightarrow M^{\mathbf{0}})$
Context Declaration	δ	$::=$	$\cdots \mid R^{m-1}$ of none
Signature Declaration	γ	$::=$	$\cdots \mid R^m$ of none
Judgement Body	J	$::=$	$\cdots \mid M^{\mathbf{0}}$ true

We introduce a defined judgement form, "R rule", for "R is a well-formed rule": the rule $(\Delta)(R_1, \ldots, R_n \Rightarrow M)$ is well-formed iff, under Δ, $R_1, \ldots, R_n$ are all well-formed rules, and $M : \mathbf{Prop}$. The rules of deduction introduced by the feature are then:

$$(\mathrm{rule}_m)\ \dfrac{\Delta^\alpha \vdash_\Sigma R^m \text{ rule}}{\Sigma, R^m \text{ sig}}$$

$$(\mathrm{hyp}_{m-1})\ \dfrac{\Gamma \vdash_\Sigma R^{m-1} \text{ rule}}{\Gamma, R^{m-1} \vdash_\Sigma \text{valid}}$$

$$(\text{rule\_use}_m)\quad \frac{\begin{array}{c}\Gamma \models_\Sigma \hat{\overrightarrow{M}}^\alpha :: \Delta^\alpha \\ \Gamma \vdash_\Sigma \{\hat{\overrightarrow{M}}^\alpha / \operatorname{dom}\Delta^\alpha\} R_1^{m-1} \text{ true} \\ \vdots \\ \Gamma \vdash_\Sigma \{\hat{\overrightarrow{M}}^\alpha / \operatorname{dom}\Delta^\alpha\} R_k^{m-1} \text{ true}\end{array}}{\Gamma \vdash_\Sigma \{\hat{\overrightarrow{M}}^\alpha / \operatorname{dom}\Delta^\alpha\} M^{\mathbf{0}} \text{ true}} \; ((\Delta^\alpha)(R_1^{m-1}, \ldots, R_k^{m-1} \Rightarrow M^{\mathbf{0}}) \in \Sigma)$$

$$(\text{hyp\_use}_{m-1})\quad \frac{\begin{array}{c}\Gamma \models_\Sigma \hat{\overrightarrow{M}}^\alpha :: \Delta^\alpha \\ \Gamma \vdash_\Sigma \{\hat{\overrightarrow{M}}^\alpha / \operatorname{dom}\Delta^\alpha\} R_1^{m-2} \text{ true} \\ \vdots \\ \Gamma \vdash_\Sigma \{\hat{\overrightarrow{M}}^\alpha / \operatorname{dom}\Delta^\alpha\} R_k^{m-2} \text{ true}\end{array}}{\Gamma \vdash_\Sigma \{\hat{\overrightarrow{M}}^\alpha / \operatorname{dom}\Delta^\alpha\} M^{\mathbf{0}} \text{ true}} \; ((\Delta^\alpha)(R_1^{m-2}, \ldots, R_k^{m-2} \Rightarrow M^{\mathbf{0}}) \in \Gamma)$$

9.2 Final Remarks

The field of logical frameworks is an exciting and fast-moving field of research. However, we contend that the world of frameworks is at present rather chaotic, and in need of some structure provided by a general theory of frames. If it is immodest to hope that the work in this thesis will directly lead to such a general theory, at least we hope that the results given here form evidence that such a theory is possible, and show the shape it could form. At the least, it does seem that the extraction of a conservative subsystem is a powerful method to add to one's armoury for the theoretical investigation of frameworks, and that considering a frame as composed of features is a useful way to find such subsystems. At best, the future development of this field may vindicate the moral of this thesis: that a logical framework is nothing but a set of features.

Bibliography

[Acz] Peter Aczel. Yet another logical framework. Unpublished.

[Ada] Robin Adams. Pure type systems with judgemental equality. Submitted for publication in the Journal of Functional Programming.

[And71] P. B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 36:414–432, 1971.

[Cer96] Iliano Cervesato. *A Linear Logical Framework*. PhD thesis, Università di Torino, 1996.

[CH88] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[CL] Paul Callaghan and Zhaohui Luo. An implementation of lf with coercive subtyping & universes.

[Coq91] Thierry Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.

[deB91] N. G. deBruijn. A plea for weaker frameworks. In Gérard Huet and G. Plotkin, editors, *Logical Frameworks*, pages 40–67. Cambridge University Press, 1991.

[Dyb94] Peter Dybjer. Inductive families. *Formal Aspects of Computing*, 6(4):440–465, 1994.

[Eri93] Lars-Henrik Eriksson. *Finitary Partial Inductive Definitions and General Logic*. PhD thesis, Department of Computer and System Sciences, Royal Institute of Technology, Stockholm, 1993.

[Eri94] Lars-Henrik Eriksson. Pi: An interactive derivation editor for the calculus of partial inductive definitions. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, volume 814 of *LNAI*, pages 821–825, Nancy, France, June 1994. Springer-Verlag.

[Fef88] Solomon Feferman. Finitary inductive systems. In R. Ferro, editor, *Proceedings of Logic Colloquium '88*, pages 191–220, Padova, Italy, August 1988. North-Holland.

[GB99] Herman Geuvers and Erik Barendsen. Some logical and syntactical observations concerning the first-order dependent type system λP. *Math. Struct. in Comp. Science.*, 9:335–359, 1999.

[Hal91] Lars Hallnäs. Partial inductive definitions. *Theoretical Computer Science*, 87(1):115–142, September 1991.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993. A preliminary version appeared in the *Proceedings of the Symposium on Logic in Computer Science*, pages 194–204, June 1987.

[Hod01] Wilfrid Hodges. *Logic.* Penguing, 2001.

[Luo94] Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science.* Number 11 in International Series of Monographs on Computer Science. Oxford University Press, 1994.

[Luo99] Zhaohui Luo. Coercive subtyping. *Journal of Logic and Computation*, 9(1):105–130, 1999.

[Luo03] Zhaohui Luo. PAL$^+$: a lambda-free logical framework. *Journal of Functional Programming*, 13(2):317–338, 2003.

[Mag95] Lena Magnusson. *The Implementation of ALF — A Proof Editor Based on Martin-Löf's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Chalmers University of Technology and Göteburg University, January 1995.

[ML85] Per Martin-Löf. Truth of a proposition, evidence of a judgement, validity of a proof. Transcript of a talk at the workshop Theories of Meaning, Centro Fiorentino di Storia e Filosofia della Scienza, Villa di Mondeggi, France, June 1985.

[MNPS91] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[MSB93] Seán Matthews, Alan Smaill, and David Basin. Experience with fs_0 as a framework theory. In Gérard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 61–82. Cambridge University Press, 1993.

[NGV94] R. P. Nederpelt, J. H. Geuvers, and R. C. D. Vrijer, editors. *Selected Papers on AUTOMATH.* Number 133 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1994.

[NM88] Gopalan Nadathur and Dale Miller. An overview of λProlog. In Kenneth A. Bowen and Robert A. Kowalski, editors, *Fifth International Logic Programming Conference*, pages 810–827, Seattle, Washington, August 1988. MIT Press.

[NP92] Tobias Nipkow and Lawrence C. Paulson. Isabelle-91. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *LNAI*, pages 673–676, Saratoga Springs, NY, 1992. Springer-Verlag.

[NPS90] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: an Introduction.* Oxford University Press, 1990.

[Pau86] Lawrence C. Paulson. Natural deduction as higher-order resolution. *Journal of Logic Programming*, 3:237–258, 1986.

[Pau89] Lawrence C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[Pfe] Frank Pfenning. Bibliography on logical frameworks. `http://www-2.cs.emu.edp/ fp/lfs-bib.html`.

[Pfe96] Frank Pfenning. The practice of logical frameworks. In *CAAP '96*, Linköping, Sweden, April 1996.

[Pra65] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study.* Number 3 in Stockholm Studies in Philosophy. Almqvist and Wiskells, 1965.

[Sch77] K. Schütte. *Proof Theory.* Springer-Verlag, 1977.

[SH84] Peter Schroeder-Heister. A natural extension of natural deduction. *Journal of Symbolic Logic*, 49(4):1284 – 1300, Dec 1984.

[Sun83] Göran Sundholm. Systems of deduction. In D. M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 1, chapter 2, pages 133–188. D. Reidel, 1983.

[Tak75] G. Takeuti. *Proof Theory.* North-Holland, 1975.

[WCPW03] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework: the propositional fragment. In *Types for Proofs and Programs*, LNCS, Torino, Italy, April 2003. Springer-Verlag.

Appendix A

List of Features

Feature	Dependencies	Syntactic Classes Introduced
BF		Base Term Base Type Base Kind
$\mathbf{SPar}\,(n)$ $(n \geqslant 0)$	**BF** (if $n = 1$) $\mathbf{SPar}\,(n-1)$ (if $n > 1$)	**m**-ary Term **m**-ary Type **m**-ary Kind
$\mathbf{LPar}\,(n)$ $(n \geqslant 0)$	$\mathbf{SPar}\,(n)$ $\mathbf{LPar}\,(n-1)$ (if $n > 1$)	
$\mathbf{S}\lambda^{R}\,(n)$ $(R \in \{\beta, \beta\eta\})$ $(n \geqslant 1)$	$\mathbf{SPar}\,(n)$ $\mathbf{S}\lambda^{T}\,(R)\,n-1$ (if $n > 1$)	α-ary Term α-ary Type α-ary Kind $(\mathrm{Ord}\,(\alpha) = n)$
$\mathbf{L}\lambda^{R}\,(n)$ $(R \in \{\beta, \beta\eta\}$ $(n \geqslant 1)$	$\mathbf{LPar}\,(n)$ $\mathbf{L}\lambda^{T}\,(R)\,n-1$ (if $n > 1$)	α-ary Term α-ary Type α-ary Kind $(\mathrm{Ord}\,(\alpha) = n)$
$\mathbf{XEq}\,(m,n)$ $(\mathbf{X} \in \{\mathbf{S}, \mathbf{L}\})$ $(n \geqslant 0)$	$\mathbf{X}\lambda^{R}\,(n)$ (if $n > 0$) $\mathbf{XPar}(m-1)$ (if $m > 0$)	
$\mathbf{XDef}\,(m,n)$ $(\mathbf{X} \in \{\mathbf{S}, \mathbf{L}\})$ $(m \geqslant 0)$ $(n \geqslant 0)$	$\mathbf{X}\lambda^{R}\,(n)$ (if $n > 0$) $\mathbf{XPar}m-1$ (if $m > 0$) $\mathbf{XDef}\,(m-1,n)$ (if $m > 0$)	

Appendix B

Judgemental Equality

We shall present here three systems: ELF, PAL, and AUT-68. Each of them we present in two versions: one which uses an external relation of convertibility (the way each system was originally presented), and one which possesses judgement forms for equality of terms and kinds (the version we used in the main body of this thesis). Our aim is to prove the two versions of each system equivalent. The proof method is almost identical for each system.

For the purposes of this chapter, we shall call the systems introduced in Chapter 8 ELF$_=$, PAL$_=$ and AUT-68$_=$. The versions without judgemental equality shall be called ELF, PAL and AUT-68.

It should thus be clear that this method could be applied to a wide range of type systems, provided those systems possess the uniqueness of types property, on which this proof method relies heavily. In particular, it can be used to prove a similar result for any functional PTS[Ada].

B.1 The Edinburgh Logical Framework, ELF

The version with judgemental equality, ELF$_=$, was given in Chapter 8. When we write a judgement of this system, we shall use the turnstile $\vdash^=$ to distinguish such judgements from those of the system without judgemental equality that we shall shortly introduce.

The results about this system that we shall need are as follows. The proofs are straightforward in each case.

Lemma B.1 (Context Validity) *If* $\Gamma, \Delta \vdash^=_\Sigma J$ *then* $\Gamma \vdash^=_\Sigma \cdot$.

Lemma B.2 (Context and Signature Conversion) *1. If* $\Gamma \vdash^=_{\Sigma_1, a:K, \Sigma_2} J$, $\vdash^=_{\Sigma_1} K = K'$ *and* $\vdash^=_{\Sigma_1} K'$, *then* $\Gamma \vdash^=_{\Sigma_1, a:K', \Sigma_2} J$.

2. If $\Gamma \vdash^=_{\Sigma_1, c:A, \Sigma_2} J$, $\vdash^=_{\Sigma_1} A = B : \mathbf{Type}$, *and* $\vdash^=_{\Sigma_1, c:B, \Sigma_2} J$.

3. If $\Gamma, x : A, \Gamma' \vdash^=_\Sigma J$, $\Gamma \vdash^=_\Sigma A = B : \mathbf{Type}$, *and* $\Gamma \vdash^=_\Sigma B : \mathbf{Type}$, *then* $\Gamma, x : B, \Gamma' \vdash^=_\Sigma J$.

Lemma B.3 (Substitution) *If* $\Gamma, x : A, \Delta \vdash^=_\Sigma J$ *and* $\Gamma \vdash^=_\Sigma M : A$, *then* $\Gamma, [M/x]\Delta \vdash^=_\Sigma [M/x] J$.

Lemma B.4 (Functionality) *1. If $\Gamma, x : A, \Delta \vdash^{=}_{\Sigma} M : B$ and $\Gamma \vdash^{=}_{\Sigma} N = P : A$, then $\Gamma, [N/x]\,\Delta \vdash^{=}_{\Sigma} [N/x]\,M = [P/x]\,M : [N/x]\,B$.*

2. If $\Gamma, x : A, \Delta \vdash^{=}_{\Sigma} B : K$ and $\Gamma \vdash^{=}_{\Sigma} M = N : A$, then $\Gamma, [M/x]\,\Delta \vdash^{=}_{\Sigma} [M/x]\,B = [N/x]\,B : [M/x]\,K$.

3. If $\Gamma, x : A, \Delta \vdash^{=}_{\Sigma} K$ and $\Gamma \vdash^{=}_{\Sigma} M = N : A$, then $\Gamma, [M/x]\,\Delta \vdash^{=}_{\Sigma} [M/x]\,K = [N/x]\,K$.

Lemma B.5 (Generation) *1. If $\Gamma \vdash^{=}_{\Sigma} a : K$, then $a : K' \in \Sigma$ for some K', and $\Gamma \vdash^{=}_{\Sigma} K = K'$.*

2. If $\Gamma \vdash^{=}_{\Sigma} c : A$, then $c : B \in \Sigma$ for some B, and $\Gamma \vdash^{=}_{\Sigma} A = B : \mathbf{Type}$.

3. If $\Gamma \vdash_{=} x : A$, then $x : B \in \Gamma$ for some B, and $\Gamma \vdash^{=}_{\Sigma} A = B : \mathbf{Type}$.

4. If $\Gamma \vdash^{=}_{\Sigma} \Pi x : A.K$, then $\Gamma \vdash^{=}_{\Sigma} A : \mathbf{Type}$ and $\Gamma, x : A \vdash^{=}_{\Sigma} K$.

5. If $\Gamma \vdash^{=}_{\Sigma} \Pi x : A.B : K$, then $\Gamma \vdash_{=} A : \mathbf{Type}$, $\Gamma, x : A \vdash^{=}_{\Sigma} B : \mathbf{Type}$, and $\Gamma \vdash^{=}_{\Sigma} K = Type$.

6. If $\Gamma \vdash^{=}_{\Sigma} \lambda x : A.B : K$, then $\Gamma \vdash^{=}_{\Sigma} A : \mathbf{Type}$, and there is a kind K' such that $\Gamma, x : A \vdash^{=}_{\Sigma} B : K'$ and $\Gamma \vdash^{=}_{\Sigma} K = \Pi x : A.K'$.

7. If $\Gamma \vdash^{=}_{\Sigma} \lambda x : A.M : B$, then $\Gamma \vdash^{=}_{\Sigma} A : \mathbf{Type}$, and there is a family C such that $\Gamma, x : A \vdash^{=}_{\Sigma} M : C$ and $\Gamma \vdash^{=}_{\Sigma} B = \Pi x : A.C : \mathbf{Type}$.

8. If $\Gamma \vdash^{=}_{\Sigma} AM : K$, then there is a family B and kind K' such that $\Gamma \vdash^{=}_{\Sigma} A : \Pi x : B.K'$, $\Gamma \vdash^{=}_{\Sigma} M : B$, and $\Gamma \vdash^{=}_{\Sigma} [M/x]K' = K$.

9. If $\Gamma \vdash^{=}_{\Sigma} MN : A$, then there are families B, C such that $\Gamma \vdash^{=}_{\Sigma} M : \Pi x : B.C$, $\Gamma \vdash^{=}_{\Sigma} N : B$, and $\Gamma \vdash^{=} [N/x]C = A : \mathbf{Type}$.

Lemma B.6 (Type Validity and Equation Validity) *1. If $\Gamma \vdash^{=}_{\Sigma} A : K$, then $\Gamma \vdash^{=}_{\Sigma} K$.*

2. If $\Gamma \vdash^{=}_{\Sigma} M : A$, then $\Gamma \vdash^{=}_{\Sigma} A : \mathbf{Type}$.

3. If $\Gamma \vdash^{=}_{\Sigma} K = K'$, then $\Gamma \vdash^{=}_{\Sigma} K$ and $\Gamma \vdash^{=}_{\Sigma} K'$.

4. If $\Gamma \vdash^{=}_{\Sigma} A = B : K$, then $\Gamma \vdash^{=}_{\Sigma} A : K$ and $\Gamma \vdash^{=}_{\Sigma} B : K$.

5. If $\Gamma \vdash^{=}_{\Sigma} M = N : A$, then $\Gamma \vdash^{=}_{\Sigma} M : A$ and $\Gamma \vdash^{=}_{\Sigma} N : A$.

We can now present the version of ELF without judgemental equality.

Grammar

Kind	K	::=	$\mathbf{Type} \mid \Pi x : A.K$
Family	A	::=	$a \mid \Pi x : A.A \mid \lambda x : A.A \mid AM$
Object	M	::=	$x \mid c \mid \lambda x : A.M \mid MM$
Signature Declaration	γ	::=	$a : K$ of $a \mid c : A$ of c
Context Declaration	δ	::=	$x : A$ of x
Judgement Body	J	::=	$\cdot \mid K \mid A : K \mid M : A$

Convertibility We define the relation of *β-convertibility* $\simeq_\beta$ on kinds, families and objects as usual, based on the contractions

$$\begin{array}{lcl} (\lambda x : A.B)M & \rightsquigarrow_\beta & [M/x]B \\ (\lambda x : A.M)N & \rightsquigarrow_\beta & [N/x]M \end{array}$$

Rules of Deduction

$$(\texttt{B-EMPTY-SIG})\ \frac{}{\langle\rangle \text{ sig}}$$

$$(\texttt{B-KIND-SIG})\ \frac{\vdash_\Sigma K \text{ kind}}{\Sigma, a : K \text{ sig}}\ (a \notin \operatorname{dom}\Sigma) \qquad (\texttt{B-TYPE-SIG})\ \frac{\vdash_\Sigma A : \mathbf{Type}}{\Sigma, c : A \text{ sig}}\ (c \notin \operatorname{dom}\Sigma)$$

$$(\texttt{B-EMPTY-CTX})\ \frac{\Sigma \text{ sig}}{\vdash_\Sigma \cdot} \qquad (\texttt{B-TYPE-CTX})\ \frac{\Gamma \vdash_\Sigma A : \mathbf{Type}}{\Gamma, x : A \vdash_\Sigma \cdot}\ (x \notin \operatorname{dom}\Gamma)$$

$$(\texttt{B-TYPE-KIND})\ \frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma \mathbf{Type}} \qquad (\texttt{B-PI-KIND})\ \frac{\Gamma, x : A \vdash_\Sigma K}{\Gamma \vdash_\Sigma \Pi x : A.K}$$

$$(\texttt{B-CONST-FAM})\ \frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma c : K}\ (c : K \in \Sigma) \qquad (\texttt{B-PI-FAM})\ \frac{\Gamma, x : A \vdash_\Sigma B : \mathbf{Type}}{\Gamma \vdash_\Sigma \Pi x : A.B : \mathbf{Type}}$$

$$(\texttt{B-ABS-FAM})\ \frac{\Gamma, x : A \vdash_\Sigma B : K}{\Gamma \vdash_\Sigma \lambda x : A.B : \Pi x : A.K} \qquad (\texttt{B-APP-FAM})\ \frac{\Gamma \vdash_\Sigma A : \Pi x : B.K \quad \Gamma \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma AM : [M/x]K}$$

$$(\texttt{B-CONV-FAM})\ \frac{\Gamma \vdash_\Sigma A : K \quad \Gamma \vdash_\Sigma K'}{\Gamma \vdash_\Sigma A : K'}\ (K \simeq_\beta K')$$

$$(\texttt{B-CONST-OBJ})\ \frac{\Gamma \vdash_\Sigma \cdot}{\Gamma \vdash_\Sigma c : A}\ (c : A \in \Sigma) \qquad (\texttt{B-ABS-OBJ})\ \frac{\Gamma, x : A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda x : A.M : \Pi x : A.B}$$

$$(\texttt{B-APP-OBJ})\ \frac{\Gamma \vdash_\Sigma M : \Pi x : A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : [N/x]B}$$

$$(\texttt{B-CONV-OBJ})\ \frac{\Gamma \vdash_\Sigma M : A \quad \Gamma \vdash_\Sigma B : \mathbf{Type}}{\Gamma \vdash_\Sigma M : B}\ (A \simeq_\beta B)$$

We shall need the following result; for its proof, see [HHP93].

Theorem B.7 (Church-Rosser) *1. If $K \simeq_\beta K'$, there exists a kind K_0 such that $K \twoheadrightarrow_\beta K_0$ and $K' \twoheadrightarrow_\beta K_0$.*

2. If $A \simeq_\beta B$, there exists a family C such that $A \twoheadrightarrow_\beta C$ and $B \twoheadrightarrow_\beta C$.

3. If $M \simeq_\beta N$, there exists an object P such that $M \twoheadrightarrow_\beta P$ and $N \twoheadrightarrow_\beta P$.

The main result that we shall prove is the following:

1. Σ sig is derivable in ELF$_=$ iff it is derivable in ELF.

2. $\Gamma \vdash_\Sigma^= \cdot$ iff $\Gamma \vdash_\Sigma \cdot$.

3. $\Gamma \vdash_\Sigma^= K$ iff $\Gamma \vdash_\Sigma K$.

4. $\Gamma \vdash^{=}_{\Sigma} K = K'$ iff $\Gamma \vdash_{\Sigma} K$, $\Gamma \vdash_{\Sigma} K'$ and $K \simeq_{\beta} K'$.

5. $\Gamma \vdash^{=}_{\Sigma} A : K$ iff $\Gamma \vdash_{\Sigma} A : K$.

6. $\Gamma \vdash^{=}_{\Sigma} A = B : K$ iff $\Gamma \vdash_{\Sigma} A : K$, $\Gamma \vdash_{\Sigma} B : K$, and $A \simeq_{\beta} B$.

7. $\Gamma \vdash^{=}_{\Sigma} M : A$ iff $\Gamma \vdash_{\Sigma} M : A$.

8. $\Gamma \vdash^{=}_{\Sigma} M = N : A$ iff $\Gamma \vdash_{\Sigma} M : A$, $\Gamma \vdash N : A$ and $M \simeq_{\beta} N$.

One direction is easy:

Theorem B.8 *1. If Σ sig is derivable in ELF$_=$, then it is derivable in ELF.*

2. If $\Gamma \vdash^{=}_{\Sigma} \cdot$ then $\Gamma \vdash_{\Sigma} \cdot$.

3. If $\Gamma \vdash^{=}_{\Sigma} K$ then $\Gamma \vdash_{\Sigma} K$.

4. If $\Gamma \vdash^{=}_{\Sigma} K = K'$ then $\Gamma \vdash_{\Sigma} K$, $\Gamma \vdash_{\Sigma} K'$ and $K \simeq_{\beta} K'$.

5. If $\Gamma \vdash^{=}_{\Sigma} A : K$ then $\Gamma \vdash_{\Sigma} A : K$.

6. If $\Gamma \vdash^{=}_{\Sigma} A = B : K$ then $\Gamma \vdash_{\Sigma} A : K$, $\Gamma \vdash_{\Sigma} B : K$, and $A \simeq_{\beta} B$.

7. If $\Gamma \vdash^{=}_{\Sigma} M : A$ then $\Gamma \vdash_{\Sigma} M : A$.

8. If $\Gamma \vdash^{=}_{\Sigma} M = N : A$ then $\Gamma \vdash_{\Sigma} M : A$, $\Gamma \vdash N : A$ and $M \simeq_{\beta} N$.

Proof Induction on the derivation of the premise in ELF$_=$. No case is very difficult, if we have the results of [HHP93] to hand. We give the details of two cases here:

$$\frac{\Gamma \vdash^{=}_{\Sigma} A = A' : \mathbf{Type} \quad \Gamma, x : A \vdash^{=}_{\Sigma} M = N : B}{\Gamma \vdash^{=}_{\Sigma} \lambda x : A.M = \lambda x : A'.N : \Pi x : A.B}$$

By i.h., $A =_{\beta} A'$ and $M =_{\beta} N$, so $\lambda x : A.M =_{\beta} \lambda x : A'.N$.

From the i.h., $\Gamma, x : A \vdash_{\Sigma} M : B$, and so $\Gamma \vdash_{\Sigma} \lambda x : A.M : \Pi x : A.B$.

Also from the i.h., $\Gamma, x : A \vdash N : B$ and $\Gamma \vdash A' : \mathbf{Type}$. By Context Conversion, $\Gamma, x : A' \vdash N : B$ and so $\Gamma \vdash \lambda x : A'.N : \Pi x : A'.B$. Also, $\Gamma \vdash \Pi x : A.B : \mathbf{Type}$, so (`B-CONV-OBJ`) gives $\Gamma \vdash \lambda x : A'.N : \Pi x : A.B$.

$$\text{(app-eq)}\ \frac{\Gamma \vdash_{=} M = M' : \Pi x : A.B \quad \Gamma \vdash_{=} N = N' : A}{\Gamma \vdash_{=} MN = M'N' : [N/x]\,B}$$

By i.h., $M =_{\beta} M'$ and $N =_{\beta} N'$, so $MN =_{\beta} M'N'$.

From the i.h., $\Gamma \vdash M : \Pi x : A.B$ and $\Gamma \vdash N : A$. Therefore, by (`B-APP-OBJ`), $\Gamma \vdash MN : [N/x]\,B$.

Also from the i.h., $\Gamma \vdash M' : \Pi x : A.B$ and $\Gamma \vdash N' : A$, so $\Gamma \vdash M'N' : [N'/x]\,B$.

Now, Type Validity gives a sort s_3 such that

$$\Gamma \vdash \Pi x : A.B : \mathbf{Type}$$

Therefore, by Generation,

$$\Gamma \vdash A : \mathbf{Type}, \qquad \Gamma, x : A \vdash B : \mathbf{Type}$$

Substitution therefore yields

$$\Gamma \vdash [N/x]\, B : \mathbf{Type}$$

and so, by (`B-CONV-OBJ`),

$$\Gamma \vdash M'N' : [N/x]\, B$$

□

B.1.1 Typed Parallel One-Step Reduction

Define the set of *extended expressions*, *extended contexts* and *extended signatures* as follows:

Extended Kind	K^+	$::=$	$\mathbf{Type} \mid \Pi x : A^+.K^+$
Extended Family	A^+	$::=$	$a \mid \Pi x : A^+.A^+ \mid \lambda x : A^+.A^+$
			$\mid \mathrm{app}_{(x)K^+}(A^+M^+)$
Extended Object	M^+	$::=$	$x \mid c \mid \lambda x : A^+.M^+ \mid \mathrm{app}_{(x)A^+}(M^+M^+)$
Extended Signature Declaration	γ^+	$::=$	$a : K^+$ of a $\mid$ $c : A^+$ of c
Extended Context Declaration	δ^+	$::=$	$x : A^+$ of x

We define β-reduction, conversion, etc. on extended expressions as we did for the standard set of expressions, based on the contractions

$$\begin{array}{lcl} \mathrm{app}_{(x)K^+}((\lambda x : A^+.B^+)M^+) & \rightsquigarrow_\beta & [M^+/x]B^+ \\ \mathrm{app}_{(x)A^+}((\lambda x : B^+.M^+)N^+) & \rightsquigarrow_\beta & [N^+/x]M^+ \end{array}$$

We define the obvious translation

$$X^+ \mapsto |X^+|$$

from the set of extended expressions to the standard set of expressions for each system. It is easily shown that, if $M \twoheadrightarrow_\beta N$ then $|M| \twoheadrightarrow_\beta |N|$, if $M =_\beta N$ then $|M| =_\beta |N|$, and so forth.

Now, we define several relations among these syntactic classes:

$$\begin{array}{lll} \Sigma^+ & \models & \mathrm{sig} \\ \Gamma^+ & \models_{\Sigma^+} & \mathrm{valid} \\ \Gamma^+ & \models_{\Sigma^+} & K_1^+ \twoheadrightarrow K_2^+ \\ \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_2^+ : K^+ \\ \Gamma^+ & \models_{\Sigma^+} & M_1^+ \twoheadrightarrow M_2^+ : A^+ \end{array}$$

The definition is given inductively by the rules of deduction in Figures B.1 and B.2

$$\frac{}{\langle\rangle \models\ \mathrm{sig}}$$

$$\frac{\models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+}{\Sigma^+, a : K_1^+ \models\ \mathrm{sig}}\,(a \notin \mathrm{dom}\,\Sigma^+) \qquad \frac{\models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type}}{\Sigma^+, c : A_1^+ \models\ \mathrm{sig}}\,(c \notin \mathrm{dom}\,\Sigma^+)$$

$$\frac{\Sigma^+ \models\ \mathrm{sig}}{\models_{\Sigma^+} \mathrm{valid}} \qquad \frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type}}{\Gamma^+, x : A_1^+ \models_{\Sigma^+} \mathrm{valid}}\,(x \notin \mathrm{dom}\,\Gamma^+)$$

$$\frac{\Gamma^+ \models_{\Sigma^+} \mathrm{valid}}{\Gamma^+ \models_{\Sigma^+} \mathbf{Type} \twoheadrightarrow \mathbf{Type}} \qquad \frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \quad \Gamma^+, x : A_1^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+}{\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K_1^+ \twoheadrightarrow \Pi x : A_2^+.K_2^+}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} \mathrm{valid}}{\Gamma^+ \models_{\Sigma^+} a \twoheadrightarrow a : K^+}\,(a : K^+ \in \Sigma^+)$$

$$\frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \quad \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type}}{\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.B_1^+ \twoheadrightarrow \Pi x : A_2^+.B_2^+ : \mathbf{Type}}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \quad \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : K^+}{\Gamma^+ \models_{\Sigma^+} \lambda x : A_1^+.B_1^+ \twoheadrightarrow \lambda x : A_2^+.B_2^+ : \Pi x : A_1^+.K^+}$$

$$\frac{\begin{array}{cc}\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \Pi x : B_1^+.K_1^+ & \Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : B_1^+ \\ \Gamma^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type} & \Gamma^+, x : B^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+\end{array}}{\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)K_1^+}(A_1^+ M_1^+) \twoheadrightarrow \mathrm{app}_{(x)K_2^+}(A_2^+ M_2^+) : [M_1^+/x]K_1^+}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} \mathrm{valid}}{\Gamma^+ \models_{\Sigma^+} x \twoheadrightarrow x : A^+}\,(x : A^+ \in \Gamma^+) \qquad \frac{\Gamma^+ \models_{\Sigma^+} \mathrm{valid}}{\Gamma^+ \models_{\Sigma^+} c \twoheadrightarrow c : A^+}\,(c : A^+ \in \Sigma^+)$$

$$\frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \quad \Gamma^+, x : A_1^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : B^+}{\Gamma^+ \models_{\Sigma^+} \lambda x : A_1^+.M_1^+ \twoheadrightarrow \lambda x : A_2^+.M_2^+ : \Pi x : A_1^+.B^+}$$

$$\frac{\begin{array}{cc}\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : \Pi x : A_1^+.B_1^+ & \Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_2^+ : A_1^+ \\ \Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} & \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type}\end{array}}{\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)B_1^+}(M_1^+ N_1^+) \twoheadrightarrow \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+) : [N_1^+/x]B_1^+}$$

Figure B.1: Typed Parallel One-Step Reduction for ELF

$$\frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : K_1^+ \quad \Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+}{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : K_2^+}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : K_2^+ \quad \Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+}{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : K_1^+}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : A_1^+ \quad \Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type}}{\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : A_2^+}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : A_2^+ \quad \Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type}}{\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : A_1^+}$$

$$\frac{\begin{array}{cc}\Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : K_1^+ & \Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : A_1^+ \\ \Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} & \Gamma^+, x : A_1^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+\end{array}}{\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)K^+}((\lambda x : A^+.B_1^+)M_1^+) \twoheadrightarrow [M_2^+/x]B_2^+ : [M_1^+/x]K^+}$$

$$\frac{\begin{array}{cc}\Gamma^+, x : A_1^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : B_1^+ & \Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_2^+ : A_1^+ \\ \Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} & \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type}\end{array}}{\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)B_1^+}((\lambda x : A_1^+.M_1^+)N_1^+) \twoheadrightarrow [N_2^+/x]M_2^+ : [N_1^+/x]B_1^+}$$

Figure B.2: Typed Parallel One-Step Reduction for ELF

We shall write $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow ? : A^+$ for

$$\exists N^+ \, \Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$$

and similarly for families and kinds.

We define $\twoheadrightarrow^+$ to be the transitive closure of the relation $\twoheadrightarrow$ just defined, and $=^*$ to be the symmetric, transitive closure. Thus, for example, the relation $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow^+ N^+ : A^+$ is defined by the rules

$$\frac{\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+}{\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow^+ N^+ : A^+} \qquad \frac{\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+ \quad \Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow P^+ : A^+}{\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow P^+ : A^+}$$

and the relation $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$ is defined by the rules

$$\frac{\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+}{\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+} \qquad \frac{\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+}{\Gamma^+ \models_{\Sigma^+} K_2^+ =^* K_1^+} \qquad \frac{\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+ \quad \Gamma^+ \models_{\Sigma^+} K_2^+ =^* K_3^+}{\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_3^+}$$

Lemma B.9 *1. If* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K_1^+$ *and* $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$*, then* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K_2^+$.

2. If $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$ *and* $\Gamma \models A^+ =^* B^+ : \mathbf{Type}$*, then* $\Gamma \models M^+ \twoheadrightarrow N^+ : B^+$.

Theorem B.10 *1. If* $\Sigma^+ \models \mathrm{sig}$ *then* $|\Sigma^+|\ \mathrm{sig}$ *is derivable in ELF*$_=$.

2. If $\Gamma^+ \models_{\Sigma^+} \mathrm{valid}$ *then* $|\Gamma^+| \vdash^{=}_{|\Sigma^+|} \cdot$.

3. If $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$*, then* $|\Gamma^+| \vdash^{=}_{|\Sigma^+|} |K_1^+| = |K_2^+|$.

4. *If* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K^+$, *then* $|\Gamma^+| \vdash^{=}_{|\Sigma^+|} |A^+| = |B^+| : |K^+|$.

5. *If* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, *then* $|\Gamma^+| \vdash^{=}_{|\Sigma^+|} |M^+| = |N^+| : |A^+|$.

Proof Proved simultaneously, by a straightforward induction on the premise. □

Lemma B.11 1. *If* $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$ *then* $K_1^+ \twoheadrightarrow_\beta K_2^+$.

2. *If* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K^+$ *then* $A^+ \twoheadrightarrow_\beta B^+$.

3. *If* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$ *then* $M^+ \twoheadrightarrow_\beta N^+$.

Corollary B.12 1. *If* $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$ *then* $K_1^+ \simeq_\beta K_2^+$.

2. *If* $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : K^+$ *then* $A^+ \simeq_\beta B^+$.

3. *If* $\Gamma^+ \models_{\Sigma^+} M^+ =^* N^+ : A^+$ *then* $M^+ \simeq_\beta N^+$.

Lemma B.13 (Context Validity) 1. *If* $\Sigma_1^+, \Sigma_2^+ \models \text{ sig}$ *or* $\Gamma^+ \models_{\Sigma_1^+,\Sigma_2^+} J$, *then* $\Sigma_1^+ \models \text{ sig}$.

2. *If* $\Gamma^+, \Delta^+ \models_{\Sigma^+} J$, *then* $\Gamma^+\text{valid}$.

Lemma B.14 (Free Variables and Constants) 1. *If* $\Sigma^+ \models \text{ sig}$, *then* Σ^+ *has no free variables. Further, if* $\Sigma^+ \equiv z_1 : Z_1^+, \ldots, z_n : Z_n^+$, *every constant in* Z_i^+ *is one of* $z_1, \ldots, z_{i-1}$.

2. *If* $\Gamma^+ \models_{\Sigma^+} J$, *then every free variable in* J *is in* Γ^+*; every constant in* J *or* Γ^+ *is in* $\operatorname{dom}\Sigma^+$*; and* Σ^+ *has no free variables. Further, if* $\Gamma^+ \equiv x_1 : A_1^+, \ldots, x_n : A_n^+$, *then every free variable in* A_i^+ *is one of* $x_1, \ldots, x_{i-1}$*; and if* $\Sigma^+ \equiv z_1 : Z_1^+, \ldots, z_n : Z_n^+$, *then every constant in* Z_i^+ *is one of* $z_1, \ldots, z_{i-1}$.

Lemma B.15 (Weakening) 1. *If* $\Gamma^+ \models_{\Sigma_1^+} J$, $\Sigma_1^+ \subseteq \Sigma_2^+$, *and* $\Sigma_2^+ \models \text{ sig}$, *then* $\Gamma^+ \models_{\Sigma_2^+} J$.

2. *If* $\Gamma^+ \models_{\Sigma^+} J$, $\Gamma^+ \subseteq \Delta^+$, *and* $\Delta^+ \models_{\Sigma^+} \text{valid}$, *then* $\Delta^+ \models_{\Sigma^+} J$.

Lemma B.16 (Context Reduction) 1. *If* $\Sigma_1^+, c : K_1^+, \Sigma_2^+ \models \text{ sig}$, $\models_{\Sigma_1^+} K_1^+ \twoheadrightarrow K_2^+$, *and* $\models_{\Sigma^+} K_2^+ \twoheadrightarrow K_2^+$, *then* $\Sigma_1^+, c : K_2^+, \Sigma_2^+ \models \text{ sig}$.

2. *If* $\Sigma_1^+, a : A^+, \Sigma_2^+ \models \text{ sig}$, $\models_{\Sigma_1^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, *and* $\models_{\Sigma_1^+} B^+ \twoheadrightarrow B^+ : \mathbf{Type}$, *then* $\Sigma_1^+, a : B^+, \Sigma_2^+ \models \text{ sig}$.

3. *If* $\Gamma^+ \models_{\Sigma_1^+, c:K_1^+, \Sigma_2^+} J$, $\models_{\Sigma_1^+} K_1^+ \twoheadrightarrow K_2^+$, *and* $\models_{\Sigma_1^+} K_2^+ \twoheadrightarrow K_2^+$, *then* $\Gamma^+ \models_{\Sigma_1^+, c:K_2^+, \Sigma_2^+} J$.

4. *If* $\Gamma^+ \models_{\Sigma_1^+, a:A^+, \Sigma_2^+} J$, $\models_{\Sigma_1^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, *and* $\models_{\Sigma_1^+} B^+ \twoheadrightarrow B^+ : \mathbf{Type}$, *then* $\Gamma^+ \models_{\Sigma_1^+, a:B^+, \Sigma_2^+} J$.

5. *If* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} J$, $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, *and* $\Gamma^+ \models_{\Sigma^+} B^+ \twoheadrightarrow B^+ : \mathbf{Type}$, *then* $\Gamma^+, x : B^+, \Delta^+ \models_{\Sigma^+} J$.

Lemma B.17 (Left-Hand Reflexivity) 1. *If* $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$, *then* $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_1^+$.

2. *If* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K^+$, *then* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow A^+ : K^+$.

3. *If* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, *then* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow M^+ : A^+$.

Lemma B.18 (Substitution) *1. If* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} \mathrm{valid}$ *and* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, *then* $\Gamma^+, [M^+/x]A^+ \models_{\Sigma^+} \mathrm{valid}$.

2. *If* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$ *and* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, *then* $\Gamma^+, [M^+/x]\Delta^+ \models_{\Sigma^+} [M^+/x]K_1^+ \twoheadrightarrow [N^+/x]K_2^+$.

3. *If* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} B^+ \twoheadrightarrow C^+ : K^+$ *and* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, *then* $\Gamma^+, [M^+/x]\Delta^+ \models_{\Sigma^+} [M^+/x]B^+ \twoheadrightarrow [M^+/x]C^+ : [M^+/x]K^+$.

4. *If* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : B^+$ *and* $\Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_2^+ : A^+$, *then* $\Gamma^+, [N_1^+/x]\Delta^+ \models_{\Sigma^+} [N_1^+/x]M_1^+ \twoheadrightarrow [N_2^+/x]M_2^+ : [N_1^+/x]B^+$.

Lemma B.19 (Right-Hand Reflexivity) *1. If* $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$ *then* $\Gamma^+ \models_{\Sigma^+} K_2^+ \twoheadrightarrow K_2^+$.

2. *If* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K^+$ *then* $\Gamma^+ \models_{\Sigma^+} B^+ \twoheadrightarrow B^+ : K^+$.

3. *If* $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$ *then* $\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow N^+ : A^+$.

Proof Induction on the premise. We give two cases here:

$$(\mathrm{app})\ \frac{\begin{array}{c}\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : \Pi x : A_1^+.B_1^+ \quad \Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_2^+ : A_1^+ \\ \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type}\end{array}}{\Gamma \models_{\Sigma^+} \mathrm{app}_{(x)B_1^+}(M_1^+ N_1^+) \twoheadrightarrow \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+) : [N_1^+/x]B_1^+}$$

By i.h.,

$$\begin{array}{rcl}\Gamma^+ & \models_{\Sigma^+} & M_2^+ \twoheadrightarrow M_2^+ : \Pi x : A_1^+.B_1^+ \\ \Gamma^+ & \models_{\Sigma^+} & N_2^+ \twoheadrightarrow N_2^+ : A_1^+ \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & B_2^+ \twoheadrightarrow B_2^+ : \mathbf{Type}\end{array}$$

By Context Validity, $\Gamma^+, x : A_1^+ \models_{\Sigma^+} \mathrm{valid}$; inverting, there exists A_2^+ such that

$$\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type}\,.$$

So:

$$\begin{array}{rcll}\Gamma^+ & \models_{\Sigma^+} & \Pi x : A_1^+.B_1^+ \twoheadrightarrow \Pi x : A_2^+.B_2^+ : \mathbf{Type} & (\mathrm{prod}) \\ \therefore \Gamma^+ & \models_{\Sigma^+} & M_2^+ \twoheadrightarrow M_2^+ : \Pi x : A_2^+.B_2^+ & (\mathrm{red}) \\ \Gamma^+ & \models_{\Sigma^+} & N_2^+ \twoheadrightarrow N_2^+ : A_2^+ & (\mathrm{red}) \\ \Gamma^+, x : A_2^+ & \models_{\Sigma^+} & B_2^+ \twoheadrightarrow B_2^+ : \mathbf{Type} & (\text{Context Reduction}) \\ \therefore \Gamma^+ & \models_{\Sigma^+} & \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+) \twoheadrightarrow \mathrm{app}_{(x)B^+2}(M_2^+ N_2^+) : [N_2^+/x]B_2^+ & (\mathrm{app})\end{array}$$

Now, Substitution gives

$$\Gamma \models [M_1^+/x]B_1^+ \twoheadrightarrow [M_2^+/x]B_2^+ : \mathbf{Type}$$

Therefore,

$$\Gamma \models \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+) \twoheadrightarrow \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+) : [N_1^+/x]B_1^+$$

$$\frac{\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \qquad \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type} \qquad \Gamma^+, x : A_1^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : B_1^+ \qquad \Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_2^+ : A_1^+}{\Gamma^+ \models \mathrm{app}_{(x)B_1^+}(M_1^+ N_1^+) \twoheadrightarrow [N_2^+/x]M_2^+ : [N_1^+/x]B_1^+}$$

By i.h.,

$$\begin{array}{rll}
\Gamma^+, x : A_1^+ & \models_{\Sigma^+} & M_2^+ \twoheadrightarrow M_2^+ : B_1^+ \\
\Gamma^+ & \models_{\Sigma^+} & N_2^+ \twoheadrightarrow N_2^+ : A \\
\therefore \Gamma^+ & \models_{\Sigma^+} & [N_2^+/x]M_2^+ \twoheadrightarrow [N_2^+/x]M_2^+ : [N_2^+/x]B_1^+ \qquad \text{(Substitution)} \\
\Gamma^+ & \models_{\Sigma^+} & [N_1^+/x]B_1^+ \twoheadrightarrow [N_2^+/x]B_2^+ : \mathbf{Type} \qquad \text{(Substitution)} \\
\therefore \Gamma^+ & \models_{\Sigma^+} & [N_2^+/x]M_2^+ \twoheadrightarrow [N_2^+/x]M_2^+ : [N_1^+/x]B_1^+ \qquad \text{(exp)}
\end{array}$$

□

Corollary B.20 *1. If $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$, then $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_1^+$ and $\Gamma^+ \models_{\Sigma^+} K_2^+ \twoheadrightarrow K_2^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : K^+$, then $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow A^+ : K^+$ and $\Gamma^+ \models_{\Sigma^+} B^+ \twoheadrightarrow B^+ : K^+$.

3. If $\Gamma^+ \models_{\Sigma^+} M^+ =^* N^+ : A^+$, then $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow M^+ : A^+$ and $\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow N^+ : A^+$.

This result now allows us to remove one of the premises of Context Reduction and state it in its usual form:

Corollary B.21 (Context Reduction) *1. If $\Sigma_1^+, c : K_1^+, \Sigma_2^+ \models$ sig and $\models_{\Sigma_1^+} K_1^+ \twoheadrightarrow K_2^+$, then $\Sigma_1^+, c : K_2^+, \Sigma_2^+ \models$ sig.*

2. If $\Sigma_1^+, a : A^+, \Sigma_2^+ \models$ sig and $\models_{\Sigma_1^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, then $\Sigma_1^+, a : B^+, \Sigma_2^+ \models$ sig.

3. If $\Gamma^+ \models_{\Sigma_1^+, c:K_1^+, \Sigma_2^+} J$ and $\models_{\Sigma_1^+} K_1^+ \twoheadrightarrow K_2^+$, then $\Gamma^+ \models_{\Sigma_1^+, c:K_2^+, \Sigma_2^+} J$.

4. If $\Gamma^+ \models_{\Sigma_1^+, a:A^+, \Sigma_2^+} J$ and $\models_{\Sigma_1^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, then $\Gamma^+ \models_{\Sigma_1^+, a:B^+, \Sigma_2^+} J$.

5. If $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} J$ and $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, then $\Gamma^+, x : B^+, \Delta^+ \models J$.

Lemma B.22 (Context Expansion) *1. If $\Sigma_1^+, c : K_2^+, \Sigma_2^+ \models$ sig and $\models_{\Sigma_1^+} K_1^+ \twoheadrightarrow K_2^+$, then $\Sigma_1^+, c : K_1^+, \Sigma_2^+ \models$ sig.*

2. If $\Sigma_1^+, a : B^+, \Sigma_2^+ \models$ sig and $\models_{\Sigma_1^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$, then $\Sigma_1^+, a : A^+, \Sigma_2^+ \models$ sig.

3. *If* $\Gamma^+ \models_{\Sigma_1^+, c:K_2^+, \Sigma_2^+} J$ *and* $\models_{\Sigma_1^+} K_1^+ \twoheadrightarrow K_2^+$*, then* $\Gamma^+ \models_{\Sigma_1^+, c:K_1^+, \Sigma_2^+} J$.

4. *If* $\Gamma^+ \models_{\Sigma_1^+, a:B^+, \Sigma_2^+} J$ *and* $\models_{\Sigma_1^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$*, then* $\Gamma^+ \models_{\Sigma_1^+, a:A^+, \Sigma_2^+} J$.

5. *If* $\Gamma^+, x : B^+, \Delta^+ \models_{\Sigma^+} J$ *and* $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$*, then* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} J$.

Corollary B.23 (Context Conversion) 1. *If* $\Sigma_1^+, c : K_1^+, \Sigma_2^+ \models \mathrm{sig}$ *and* $\models_{\Sigma_1^+} K_1^+ =^* K_2^+$*, then* $\Sigma_1^+, c : K_2^+, \Sigma_2^+ \models \mathrm{sig}$.

2. *If* $\Sigma_1^+, a : A^+, \Sigma_2^+ \models \mathrm{sig}$ *and* $\models_{\Sigma_1^+} A^+ =^* B^+ : \mathbf{Type}$*, then* $\Sigma_1^+, a : B^+, \Sigma_2^+ \models \mathrm{sig}$.

3. *If* $\Gamma^+ \models_{\Sigma_1^+, c:K_1^+, \Sigma_2^+} J$ *and* $\models_{\Sigma_1^+} K_1^+ =^* K_2^+$*, then* $\Gamma^+ \models_{\Sigma_1^+, c:K_2^+, \Sigma_2^+} J$.

4. *If* $\Gamma^+ \models_{\Sigma_1^+, a:A^+, \Sigma_2^+} J$ *and* $\models_{\Sigma_1^+} A^+ =^* B^+ : \mathbf{Type}$*, then* $\Gamma^+ \models_{\Sigma_1^+, a:B^+, \Sigma_2^+} J$.

5. *If* $\Gamma^+, x : A^+, \Delta^+ \models_{\Sigma^+} J$ *and* $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : \mathbf{Type}$*, then* $\Gamma^+, x : B^+, \Delta^+ \models_{\Sigma^+} J$.

Lemma B.24 (Generation) 1. *If* $\Gamma^+ \models_{\Sigma^+} \mathbf{Type} \twoheadrightarrow K^+$*, then* $K^+ \equiv \mathbf{Type}$.

2. *If* $\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K_1^+ \twoheadrightarrow L^+$*, then there exist* A_2^+*,* K_2^+ *such that*

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & K_1^+ \twoheadrightarrow K_2^+ \\ L^+ & \equiv & \Pi x : A_2^+.K_2^+ \end{array}$$

3. *If* $\Gamma^+ \models_{\Sigma^+} a \twoheadrightarrow B^+ : K^+$*, then* $B^+ \equiv c$*, and there exists* L^+ *such that* $a : L^+ \in \Sigma^+$ *and* $\Gamma^+ \models_{\Sigma^+} K^+ =^* L^+$.

4. *If* $\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.B_1^+ \twoheadrightarrow C^+ : K^+$*, then* $K^+ \equiv \mathbf{Type}$*, and there exist* A_2^+*,* B_2^+ *such that*

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type} \\ C^+ & \equiv & \Pi x : A_2^+.B_2^+ \end{array}$$

5. *If* $\Gamma^+ \models_{\Sigma^+} \lambda x : A_1^+.B_1^+ \twoheadrightarrow C^+ : K^+$*, then there exist* A_2^+*,* B_2^+*,* L_1^+*,* L_2^+ *such that*

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & B_1^+ \twoheadrightarrow B_2^+ : L_1^+ \\ \Gamma^+ & \models_{\Sigma^+} & L_1^+ \twoheadrightarrow L_2^+ \\ C^+ & \equiv & \lambda x : A_2^+.B_2^+ \\ \Gamma^+ & \models_{\Sigma^+} & K^+ =^* \Pi x : A_1^+.L_1^+ \end{array}$$

6. *If* $\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)L_1^+} A_1^+ M_1^+ \twoheadrightarrow C^+ : K^+$*, then there exist* M_2^+*,* B_1^+*,* B_2^+*,* L_2^+ *such that*

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type} \\ \Gamma^+, x : B_1^+ & \models_{\Sigma^+} & L_1^+ \twoheadrightarrow L_2^+ \\ \Gamma^+, x : B_1^+ & \models_{\Sigma^+} & M_1^+ \twoheadrightarrow M_2^+ : B_1^+ \\ \Gamma^+ & \models_{\Sigma^+} & K^+ =^* [M_1^+/x]L_1^+ \end{array}$$

Further, either *there exists* A_2^+ *such that*

$$\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \Pi x : B_1^+.L_1^+, \qquad C^+ \equiv \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+)$$

or $A_1^+ \equiv \lambda x : B_1^+.D_1^+$*, and there exists* D_2^+ *such that*

$$\Gamma^+, x : B_1^+ \models_{\Sigma^+} D_1^+ \twoheadrightarrow D_2^+ : L_1^+, \qquad C^+ \equiv [M_2^+/x]D_2^+$$

7. *If* $\Gamma^+ \models_{\Sigma^+} x \twoheadrightarrow N^+ : A^+$*, then* $N^+ \equiv x$*, and there exists* B^+ *such that* $x : B^+ \in \Gamma^+$ *and* $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : \mathbf{Type}$.

8. *If* $\Gamma^+ \models_{\Sigma^+} c \twoheadrightarrow N^+ : A^+$*, then* $N^+ \equiv c$*, and there exists* B^+ *such that* $c : B^+ \in \Sigma^+$ *and* $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : \mathbf{Type}$.

9. *If* $\Gamma^+ \models_{\Sigma^+} \lambda x : A_1^+.M_1^+ \twoheadrightarrow N^+ : B^+$*, then there exist* A_2^+*,* M_2^+*,* C_1^+*,* C_2^+ *such that*

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & C_1^+ \twoheadrightarrow C_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & M_1^+ \twoheadrightarrow M_2^+ : C_1^+ \\ N^+ & \equiv & \lambda x : A_2^+.M_2^+ \\ \Gamma^+ & \models_{\Sigma^+} & B^+ =^* \Pi x : A_1^+.C_1^+ : \mathbf{Type} \end{array}$$

10. *If* $\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)B_1^+} M_1^+ N_1^+ \twoheadrightarrow P^+ : C^+$*, then there exist* N_2^+*,* A_2^+*,* B_1^+*,* B_2^+ *such that*

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type} \\ \Gamma^+ & \models_{\Sigma^+} & N_1^+ \twoheadrightarrow N_2^+ : A_1^+ \\ \Gamma^+ & \models_{\Sigma^+} & C^+ =^* [N_1^+/x]B_1^+ : \mathbf{Type} \end{array}$$

Further, either *there exists* M_2^+ *such that*

$$\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : \Pi x : A_1^+.B_1^+, \qquad P^+ \equiv \mathrm{app}_{(x)B_2^+}(M_2^+ N_2^+)$$

or $M_1^+ \equiv \lambda x : A_1^+.Q_1^+$*, and there exists* Q_2^+ *such that*

$$\Gamma^+, x : A_1^+ \models_{\Sigma^+} Q_1^+ \twoheadrightarrow Q_2^+ : B_1^+, \qquad P^+ \equiv [N_2^+/x]Q_2^+$$

Proof Straightforward induction on the premise in each case. □

Corollary B.25 (Type Validity) *1. If $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K_1^+$, then there exists K_2^+ such that*

$$\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$$

2. If $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, then there exists B^+ such that

$$\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : \mathbf{Type}$$

Proof Inspection of each case of Generation. □

Lemma B.26 (Functionality of Π) *1. If $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* A_2^+ : \mathbf{Type}$, and $\Gamma^+, x : A_1^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$, then $\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K_1^+ =^* \Pi x : A_2^+.K_2^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* A_2^+ : \mathbf{Type}$, and $\Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ =^* B_2^+ : \mathbf{Type}$, then $\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.B_1^+ =^* \Pi x : A_2^+.B_2^+ : \mathbf{Type}$.

Proof

1. We prove the two statements:

   (a) If $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow A^+ : \mathbf{Type}$ and $\Gamma^+, x : A^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$, then $\Gamma^+ \models_{\Sigma^+} \Pi x : A^+.K_1^+ =^* \Pi x : A^+.K_2^+$.

   (b) If $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* A_2^+ : \mathbf{Type}$ and $\Gamma^+, x : A_1^+ \models K^+ \twoheadrightarrow K^+$, then $\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K^+ =^* \Pi x : A_2^+.K^+$.

   . These, together with Reflexivity, suffice to prove the lemma. The proofs are by induction on $\Gamma^+, x : A^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$ in the first case, and $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* A_2^+ : \mathbf{Type}$ in the second.

2. Similar.

□

Lemma B.27 (Uniqueness of Types) *1. If $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K_1^+$ and $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow C^+ : K_2^+$, then $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$ and $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow P^+ : B^+$, then $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : \mathbf{Type}$.

Proof

1. The proof is by induction on A^+, using Generation. We shall deal here only with two cases.

$A^+ \equiv \lambda x : D_1^+.E_1^+$ By Generation, there exist families E_2^+, E_3^+ and kinds L_1^+, M_1^+ such that

$$\Gamma^+, x : D_1^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : L_1^+ \qquad \Gamma^+, x : D_1^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_3^+ : M_1^+$$
$$\Gamma^+ \models_{\Sigma^+} K_1^+ =^* \Pi x : D_1^+.L_1^+ \qquad \Gamma^+ \models_{\Sigma^+} K_2^+ =^* \Pi x : D_1^+.M_1^+$$

By i.h.,

$$\Gamma^+, x : D_1^+ \models_{\Sigma^+} L_1^+ =^* M_1^+$$

The result follows by Functionality of Π.

$A^+ \equiv \mathrm{app}_{(x)L^+}(D^+ M^+)$ By Generation,

$$\begin{array}{lll} \Gamma^+ & \models_{\Sigma^+} & K_1^+ =^* [M^+/x]L^+ \\ \Gamma^+ & \models_{\Sigma^+} & K_2^+ =^* [M^+/x]L^+ \end{array}$$

and the result follows easily.

2. Similar.

□

Theorem B.28 (Church-Rosser) *1. If $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$ and $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_3^+$, then there exists K_4^+ such that $\Gamma^+ \models_{\Sigma^+} K_2^+ \twoheadrightarrow K_4^+$ and $\Gamma^+ \models_{\Sigma^+} K_3^+ \twoheadrightarrow K_4^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K_1^+$ and $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow C^+ : K_2^+$, then there exists D^+ such that

$$\Gamma^+ \models_{\Sigma^+} B^+ \twoheadrightarrow D^+ : K_1^+ \qquad \Gamma^+ \models_{\Sigma^+} C^+ \twoheadrightarrow D^+ : K_1^+$$
$$\Gamma^+ \models_{\Sigma^+} B^+ \twoheadrightarrow D^+ : K_2^+ \qquad \Gamma^+ \models_{\Sigma^+} C^+ \twoheadrightarrow D^+ : K_2^+$$

3. If $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$ and $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow P^+ : B^+$, then there exists Q^+ such that

$$\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow Q^+ : A^+ \qquad \Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow Q^+ : A^+$$
$$\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow Q^+ : B^+ \qquad \Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow Q^+ : B^+$$

Proof We prove all three parts simultaneously, by double induction on the premises.

By Uniqueness of Types, the premises in the third part entail $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : \mathbf{Type}$ so it is sufficient to prove the first two conclusions: $\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow Q^+ : A^+$ and $\Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow Q^+ : A^+$. Similarly for the second part.

If either premise were derived using (red) or (exp), the proof is almost trivial. Suppose, for example, the first were derived by (red):

$$(\mathrm{red})\ \frac{\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : C^+ \quad \Gamma^+ \models_{\Sigma^+} C^+ \twoheadrightarrow A^+ : \mathbf{Type}}{\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+}$$

By i.h., there exists Q such that

$$\begin{array}{rr} \Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow Q^+ : C^+ & \Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow Q^+ : C^+ \\ \Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow Q^+ : B^+ & \Gamma \models P^+ \twoheadrightarrow Q^+ : C^+ \end{array}$$

Applying (red) to the first two gives

$$\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow Q^+ : A^+ \qquad \Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow Q^+ : A^+$$

If both premises were derived using the same rule, the proof is relatively straightforward (if we have Context Conversion to hand), and we omit the details.

There remains the case where one premise was derived using (app), and the other using (beta). We shall deal with this case for objects here; the proof for families is similar.

So we assume the last steps in the two derivations were

$$(\text{beta})\ \frac{\begin{array}{cc} \Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : \mathbf{Type} & \Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : B_1^+ & \Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_1^+ : A_1^+ \end{array}}{\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)B_1^+}((\lambda x : A_1^+ . M_1^+) N_1^+) \twoheadrightarrow [N_2^+/x] M_2^+ : [N_1^+/x] B_1^+}$$

and

$$(\text{lambda})\ \frac{\begin{array}{cc} \Gamma^+ \models_{\Sigma^+} \lambda x : A_1^+ . M_1^+ \twoheadrightarrow P_3^+ : \Pi x : C_1^+ . B_1^+ & \Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_3^+ : C_1^+ \\ \Gamma^+ \models_{\Sigma^+} C_1^+ \twoheadrightarrow C_3^+ : \mathbf{Type} & \Gamma^+, x : C_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_3^+ : \mathbf{Type} \end{array}}{\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)B_1^+}((\lambda x : A_1^+ . M_1^+) N_1^+) \twoheadrightarrow \mathrm{app}_{(x)B_3^+}(P_3^+ N_3^+) : [N_1^+/x] B_1^+}$$

Now, Uniqueness of Types gives $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* C_1^+ : \mathbf{Type}$. Therefore, by Context Conversion, $\Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : \mathbf{Type}$

Applying Generation to the premise $\Gamma^+ \models_{\Sigma^+} \lambda x : A_1^+ . M_1^+ \twoheadrightarrow P_3^+ : \Pi x : C_1^+ . B_1^+$, there exist A_3^+, M_3^+ , D_1^+, D_3^+ such that

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & A_1^+ \twoheadrightarrow A_3^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & M_1^+ \twoheadrightarrow M_3^+ : D_1^+ \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & D_1^+ \twoheadrightarrow D_3^+ : \mathbf{Type} \\ P_3^+ & \equiv & \lambda x : A_3^+ . M_3^+ \\ \Gamma^+ & \models_{\Sigma^+} & \Pi x : C_1^+ . B_1^+ =^* \Pi x : A_1^+ . D_1^+ : \mathbf{Type} \end{array}$$

Now, applying Uniqueness of Types and Context Conversion, we can derive the judgements we need to apply the induction hypothesis:

$$\begin{array}{rcl} \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & M_1^+ \twoheadrightarrow M_3^+ : B_1^+ \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} & B_1^+ \twoheadrightarrow B_3^+ : \mathbf{Type} \\ \Gamma^+ & \models_{\Sigma^+} & N_1^+ \twoheadrightarrow N_3^+ : A_1^+ \end{array}$$

So: the induction hypothesis gives us terms A_0^+, B_0^+, M_0^+, N_0^+ such that

$$\begin{array}{rlrl} \Gamma^+ & \models_{\Sigma^+} A_2^+ \twoheadrightarrow A_0^+ : \mathbf{Type} & \Gamma^+ & \models_{\Sigma^+} A_3^+ \twoheadrightarrow A_0^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} B_2^+ \twoheadrightarrow B_0^+ : \mathbf{Type} & \Gamma^+, x : A_1^+ & \models_{\Sigma^+} B_3^+ \twoheadrightarrow B_0^+ : \mathbf{Type} \\ \Gamma^+, x : A_1^+ & \models_{\Sigma^+} M_2^+ \twoheadrightarrow M_0^+ : B_1^+ & \Gamma^+, x : A_1^+ & \models_{\Sigma^+} M_3^+ \twoheadrightarrow M_0^+ : B_1^+ \\ \Gamma^+ & \models_{\Sigma^+} N_2^+ \twoheadrightarrow N_0^+ : A_1^+ & \Gamma^+ & \models_{\Sigma^+} N_3^+ \twoheadrightarrow N_0^+ : A_1^+ \end{array}$$

Substitution now gives

$$\Gamma^+ \models_{\Sigma^+} [N_2^+/x]M_2^+ \twoheadrightarrow [N_0^+/x]M_0^+ : [N_2^+/x]B_1^+$$

We can derive the judgements

$$\begin{array}{rcl} \Gamma^+, x : A_3^+ & \models_{\Sigma^+} & B_3^+ \twoheadrightarrow B_0^+ : \mathbf{Type} \\ \Gamma^+, x : A_3^+ & \models_{\Sigma^+} & M_3^+ \twoheadrightarrow M_0^+ : B_3^+ \end{array}$$

and (beta) then gives

$$\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)B_3^+}((\lambda x : A_3^+.M_3^+)N_3^+) \twoheadrightarrow [N_0^+/x]M_0^+ : [N_3^+/x]B_3^+$$

It remains only to adjust the types of these judgements. Reflexivity gives

$$\Gamma^+, x : A_1^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_1^+ : \mathbf{Type}$$

Substitution, therefore, gives

$$\Gamma^+ \models_{\Sigma^+} [N_1^+/x]B_1^+ \twoheadrightarrow [N_2^+/x]B_1^+ : \mathbf{Type}$$

as well as

$$\Gamma^+ \models_{\Sigma^+} [N_1^+/x]B_1^+ \twoheadrightarrow [N_3^+/x]B_3^+ : \mathbf{Type}$$

(exp) finally gives the judgements we need:

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & [N_2^+/x]M_2^+ \twoheadrightarrow [N_0^+/x]M_0^+ : [N_1^+/x]B_1^+ \\ \Gamma^+ & \models_{\Sigma^+} & \mathrm{app}_{(x)B_3^+}((\lambda x : A_3^+.M_3^+)N_3^+) \twoheadrightarrow [N_0^+/x]M_0^+ : [N_1^+/x]B_1^+ \end{array}$$

□

Corollary B.29 *1. If $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$, there exists K_3^+ such that $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow^+ K_3^+$ and $\Gamma^+ \models_{\Sigma^+} K_2^+ \twoheadrightarrow^+ K_3^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : K^+$, there exists C^+ such that $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow^+ C^+ : K^+$ and $\Gamma^+ \models_{\Sigma^+} B^+ \twoheadrightarrow^+ C^+ : K^+$.

3. If $\Gamma^+ \models_{\Sigma^+} M^+ =^* N^+ : A^+$, there exists P^+ such that $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow^+ P^+ : A^+$ and $\Gamma^+ \models_{\Sigma^+} N^+ \twoheadrightarrow^+ P^+ : A^+$.

Corollary B.30 (Injectivity of Π) *1. If $\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K_1^+ =^* \Pi x : A_2^+.K_2^+$, then $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* A_2^+ : \mathbf{Type}$ and $\Gamma^+, x : A_1^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} \Pi x : A^+.B^+ =^* \Pi x : C^+.D^+ : \mathbf{Type}$, then $\Gamma^+ \models_{\Sigma^+} A^+ =^* C^+ : \mathbf{Type}$ and $\Gamma^+, x : A^+ \models_{\Sigma^+} B^+ =^* D^+ : \mathbf{Type}$.

Proof

1. By Church-Rosser, there exists a kind L^+ such that

$$\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K_1^+ \twoheadrightarrow^+ L^+, \qquad \Gamma^+ \models_{\Sigma^+} \Pi x : A_2^+.K_2^+ \twoheadrightarrow^+ L^+$$

   Therefore, there are sequences of kinds $L_1^+, \ldots, L_m^+$ and $J_1^+, \ldots, J_n^+$ such that

$$\Gamma^+ \models_{\Sigma^+} \Pi x : A_1^+.K_1^+ \twoheadrightarrow L_1^+,\ \Gamma^+ \models_{\Sigma^+} L_1^+ \twoheadrightarrow L_2^+,\ \cdots\ \Gamma^+ \models_{\Sigma^+} L_m^+ \twoheadrightarrow L^+$$

   and

$$\Gamma^+ \models_{\Sigma^+} \Pi x : A_2^+.K_2^+ \twoheadrightarrow J_1^+,\ \Gamma^+ \models_{\Sigma^+} J_1^+ \twoheadrightarrow J_2^+,\ \cdots\ \Gamma^+ \models_{\Sigma^+} J_n^+ \twoheadrightarrow L^+$$

   By Generation, each K_i^+, J_j^+, and L^+ itself is a Π-term, say

$$K_i^+ \equiv \Pi x : C_i^+.K_{i1}^+, \qquad J_j^+ \equiv \Pi x : D_j^+.J_{j1}^+, \qquad L^+ \equiv \Pi x : C_0^+.L_0^+$$

$$\begin{gathered}\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow C_1^+ \twoheadrightarrow C_2^+ \twoheadrightarrow \cdots \twoheadrightarrow C_0^+ : \mathbf{Type} \\ \Gamma^+ \models_{\Sigma^+} A_2^+ \twoheadrightarrow D_1^+ \twoheadrightarrow D_2^+ \twoheadrightarrow \cdots \twoheadrightarrow C_0^+ : \mathbf{Type} \\ \therefore \Gamma^+ \models_{\Sigma^+} A_1^+ =^* A_2^+ : \mathbf{Type}\end{gathered}$$

   Similarly (using Context Conversion)

$$\Gamma^+, x : A_1^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$$

2. Similar.

□

Theorem B.31 (Subject Reduction) *1. If $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$, and $K_1^+ \to_\beta K_3^+$, then $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_3^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K^+$, and $A^+ \to_\beta C^+$, then $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow C^+ : K^+$.

3. If $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$ and $M^+ \to_\beta P^+$, then $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow P^+ : A^+$.

Proof The three parts are proved simultaneously, by induction on K_1^+, A^+ or M^+, using Generation. Apart from the β-contraction cases, most cases are similar to the following from the third part.

$M^+ \equiv \lambda x : C_1^+.R_1^+, P^+ \equiv \lambda x : C_2^+.R_2^+, C_1^+ \rightarrow_\beta C_2^+, R_1^+ \rightarrow_\beta R_2^+$ By Generation, there exist families C_0^+, B_1^+, B_0^+ and a term R_0^+ such that

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & C_1^+ \twoheadrightarrow C_0^+ : \mathbf{Type} \\ \Gamma^+, x : C_1^+ & \models_{\Sigma^+} & R_1^+ \twoheadrightarrow R_0^+ : B_1^+ \\ \Gamma^+, x : C_1^+ & \models_{\Sigma^+} & B_1^+ \twoheadrightarrow B_0^+ : \mathbf{Type} \\ N^+ & \equiv & \lambda x : C_0^+.R_0^+ \\ \Gamma^+ & \models_{\Sigma^+} & A^+ =^* \Pi x : C_1^+.B_1^+ : \mathbf{Type} \end{array}$$

Therefore, by i.h.,

$$\begin{array}{rcll} \Gamma^+ & \models_{\Sigma^+} & C_1^+ \twoheadrightarrow C_2^+ : \mathbf{Type} & \\ \Gamma^+, x : C_1^+ & \models_{\Sigma^+} & R_1^+ \twoheadrightarrow R_2^+ : B_1^+ & \\ \therefore \Gamma^+ & \models_{\Sigma^+} & \lambda x : C_1^+.R_1^+ \twoheadrightarrow \lambda x : C_2^+.R_2^+ : \Pi x : C_1^+.B_1^+ & \text{(lambda)} \\ \therefore \Gamma^+ & \models_{\Sigma^+} & \lambda x : C_1^+.R_1^+ \twoheadrightarrow \lambda x : C_2^+.R_2^+ : A^+ & \text{((red) and (exp))} \end{array}$$

The β-contraction cases are handled as follows. We deal only with the object case; the family case is similar.

$M^+ \equiv \mathrm{app}_{(x)D_1^+}((\lambda x : C^+.R^+)S_1^+), P^+ \equiv [S^+/x]R^+$ By Generation, there exists an object S_2^+ and families E_1^+, E_2^+, D_2^+,

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & E_1^+ \twoheadrightarrow E_2^+ : \mathbf{Type} \\ \Gamma^+ & \models_{\Sigma^+} & S_1^+ \twoheadrightarrow S_2^+ : E_1^+ \\ \Gamma^+, x : E_1^+ & \models_{\Sigma^+} & D_1^+ \twoheadrightarrow D_2^+ : \mathbf{Type} \\ \Gamma^+ & \models_{\Sigma^+} & A^+ =^* [S_1^+/x]D_1^+ : \mathbf{Type} \end{array}$$

and either there exists T_2^+ such that

$$\Gamma^+ \models_{\Sigma^+} \lambda x : C^+.R^+ \twoheadrightarrow T_2^+ : \Pi x : E_1^+.D_1^+, \quad N^+ \equiv \mathrm{app}_{(x)D_2^+}(T_2^+ S_2^+)$$

or $C^+ \equiv E_1^+$ and there exists R_2^+ such that

$$\Gamma^+, x : C^+ \models_{\Sigma^+} R_1^+ \twoheadrightarrow R_2^+ : D_1^+, \quad N^+ \equiv [S_2^+/x]R_2^+$$

In either case (using Generation in the first case), there exists an object R_2^+ and a family F^+ such that

$$\Gamma^+, x : C^+ \models_{\Sigma^+} R_1^+ \twoheadrightarrow R_2^+ : F^+, \quad \Gamma^+ \models_{\Sigma^+} \Pi x : E^+.D^+ =^* \Pi x : C^+.F^+ : \mathbf{Type}$$

Therefore, by Π-injectivity,

$$\begin{array}{rcll}
\Gamma^+ & \models_{\Sigma^+} & E^+ =^* C^+ : \mathbf{Type} & \\
\Gamma^+, x : C^+ & \models_{\Sigma^+} & D_1^+ =^* F^+ : \mathbf{Type} & \\
\therefore \Gamma^+ & \models_{\Sigma^+} & C^+ \twoheadrightarrow C^+ : \mathbf{Type} \qquad \text{(Reflexivity)} & \\
\Gamma^+, x : C^+ & \models_{\Sigma^+} & R^+ \twoheadrightarrow R^+ : D_1^+ \qquad \text{(Reflexivity)} & \\
\Gamma^+ & \models_{\Sigma^+} & S^+ \twoheadrightarrow S^+ : C^+ \qquad \text{(Reflexivity)} & \\
\therefore \Gamma^+ & \models_{\Sigma^+} & \mathrm{app}_{(x)D_1^+}((\lambda x : C^+ . R^+) S^+) \twoheadrightarrow [S^+/x]R^+ : [R^+/x]D_1^+ & \text{(beta)} \\
\therefore \Gamma^+ & \models & \mathrm{app}_{(x)D_1^+}((\lambda x : C^+ . R^+) S^+) \twoheadrightarrow [S^+/x]R^+ : A^+ &
\end{array}$$

□

B.1.2 The Translation from Typed Reduction to ELF

Lemma B.32 *1. Suppose $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_2^+$, $\Gamma^+ \models_{\Sigma^+} L_1^+ \twoheadrightarrow L_2^+$, and $|K_1^+| \equiv |L_1^+|$. Then there exists J^+ such that*

$$\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow^+ J^+, \qquad \Gamma^+ \models_{\Sigma^+} L_1^+ \twoheadrightarrow J^+$$

2. Suppose $\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow B^+ : K^+$, $\Gamma^+ \models_{\Sigma^+} C^+ \twoheadrightarrow D^+ : L^+$, and $|A^+| \equiv |C^+|$. Then there exists E^+ such that

$$\begin{array}{ll}
\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow^+ E^+ : K^+, & \Gamma^+ \models_{\Sigma^+} C^+ \twoheadrightarrow^+ E^+ : K^+, \\
\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow^+ E^+ : L^+, & \Gamma^+ \models_{\Sigma^+} C^+ \twoheadrightarrow^+ E^+ : L^+
\end{array}$$

3. Suppose $\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow N^+ : A^+$, $\Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow Q^+ : B^+$, and $|M^+| \equiv |P^+|$. Then there exists R^+ such that

$$\begin{array}{ll}
\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow^+ R^+ : A^+ & \Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow^+ R^+ : A^+ \\
\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow^+ R^+ : B^+ & \Gamma^+ \models_{\Sigma^+} P^+ \twoheadrightarrow^+ R^+ : B^+
\end{array}$$

Proof All three parts are similar; we deal only with the third.

The proof is by induction on M^+. The cases where M^+ is a constant or variable are immediate from Church-Rosser, since then $M^+ \equiv P^+$.

Case $M^+ \equiv \lambda x : C_1^+ . S_1^+$: Then $P^+ \equiv \lambda x : C_2^+ . S_2^+$, where $|C_1^+| \equiv |C_2^+|$, $|S_1^+| \equiv |S_2^+|$. By Generation, there exist extended families C_3^+, C_4^+, D_1^+, D_2^+, D_3^+, D_4^+, and extended objects

S_3^+, S_4^+ such that

$$\begin{array}{rlrl}
\Gamma^+ \models_{\Sigma^+} C_1^+ \twoheadrightarrow C_3^+ : \mathbf{Type} & & \Gamma^+ \models_{\Sigma^+} C_2^+ \twoheadrightarrow C_4^+ : \mathbf{Type} \\
\Gamma^+, x : C_1^+ \models_{\Sigma^+} S_1^+ \twoheadrightarrow S_3^+ : D_1^+ & & \Gamma^+, x : C_2^+ \models_{\Sigma^+} S_2^+ \twoheadrightarrow S_4^+ : D_2^+ \\
\Gamma^+, x : C_1^+ \models_{\Sigma^+} D_1^+ \twoheadrightarrow D_3^+ : \mathbf{Type} & & \Gamma^+, x : C_2^+ \models_{\Sigma^+} D_2^+ \twoheadrightarrow D_4^+ : \mathbf{Type} \\
\Gamma^+ \models_{\Sigma^+} A^+ =^* \Pi x : C_1^+ . D_1^+ : \mathbf{Type} & & \Gamma^+ \models_{\Sigma^+} B^+ =^* \Pi x : C_2^+ . D_2^+ : \mathbf{Type}
\end{array}$$

By induction hypothesis, there exists C_0^+ such that

$$\begin{array}{cc}
\Gamma^+ \models_{\Sigma^+} C_1^+ \twoheadrightarrow^+ C_0^+ : \mathbf{Type} \qquad \Gamma^+ \models_{\Sigma^+} C_2^+ \twoheadrightarrow^+ C_0^+ : \mathbf{Type} \\
\therefore \Gamma^+, x : C_1^+ \models_{\Sigma^+} S_2^+ \twoheadrightarrow S_4^+ : D_2^+ \qquad \text{(Context Conversion)} \\
\Gamma^+, x : C_1^+ \models_{\Sigma^+} D_2^+ \twoheadrightarrow D_4^+ : \mathbf{Type} \qquad \text{(Context Conversion)}
\end{array}$$

Therefore, by the induction hypothesis, there exists S_0^+, D_0^+ such that

$$\begin{array}{rll}
\Gamma^+, x : C_1^+ & \models_{\Sigma^+} & S_1^+ \twoheadrightarrow^+ S_0^+ : D_1^+, \\
\Gamma^+, x : C_1^+ & \models_{\Sigma^+} & S_2^+ \twoheadrightarrow^+ S_0^+ : D_2^+ \\
\therefore \Gamma^+, x : C_2^+ & \models_{\Sigma^+} & S_2^+ \twoheadrightarrow^+ S_0^+ : D_2^+ \qquad \text{(Context Conversion)} \\
\Gamma^+, x : C_1^+ & \models_{\Sigma^+} & D_1^+ \twoheadrightarrow^+ D_0^+ : \mathbf{Type} \\
\Gamma^+, x : C_1^+ & \models_{\Sigma^+} & D_2^+ \twoheadrightarrow^+ D_0^+ : \mathbf{Type} \\
\therefore \Gamma^+ & \models_{\Sigma^+} & \lambda x : C_1^+ . S_1^+ \twoheadrightarrow^+ \lambda x : C_0^+ . S_0^+ : \Pi x : C_1^+ . D_1^+ \\
\Gamma^+ & \models_{\Sigma^+} & \lambda x : C_2^+ . S_2^+ \twoheadrightarrow^+ \lambda x : C_0^+ . S_0^+ : \Pi x : C_2^+ . D_2^+ \qquad \text{(lambda)} \\
\Gamma^+ & \models_{\Sigma^+} & \Pi x : C_1^+ . D_1^+ \twoheadrightarrow^+ \Pi x : C_0^+ . D_0^+ : \mathbf{Type} \\
\Gamma^+ & \models_{\Sigma^+} & \Pi x : C_2^+ . D_2^+ \twoheadrightarrow^+ \Pi x : C_0^+ . D_0^+ : \mathbf{Type} \qquad \text{(prod)}
\end{array}$$

Therefore,

$$\begin{array}{c}
\Gamma^+ \models_{\Sigma^+} A^+ =^* \Pi x : C_1^+ . D_1^+ =^* \Pi x : C_2^+ . D_2^+ =^* B^+ : \mathbf{Type} \\
\Gamma^+ \models_{\Sigma^+} \lambda x : C_1^+ . S_1^+ \twoheadrightarrow^+ \lambda x : C_0^+ . S_0^+ : A^+ \qquad \Gamma^+ \models_{\Sigma^+} \lambda x : C_2^+ . S_2^+ \twoheadrightarrow^+ \lambda x : C_0^+ . S_0^+ : A^+ \\
\Gamma^+ \models_{\Sigma^+} \lambda x : C_1^+ . S_1^+ \twoheadrightarrow^+ \lambda x : C_0^+ . S_0^+ : B^+ \qquad \Gamma^+ \models_{\Sigma^+} \lambda x : C_2^+ . S_2^+ \twoheadrightarrow^+ \lambda x : C_0^+ . S_0^+ : B^+
\end{array}$$

Case $M^+ \equiv \mathrm{app}_{(x)D_1^+}(S_1^+ T_1^+)$: Then $P^+ \equiv \mathrm{app}_{(x)D_2^+}(S_2^+ T_2^+)$, where $|S_1^+| \equiv |S_2^+|$ and $|T_1^+| \equiv |T_2^+|$. By Generation, there exists an extended term T_3^+ and extended families D_3^+, C_1^+, C_3^+ such that

$$\begin{array}{rll}
\Gamma^+ & \models_{\Sigma^+} & C_1^+ \twoheadrightarrow C_3^+ : \mathbf{Type} \\
\Gamma^+, x : C_1^+ & \models_{\Sigma^+} & D_1^+ \twoheadrightarrow D_3^+ : \mathbf{Type} \\
\Gamma^+ & \models_{\Sigma^+} & T_1^+ \twoheadrightarrow T_3^+ : C_1^+ \\
\Gamma^+ & \models_{\Sigma^+} & A^+ =^* [T_1^+/x]D_1^+ : \mathbf{Type}
\end{array}$$

and either there exists S_3^+ such that

$$\Gamma^+ \models_{\Sigma^+} S_1^+ \twoheadrightarrow S_3^+ : \Pi x : C_1^+ . D_1^+, \qquad N^+ \equiv \mathrm{app}_{(x)D_3^+}(S_3^+ T_3^+)$$

or $S_1^+ \equiv \lambda x : C_1^+ . U_1^+$, and there exists U_3^+ such that

$$\Gamma^+, x : C_1^+ \models_{\Sigma^+} U_1^+ \twoheadrightarrow U_3^+ : D_1^+, \qquad N^+ \equiv [T_3^+/x]U_3^+$$

In the latter case,

$$\Gamma^+ \models_{\Sigma^+} \lambda x : C_1^+ . U_1^+ \twoheadrightarrow \lambda x : C_3^+ . U_3^+ : \Pi x : C_1^+ . D_1^+$$

so in either case there exists S_3^+ such that

$$\Gamma^+ \models_{\Sigma^+} S_1^+ \twoheadrightarrow S_3^+ : \Pi x : C_1^+ . D_1^+$$

Similarly, there exist C_2^+, C_4^+, D_4^+, S_4^+, T_4^+ such that

$$\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & C_2^+ \twoheadrightarrow C_4^+ : \mathbf{Type} \\ \Gamma^+, x : C_2^+ & \models_{\Sigma^+} & D_2^+ \twoheadrightarrow D_4^+ : \mathbf{Type} \\ \Gamma^+ & \models_{\Sigma^+} & T_2^+ \twoheadrightarrow T_4^+ : C_2^+ \\ \Gamma^+ & \models_{\Sigma^+} & B^+ =^* [T_2^+/x]D_2^+ : \mathbf{Type} \\ \Gamma^+ & \models_{\Sigma^+} & S_2^+ \twoheadrightarrow S_4^+ : \Pi x : C_2^+ . D_2^+ \end{array}$$

Therefore, by induction hypothesis, there exist S_0^+, T_0^+ such that

$$\begin{array}{ll} \Gamma^+ \models_{\Sigma^+} S_1^+ \twoheadrightarrow^+ S_0^+ : \Pi x : C_1^+ . D_1^+ & \Gamma^+ \models_{\Sigma^+} S_2^+ \twoheadrightarrow^+ S_0^+ : \Pi x : C_1^+ . D_1^+ \\ \Gamma^+ \models_{\Sigma^+} S_1^+ \twoheadrightarrow^+ S_0^+ : \Pi x : C_2^+ . D_2^+ & \Gamma^+ \models_{\Sigma^+} S_2^+ \twoheadrightarrow^+ S_0^+ : \Pi x : C_2^+ . D_2^+ \\ \Gamma^+ \models_{\Sigma^+} T_1^+ \twoheadrightarrow^+ T_0^+ : C_1^+ & \Gamma^+ \models_{\Sigma^+} T_2^+ \twoheadrightarrow^+ T_0^+ : C_1^+ \\ \Gamma^+ \models_{\Sigma^+} T_1^+ \twoheadrightarrow^+ T_0^+ : C_2^+ & \Gamma^+ \models_{\Sigma^+} T_2^+ \twoheadrightarrow^+ T_0^+ : C_2^+ \end{array}$$

Hence, by Uniqueness of Types, we have

$$\Gamma^+ \models_{\Sigma^+} \Pi x : C_1^+ . D_1^+ =^* \Pi x : C_2^+ . D_2^+ : \mathbf{Type}, \qquad \Gamma^+ \models_{\Sigma^+} C_1^+ =^* C_2^+ : \mathbf{Type}$$

By Π-injectivity,

$$\Gamma^+, x : C_1^+ \models_{\Sigma^+} D_1^+ =^* D_2^+ : \mathbf{Type}$$

Hence, by Church-Rosser, there exists D_0^+ such that

$$\Gamma^+, x : C_1^+ \models_{\Sigma^+} D_1^+ \twoheadrightarrow^+ D_0^+ : \mathbf{Type}, \qquad \Gamma^+, x : C_1^+ \models_{\Sigma^+} D_2^+ \twoheadrightarrow^+ D_0^+ : \mathbf{Type}$$

Therefore,

$$\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)D_1^+}(S_1^+ T_1^+) \twoheadrightarrow^+ \mathrm{app}_{(x)D_0^+}(S_0^+ T_0^+) : [T_1^+/x]D_1^+,$$

$$\Gamma^+ \models_{\Sigma^+} \mathrm{app}_{(x)D_2^+}(S_2^+ T_2^+) \twoheadrightarrow \mathrm{app}_{(x)D_0^+}(S_0^+ T_0^+) : [T_2^+/x]D_2^+$$

As we also have $\Gamma^+ \models_{\Sigma^+} A^+ =^* [T_1^+/x]D_1^+ =^* [T_2^+/x]D_2^+ =^* B^+ : \mathbf{Type}$, the desired conclusion follows. $\square$

Corollary B.33 *1. If $\Gamma^+ \models_{\Sigma^+} K_1^+ \twoheadrightarrow K_3^+$, $\Gamma^+ \models_{\Sigma^+} K_2^+ \twoheadrightarrow K_4^+$, and $|K_1^+| \equiv |K_2^+|$, then $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$.*

2. If $\Gamma^+ \models_{\Sigma^+} A_1^+ \twoheadrightarrow A_2^+ : K_1^+$, $\Gamma^+ \models_{\Sigma^+} B_1^+ \twoheadrightarrow B_2^+ : K_2^+$, and $|A_1^+| \equiv |B_1^+|$, then $\Gamma^+ \models_{\Sigma^+} K_1^+ =^* K_2^+$, $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* B_1^+ : K_1^+$ and $\Gamma^+ \models_{\Sigma^+} A_1^+ =^* B_1^+ : K_2^+$.

3. If $\Gamma^+ \models_{\Sigma^+} M_1^+ \twoheadrightarrow M_2^+ : A^+$, $\Gamma^+ \models_{\Sigma^+} N_1^+ \twoheadrightarrow N_2^+ : B^+$, and $|M_1^+| \equiv |N_1^+|$, then $\Gamma^+ \models_{\Sigma^+} A^+ =^* B^+ : \mathbf{Type}$, $\Gamma^+ \models_{\Sigma^+} M_1^+ =^* N_1^+ : A^+$, and $\Gamma^+ \models_{\Sigma^+} M_1^+ =^* N_1^+ : B^+$.

Proof From the previous theorem and Uniqueness of Types. $\square$

Corollary B.34 *1. If $\Gamma^+ \models_{\Sigma^+} J$, $\Delta^+ \models_{\Sigma^+}$ valid, and $|\Gamma^+| \equiv |\Delta^+|$, then $\Delta^+ \models_{\Sigma^+} J$.*

2. If $\Gamma^+ \models_{\Sigma_1^+} J$, $\Sigma_2^+ \models$ sig, and $|\Sigma_1^+| \equiv |\Sigma_2^+|$, then $\Gamma^+ \models_{\Sigma_2^+} J$.

Proof By the previous corollary and Context Conversion. $\square$

B.1.3 The Translation from ELF to Typed Reduction

Theorem B.35 *1. If Σ sig is derivable in ELF, then there exists Σ^+ such that*

$$|\Sigma^+| \equiv \Sigma$$

and

$$\Sigma^+ \models \mathrm{sig}$$

2. If $\Gamma \vdash_\Sigma \cdot$, then there exist Γ^+, Σ^+ such that

$$\begin{array}{rcl} |\Gamma^+| & \equiv & \Gamma \\ |\Sigma^+| & \equiv & \Sigma \end{array}$$

and

$$\Gamma^+ \models_{\Sigma^+} \mathrm{valid}$$

3. If $\Gamma \vdash_\Sigma K$, then there exist Γ^+, Σ^+, K^+ such that

$$\begin{array}{rcl} |\Gamma^+| & \equiv & \Gamma \\ |\Sigma^+| & \equiv & \Sigma \\ |K^+| & \equiv & K \end{array}$$

and

$$\Gamma^+ \models_{\Sigma^+} K^+ \twoheadrightarrow K^+$$

4. *Ifd* $\Gamma \vdash_\Sigma A : K$*, then there exist* Γ^+*,* Σ^+*,* A^+*,* K^+ *such that*

$$\begin{array}{rcl} |\Gamma^+| & \equiv & \Gamma \\ |\Sigma^+| & \equiv & \Sigma \\ |A^+| & \equiv & A \\ |K^+| & \equiv & K \end{array}$$

and

$$\Gamma^+ \models_{\Sigma^+} A^+ \twoheadrightarrow A^+ : K^+$$

5. *If* $\Gamma \vdash_\Sigma M : A$ *then there exist* Γ^+*,* Σ^+*,* M^+*,* A^+ *such that*

$$\begin{array}{rcl} |\Gamma^+| & \equiv & \Gamma \\ |\Sigma^+| & \equiv & \Sigma \\ |M^+| & \equiv & M \\ |A^+| & \equiv & A \end{array}$$

and

$$\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow M^+ : A^+$$

Proof Simultaneous, by induction on the premise. We deal here with the two most difficult cases for families; the corresponding cases for objects are similar, and all other cases are easy.

(`B-APP-FAM`)

$$\frac{\Gamma \vdash_\Sigma M : \Pi x : A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : [N/x]B}$$

By i.h., there exist Γ_1^+, Σ_1^+, M^+, A_1^+, B^+, Γ_2^+, Σ_2^+, N^+, A_2^+ such that $|\Gamma_1^+| \equiv |\Gamma_2^+| \equiv \Gamma$, $|\Sigma_1^+| \equiv |\Sigma_2^+| \equiv \Sigma$, $|M^+| \equiv M$, $|A_1^+| \equiv |A_2^+| \equiv A$, $|B^+| \equiv B$, $|N^+| \equiv N$, and

$$\begin{array}{lll} \Gamma_1^+ & \models_{\Sigma_1^+} & M^+ \twoheadrightarrow M^+ : \Pi x : A_1^+.B^+ \\ \Gamma_2^+ & \models_{\Sigma_2^+} & N^+ \twoheadrightarrow N^+ : A_2^+ \end{array}$$

By Corollary B.34 (and Context Validity),

$$\Gamma_1^+ \models_{\Sigma_1^+} N^+ \twoheadrightarrow N^+ : A_2^+$$

Now, by Type Validity and Generation,

$$\begin{array}{lll} \Gamma_1^+ & \models_{\Sigma_1^+} & A_1^+ \twoheadrightarrow ? : \mathbf{Type} \\ \Gamma_1^+ & \models_{\Sigma_1^+} & A_2^+ \twoheadrightarrow ? : \mathbf{Type} \end{array}$$

Therefore, by Corollary B.33,

$$\begin{array}{rll} \Gamma_1^+ & \models_{\Sigma_1^+} & A_1^+ =^* A_2^+ : \mathbf{Type} \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & N^+ \twoheadrightarrow N^+ : A_1^+ \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & \mathrm{app}_{(x)B^+}(M^+N^+) \twoheadrightarrow \mathrm{app}_{(x)B^+}(M^+N^+) : [N^+/x]B^+ \end{array}$$

(B-CONV-FAM)

$$\frac{\Gamma \vdash_\Sigma A : K \quad \Gamma \vdash_\Sigma K'}{\Gamma \vdash_\Sigma A : K'} \, (K \simeq_\beta K')$$

By i.h., there exist Γ_1^+, Σ_1^+, A^+, K_1^+, Γ_2^+, Σ_2^+, K_2^+ such that $|\Gamma_1^+| \equiv |\Gamma_2^+| \equiv \Gamma$, $|\Sigma_1^+| \equiv |\Sigma_2^+| \equiv \Sigma$, $|A^+| \equiv A$, $|K_1^+| \equiv K$, $|K_2^+| \equiv K'$, and

$$\begin{array}{rll} \Gamma_1^+ & \models_{\Sigma_1^+} & A^+ \twoheadrightarrow A^+ : K_1^+ \\ \Gamma_2^+ & \models_{\Sigma_2^+} & K_2^+ \twoheadrightarrow K_2^+ \end{array}$$

Corollary B.34 gives

$$\Gamma_1^+ \models_{\Sigma_1^+} K_2^+ \twoheadrightarrow K_2^+$$

By Church-Rosser, there exists a kind K_0 such that $K \twoheadrightarrow_\beta K_0$ and $K' \twoheadrightarrow_\beta K_0$. Mimicking the reduction sequence in extended kinds, there exist K_3^+, K_4^+ such that

$$K_1^+ \twoheadrightarrow_\beta K_3^+, \qquad K_2^+ \twoheadrightarrow_\beta K_4^+$$

and $|K_3^+| \equiv |K_4^+| \equiv K_0$. So:

$$\begin{array}{rlll} \Gamma_1^+ & \models_{\Sigma_1^+} & K_1^+ \twoheadrightarrow ? & \text{(Type Validity)} \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & K_1^+ \twoheadrightarrow^+ K_3^+ & \text{(Subject Reduction)} \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & K_3^+ \twoheadrightarrow K_3^+ & \text{(Reflexivity)} \\ \Gamma_1^+ & \models_{\Sigma_1^+} & K_2^+ \twoheadrightarrow^+ K_4^+ & \text{(Subject Reduction)} \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & K_4^+ \twoheadrightarrow K_4^+ & \text{(Reflexivity)} \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & K_3^+ =^* K_4^+ & \text{(Corollary B.33)} \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & K_1^+ =^* K_2^+ & \\ \therefore \Gamma_1^+ & \models_{\Sigma_1^+} & A^+ \twoheadrightarrow A^+ : K_2^+ & \end{array}$$

□

Theorem B.36 *1. If Σ sig is derivable in ELF, it is derivable in ELF$_=$.*

2. If $\Gamma \vdash_\Sigma \cdot$, then $\Gamma \vdash_\Sigma^= \cdot$.

3. If $\Gamma \vdash_\Sigma K$, then $\Gamma \vdash_\Sigma^= K$.

4. If $\Gamma \vdash_\Sigma K$, $\Gamma \vdash_\Sigma K'$, and $K \simeq_\beta K'$, then $\Gamma \vdash_\Sigma^= K = K'$.

5. *If* $\Gamma \vdash_\Sigma A : K$*, then* $\Gamma \vdash^{=}_{\Sigma} A : K$*.*

6. *If* $\Gamma \vdash_\Sigma A : K$*,* $\Gamma \vdash_\Sigma B : K$*, and* $A \simeq_\beta B$*, then* $\Gamma \vdash^{=}_{\Sigma} A = B : K$*.*

7. *If* $\Gamma \vdash_\Sigma M : A$*, then* $\Gamma \vdash^{=}_{\Sigma} M : A$*.*

8. *If* $\Gamma \vdash_\Sigma M : A$*,* $\Gamma \vdash_\Sigma N : A$*, and* $M \simeq_\beta N$*, then* $\Gamma \vdash^{=}_{\Sigma} M = N : A$*.*

Proof We deal with cases 7 and 8; each case is similar to one of these.

7. By Theorem B.35, there exist Γ^+, Σ^+, M^+, A^+ such that $|\Gamma^+| \equiv \Gamma$, $|\Sigma^+| \equiv \Sigma$, $|M^+| \equiv M$, $|A^+| \equiv A$, and

$$\begin{array}{lll} \Gamma^+ & \models_{\Sigma^+} & M^+ \twoheadrightarrow M^+ : A^+ \\ \therefore \Gamma & \vdash^{=}_{\Sigma} & M = M : A \qquad \text{(Theorem B.10)} \\ \therefore \Gamma & \vdash^{=}_{\Sigma} & M : A \qquad \text{(Equation Validity)} \end{array}$$

8. By Church-Rosser, there exists P such that $M \twoheadrightarrow_\beta P$ and $N \twoheadrightarrow_\beta P$. By Theorem B.35, there exist Γ^+, Σ^+, M^+, A^+ such that $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$, $|A^+| \equiv A$, and

$$\Gamma^+ \models_{\Sigma^+} M^+ \twoheadrightarrow M^+ : A^+$$

Aping the reduction from M to P in extended objects, there is an extended object P^+ such that $|P^+| \equiv P$ and $M^+ \twoheadrightarrow_\beta P^+$.

$$\begin{array}{lll} \therefore \Gamma^+ & \models_{\Sigma^+} & M^+ \twoheadrightarrow^+ P^+ : A^+ \qquad \text{(Subject Reduction)} \\ \therefore \Gamma & \vdash^{=}_{\Sigma} & M = P : A \qquad \text{(Theorem B.10 and (}\texttt{B-TRANS-OBJ}\text{))} \end{array}$$

Similarly,

$$\begin{array}{lll} \Gamma & \vdash^{=}_{\Sigma} & N = P : A \\ \therefore \Gamma & \vdash^{=}_{\Sigma} & M = N : A \qquad \text{((}\texttt{B-SYM-OBJ}\text{) and (}\texttt{B-TRANS-OBJ}\text{))} \end{array}$$

□

Theorem B.37 1. Σ sig *is derivable in ELF*$_=$ *iff it is derivable in ELF.*

2. $\Gamma \vdash^{=}_{\Sigma} \cdot$ *iff* $\Gamma \vdash_\Sigma \cdot$*.*

3. $\Gamma \vdash^{=}_{\Sigma} K$ *iff* $\Gamma \vdash_\Sigma K$*.*

4. $\Gamma \vdash^{=}_{\Sigma} K = K'$ *iff* $\Gamma \vdash_\Sigma K$*,* $\Gamma \vdash_\Sigma K'$ *and* $K \simeq_\beta K'$*.*

5. $\Gamma \vdash^{=}_{\Sigma} A : K$ *iff* $\Gamma \vdash_\Sigma A : K$*.*

6. $\Gamma \vdash^{=}_{\Sigma} A = B : K$ *iff* $\Gamma \vdash_\Sigma A : K$*,* $\Gamma \vdash_\Sigma B : K$*, and* $A \simeq_\beta B$*.*

7. $\Gamma \vdash^{=}_{\Sigma} M : A$ *iff* $\Gamma \vdash_\Sigma M : A$*.*

8. $\Gamma \vdash^{=}_{\Sigma} M = N : A$ *iff* $\Gamma \vdash_\Sigma M : A$*,* $\Gamma \vdash_\Sigma N : A$ *and* $M \simeq_\beta N$*.*

Proof From Theorems B.8 and B.36. □

B.1.4 Metatheory of ELF$_=$

From this correspondence, we can deduce the following results in the metatheory of ELF$_=$ from the fact that they hold in ELF.

Lemma B.38 (Subject Reduction) *1. If $\Gamma \vdash^{=}_{\Sigma} M : A$ and $M \twoheadrightarrow_\beta N$, then $\Gamma \vdash_= M = N : A$.*

2. If $\Gamma \vdash^{=}_{\Sigma} A : K$ and $A \twoheadrightarrow_\beta B$, then $\Gamma \vdash_= A = B : K$.

3. If $\Gamma \vdash^{=}_{\Sigma} K$ and $K \twoheadrightarrow_\beta K'$, then $\Gamma \vdash_= K = K'$.

Proof

1. By Theorem B.37, $\Gamma \vdash_\Sigma M : A$. Therefore, by Subject Reduction in ELF, $\Gamma \vdash_\Sigma N : A$. Certainly $M \simeq_\beta N$; therefore, $\Gamma \vdash^{=}_{\Sigma} M = N : A$ by Theorem B.37.

Parts 2 and 3 are proved similarly. □

The proofs of these other two lemmas proceed similarly, or can be deduced quite simply from the previous lemma and Predicate Validity:

Lemma B.39 (Predicate Reduction) *1. If $\Gamma \vdash^{=}_{\Sigma} M : A$ and $A \twoheadrightarrow_\beta B$, then $\Gamma \vdash_= M : B$.*

2. If $\Gamma \vdash^{=}_{\Sigma} A : K$ and $K \twoheadrightarrow_\beta K'$, then $\Gamma \vdash_= A : K'$.

Lemma B.40 (Predicate Expansion) *1. If $\Gamma \vdash^{=}_{\Sigma} M : A$, $\Gamma \vdash^{=}_{\Sigma} B :$ **Type** and $B \twoheadrightarrow_\beta A$, then $\Gamma \vdash^{=}_{\Sigma} M : B$.*

2. If $\Gamma \vdash^{=}_{\Sigma} A : K$, $\Gamma \vdash^{=}_{\Sigma} K'$ and $K' \twoheadrightarrow_\beta K$, then $\Gamma \vdash^{=}_{\Sigma} A : K'$.

B.2 The AUTOMATH Systems

The versions of PAL and AUT-68 with judgemental equality were given in Chapter 8. The versions without are as follows:

B.2.1 PAL

Grammar

Expression	E	$::=$	$x \mid c(E, \ldots, E)$
Type	T	$::=$	$E \mid$ **Type**
Signature Declaration	γ	$::=$	$(\Gamma, c, \mathtt{PN}, T)$ of $c \mid (\Gamma, c, E, T)$ of c
Context Declaration	δ	$::=$	$x : T$ of x
Judgement	J	$::=$	$\mathtt{OK} \mid E : T$

Convertibility We define the relation of convertibility relative to a given signature Σ, $\simeq_\Sigma$, to be the least congruence such that, if $(\langle x_1 : T_1, \ldots, x_n : T_n\rangle, c, E, T)$ is in Σ, then

$$c(E_1, \ldots, E_n) \simeq [E_1/x_1, \ldots, E_n/x_n]E$$

Rules of Deduction

$$\frac{}{\langle\rangle \text{ sig}} \qquad \frac{\Gamma \vdash_\Sigma \texttt{OK}}{\Sigma, (\Gamma, c, \texttt{PN}, \textbf{Type}) \text{ sig}} (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E : \textbf{Type}}{\Sigma, (\Gamma, c, \texttt{PN}, E) \text{ sig}} (c \notin \operatorname{dom} \Sigma) \qquad \frac{\Gamma \vdash_\Sigma E : T}{\Sigma, (\Gamma, c, E, T) \text{ sig}} (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Sigma \text{ sig}}{\vdash_\Sigma \texttt{OK}} \qquad \frac{\Gamma \vdash_\Sigma \texttt{OK}}{\Gamma, x : \textbf{Type} \vdash_\Sigma \texttt{OK}} (x \notin \operatorname{dom} \Gamma) \qquad \frac{\Gamma \vdash_\Sigma E : \textbf{Type}}{\Gamma, x : E \vdash_\Sigma \texttt{OK}} (x \notin \operatorname{dom} \Gamma)$$

$$\frac{\Gamma \vdash_\Sigma \texttt{OK}}{\Gamma \vdash_\Sigma x : T} (x : T \in \Gamma)$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E}/\operatorname{dom} \Delta]T} ((\Delta, c, \texttt{PN}, T) \in \Sigma) \qquad \frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E}/\operatorname{dom} \Delta]T} ((\Delta, c, E, T) \in \Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E_1 : E_2 \quad \Gamma \vdash_\Sigma E_3 : \textbf{Type}}{\Gamma \vdash_\Sigma E_1 : E_3} (E_2 \simeq_\Sigma E_3)$$

B.2.2 AUT-68

Grammar

Expression	E	$::=$	$x \mid c(E, \ldots, E) \mid [x : E]E \mid \langle E\rangle E$
Type	T	$::=$	$E \mid \textbf{Type}$
Signature Declaration	γ	$::=$	$(\Gamma, c, \texttt{PN}, T)$ of $c \mid (\Gamma, c, E, T)$ of c
Context Declaration	δ	$::=$	$x : T$ of x
Judgement	J	$::=$	$\texttt{OK} \mid E : T$

Reductions Define the notion of convertibility relative to a given signature Σ, $\simeq_\Sigma$, to be the least congruence such that

$$\langle E_1\rangle([x : E_2]E_3) \rightsquigarrow [E_1/x]E_3$$

and, if (Γ, c, E', T) is in Σ,

$$c(\vec{E}) \rightsquigarrow [\vec{E}/\operatorname{dom} \Gamma]E'$$

Rules of Deduction

$$\frac{}{\langle\rangle \text{ sig}} \qquad \frac{\Gamma \vdash_\Sigma \texttt{OK}}{\Sigma, (\Gamma, c, \texttt{PN}, \textbf{Type}) \text{ sig}} (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E : \textbf{Type}}{\Sigma, (\Gamma, c, \texttt{PN}, E) \text{ sig}} (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Gamma \vdash_\Sigma E : T}{\Sigma, (\Gamma, c, E, T) \text{ sig}} \, (c \notin \operatorname{dom} \Sigma)$$

$$\frac{\Sigma \text{ sig}}{\Gamma \vdash_\Sigma \mathtt{OK}} \qquad \frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Gamma, x : \mathbf{Type} \vdash_\Sigma \mathtt{OK}} \, (x \notin \operatorname{dom} \Gamma) \qquad \frac{\Gamma \vdash_\Sigma E : \mathbf{Type}}{\Gamma, x : E \vdash_\Sigma \mathtt{OK}} \, (x \notin \operatorname{dom} \Gamma)$$

$$\frac{\Gamma \vdash_\Sigma \mathtt{OK}}{\Gamma \vdash_\Sigma x : T} \, (x : T \in \Gamma) \qquad \frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E} / \operatorname{dom} \Delta] T} \, ((\Delta, c, \mathtt{PN}, T) \in \Sigma)$$

$$\frac{\Gamma \Vdash_\Sigma \vec{E} :: \Delta}{\Gamma \vdash_\Sigma c(\vec{E}) : [\vec{E} / \operatorname{dom} \Delta] T} \, ((\Delta, c, E', T) \in \Sigma)$$

$$\frac{\Gamma, x : E_1 \vdash_\Sigma E_2 : \mathbf{Type}}{\Gamma \vdash_\Sigma [x : E_1] E_2 : \mathbf{Type}} \qquad \frac{\Gamma, x : E_1 \vdash_\Sigma E_2 : E_3}{\Gamma \vdash_\Sigma [x : E_1] E_2 : [x : E_1] E_3}$$

$$\frac{\Gamma \vdash_\Sigma E_1 : [x : E_2] E_3 \quad \Gamma \vdash_\Sigma E_4 : E_2}{\Gamma \vdash_\Sigma \langle E_4 \rangle E_1 : [E_4 / x] E_3}$$

$$\frac{\Gamma \vdash_\Sigma E_1 : E_2 \quad \Gamma \vdash_\Sigma E_3 : \mathbf{Type}}{\Gamma \vdash_\Sigma E_1 : E_3} \, (E_2 \simeq_\Sigma E_3)$$

The sets of extended expressions, types, etc. are defined as follows:

PAL

Extended Expression	E^+	$::=$	$x \mid c_{(x,\ldots,x)T^+}(E^+, \ldots, E^+)$
Extended Type	T^+	$::=$	$\mathbf{Type} \mid E^+$
Extended Signature Declaration	γ^+	$::=$	$(\Gamma^+, c, \mathtt{PN}, T^+)$ of c $\mid$ (Γ^+, c, E^+, T^+) of c
Extended Context Declaration	δ^+	$::=$	$x : T^+$ of x

AUT-68

Extended Expression	E^+	$::=$	$x \mid c_{(x,\ldots,x)T^+}(E^+, \ldots, E^+) \mid [x : E^+]E^+$ $\mid \langle E^+ \rangle_{(x)E^+} E^+$
Extended Type	T^+	$::=$	$E^+ \mid \mathbf{Type}$
Extended Signatrue Declaration	γ^+	$::=$	$(\Gamma^+, c, \mathtt{PN}, T^+)$ of c $\mid$ (Γ^+, c, E^+, T^+) of c
Extended Context Declaration	δ^+	$::=$	$x : T^+$ of x

The relation of typed parallel one-step reduction for PAL is defined as in Figures B.3 and B.4. For AUT-68, the definition consists of the same rules together with those in Figure B.5.

With these definitions, we are able to prove:

Theorem B.41 *1.* Σ sig *is derivable in PAL$_=$ iff it is derivable in PAL.*

2. $\Gamma \vdash_\Sigma \mathtt{OK}$ *is derivable in PAL$_=$ iff it is derivable in PAL.*

3. $\Gamma \vdash_\Sigma E : T$ *is derivable in PAL$_=$ iff it is derivable in PAL.*

4. $\Gamma \vdash_\Sigma E_1 = E_2 : T$ *is derivable in PAL$_=$ iff* $\Gamma \vdash_\Sigma E_1 : T$ *and* $\Gamma \vdash_\Sigma E_2 : T$ *are derivable in PAL, and* $E_1 \simeq_\Sigma E_2$.

$$\frac{}{\langle\rangle \models \ \mathrm{sig}} \qquad \frac{\Gamma^+ \models_{\Sigma^+} T_1^+ \twoheadrightarrow T_2^+}{\Sigma^+, (\Gamma^+, c, \mathtt{PN}, T_1^+) \models \ \mathrm{sig}}\,(c \notin \mathrm{dom}\,\Sigma^+)$$

$$\frac{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : T^+}{\Sigma^+, (\Gamma^+, c, E_1^+, T^+) \models \ \mathrm{sig}}\,(c \notin \mathrm{dom}\,\Sigma^+)$$

$$\frac{\Sigma^+ \models \ \mathrm{sig}}{\models_{\Sigma^+} \mathrm{valid}} \qquad \frac{\Gamma^+ \models_{\Sigma^+} T_1^+ \twoheadrightarrow T_2^+}{\Gamma^+, x : T_1^+ \models_{\Sigma^+} \mathrm{valid}}\,(x \notin \mathrm{dom}\,\Gamma) \qquad \frac{\Gamma^+ \models_{\Sigma^+} \mathrm{valid}}{\Gamma^+ \models_{\Sigma^+} x \twoheadrightarrow x : T^+}\,(x : T^+ \in \Gamma)$$

$$\frac{\begin{array}{rcl}
\Gamma^+ & \models_{\Sigma^+} & E_1^+ \twoheadrightarrow F_1^+ : T_1^+ \\
\Gamma^+ & \models_{\Sigma^+} & E_2^+ \twoheadrightarrow F_2^+ : [E_1^+/x_1]T_2^+ \\
 & \vdots & \\
\Gamma^+ & \models_{\Sigma^+} & E_n^+ \twoheadrightarrow F_n^+ : [E_1^+/x_1, \ldots, E_{n-1}^+/x_{n-1}]T_n^+ \\
\Gamma^+ & \models_{\Sigma^+} & T_1^+ \twoheadrightarrow U_1^+ \\
\Gamma^+, x_1 : T_1^+ & \models_{\Sigma^+} & T_2^+ \twoheadrightarrow U_2^+ \\
 & \vdots & \\
\Gamma^+, x_1 : T_1^+, \ldots, x_{n-1} : T_{n-1}^+ & \models_{\Sigma^+} & T_n^+ \twoheadrightarrow U_n^+ \\
\Gamma^+, x_1 : T_1^+, \ldots, x_n : T_n^+ & \models_{\Sigma^+} & T^+ \twoheadrightarrow U^+ \\
\multicolumn{3}{l}{((\langle x_1 : T_1^+, \ldots, x_n : T_n^+\rangle, c, E^+, T^+) \in \Sigma)}
\end{array}}{\Gamma^+ \models_{\Sigma^+} c_{(x_1,\ldots,x_n)T^+}(E_1^+, \ldots, E_n^+) \twoheadrightarrow c_{(x_1,\ldots,x_n)U^+}(F_1^+, \ldots, F_n^+) : [E_1^+/x_1, \ldots, E_n^+/x_n]T^+}$$

$$\frac{\begin{array}{rcl}
\Gamma^+ & \models_{\Sigma^+} & E_1^+ \twoheadrightarrow F_1^+ : T_1^+ \\
\Gamma^+ & \models_{\Sigma^+} & E_2^+ \twoheadrightarrow F_2^+ : [E_1^+/x_1]T_2^+ \\
 & \vdots & \\
\Gamma^+ & \models_{\Sigma^+} & E_n^+ \twoheadrightarrow F_n^+ : [E_1^+/x_1, \ldots, E_{n-1}^+/x_{n-1}]T_n^+ \\
\Gamma^+ & \models_{\Sigma^+} & T_1^+ \twoheadrightarrow U_1^+ \\
\Gamma^+, x_1 : T_1^+ & \models_{\Sigma^+} & T_2^+ \twoheadrightarrow U_2^+ \\
 & \vdots & \\
\Gamma^+, x_1 : T_1^+, \ldots, x_{n-1} : T_{n-1}^+ & \models_{\Sigma^+} & T_n^+ \twoheadrightarrow U_n^+ \\
\Gamma^+, x_1 : T_1^+, \ldots, x_n : T_n^+ & \models_{\Sigma^+} & T^+ \twoheadrightarrow U^+ \\
\multicolumn{3}{l}{(((\langle x_1 : T_1^+, \ldots, x_n : T_n^+\rangle, c, E^+, T^+) \in \Sigma)}
\end{array}}{\Gamma^+ \models_{\Sigma^+} c_{(x_1,\ldots,x_n)T^+}(E_1^+, \ldots, E_n^+) \twoheadrightarrow c_{(x_1,\ldots,x_n)U^+}(F_1^+, \ldots, F_n^+) : [E_1^+/x_1, \ldots, E_n^+/x_n]T^+}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} \mathrm{valid}}{\Gamma^+ \models_{\Sigma^+} \mathbf{Type} \twoheadrightarrow \mathbf{Type}} \qquad \frac{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : \mathbf{Type}}{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+}$$

Figure B.3: Typed Parallel One-Step Reduction for PAL

$$\frac{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : T_1^+ \quad \Gamma^+ \models_{\Sigma^+} T_1^+ \twoheadrightarrow T_2^+}{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : T_2^+} \qquad \frac{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : T_2^+ \quad \Gamma^+ \models_{\Sigma^+} T_1^+ \twoheadrightarrow T_2^+}{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : T_1^+}$$

$$\frac{\begin{array}{rcl} \Gamma^+ & \models_{\Sigma^+} & E_1^+ \twoheadrightarrow F_1^+ : T_1^+ \\ \Gamma^+ & \models_{\Sigma^+} & E_2^+ \twoheadrightarrow F_2^+ : [E_1^+/x_1]T_2^+ \\ & \vdots & \\ \Gamma^+ & \models_{\Sigma^+} & E_n^+ \twoheadrightarrow F_n^+ : [E_1^+/x_1, \ldots, E_{n-1}^+/x_{n-1}]T_n^+ \\ \Gamma^+ & \models_{\Sigma^+} & T_1^+ \twoheadrightarrow U_1^+ \\ \Gamma^+, x_1 : T_1^+ & \models_{\Sigma^+} & T_2^+ \twoheadrightarrow U_2^+ \\ & \vdots & \\ \Gamma^+, x_1 : T_1^+, \ldots, x_{n-1} : T_{n-1}^+ & \models_{\Sigma^+} & T_n^+ \twoheadrightarrow U_n^+ \\ \Gamma^+, x_1 : T_1^+, \ldots, x_n : T_n^+ & \models_{\Sigma^+} & T^+ \twoheadrightarrow U^+ \\ ((\langle x_1 : T_1^+, \ldots, x_n : T_n^+ \rangle, c, E^+, T^+) \in \Sigma) & & \end{array}}{\Gamma^+ \models_{\Sigma^+} c_{(x_1,\ldots,x_n)T^+}(E_1^+, \ldots, E_n^+) \twoheadrightarrow [F_1^+/x_1, \ldots, F_n^+/x_n]E^+ : [E_1^+/x_1, \ldots, E_n^+/x_n]T^+}$$

Figure B.4: Typed Parallel One-Step Reduction for PAL

$$\frac{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : \mathbf{Type} \quad \Gamma^+, x : E_1^+ \models_{\Sigma^+} F_1^+ \twoheadrightarrow F_2^+ : \mathbf{Type}}{\Gamma^+ \models_{\Sigma^+} [x : E_1^+]F_1^+ \twoheadrightarrow [x : E_2^+]F_2^+ : \mathbf{Type}}$$

$$\frac{\Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : \mathbf{Type} \quad \Gamma^+, x : E_1^+ \models_{\Sigma^+} F_1^+ \twoheadrightarrow F_2^+ : G^+}{\Gamma^+ \models_{\Sigma^+} [x : E_1^+]F_1^+ \twoheadrightarrow [x : E_2^+]F_2^+ : [x : E_1^+]G^+}$$

$$\frac{\begin{array}{cc} \Gamma^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : [x : G_1^+]H_1^+ & \Gamma^+ \models_{\Sigma^+} F_1^+ \twoheadrightarrow F_2^+ : G_1^+ \\ \Gamma^+ \models_{\Sigma^+} G_1^+ \twoheadrightarrow G_2^+ & \\ & \Gamma^+, x : G_1^+ \models_{\Sigma^+} H_1^+ \twoheadrightarrow H_2^+ \end{array}}{\Gamma^+ \models_{\Sigma^+} \langle F_1^+ \rangle_{(x)H_1^+} E_1^+ \twoheadrightarrow \langle F_2^+ \rangle_{(x)H_2^+} E_2^+ : [F_1^+/x]H_1^+}$$

$$\frac{\begin{array}{cc} \Gamma^+, x : G_1^+ \models_{\Sigma^+} E_1^+ \twoheadrightarrow E_2^+ : H_1^+ & \Gamma^+ \models_{\Sigma^+} F_1^+ \twoheadrightarrow F_2^+ : G_1^+ \\ \Gamma^+ \models_{\Sigma^+} G_1^+ \twoheadrightarrow G_2^+ & \Gamma^+, x : G_1^+ \models_{\Sigma^+} H_1^+ \twoheadrightarrow H_2^+ \end{array}}{\Gamma^+ \models_{\Sigma^+} \langle F_1^+ \rangle_{(x)H_1^+} [x : G_1^+]E_1^+ \twoheadrightarrow [F_2^+/x]E_2^+ : [F_1^+/x]H_1^+}$$

Figure B.5: Typed Parallel One-Step Reduction for AUT-68

5. Σ sig *is derivable in AUT-68*$_=$ *iff it is derivable in AUT-68.*

6. $\Gamma \vdash_\Sigma \mathtt{OK}$ *is derivable in AUT-68*$_=$ *iff it is derivable in AUT-68.*

7. $\Gamma \vdash_\Sigma E : T$ *is derivable in AUT-68*$_=$ *iff it is derivable in AUT-68.*

8. $\Gamma \vdash_\Sigma E_1 = E_2 : T$ *is derivable in AUT-68*$_=$ *iff* $\Gamma \vdash_\Sigma E_1 : T$ *and* $\Gamma \vdash_\Sigma E_2 : T$ *are derivable in AUT-68, and* $E_1 \simeq_\Sigma E_2$.

Proof Exactly similar to the proof for ELF. $\square$

Appendix C

Predicate Logic in ELF

For reference, we briefly reproduce here the details of the representation of natural deduction predicate logic in ELF, as given in [HHP93]. The details are given in that paper for the language of first-order arithmetic; we have made the obvious changes for an arbitrary language. Likewise, we have extended their translation on proof expressions, from the few examples of rules of deduction with which they deal, to all the rules which we present in Chapter 2.

Fix a particular first-order language. We are to construct an ELF signature, Σ_{FOL}. The signature begins with

$$\begin{array}{rcl} \iota & : & \mathbf{Type} \\ o & : & \mathbf{Type} \end{array}$$

For each constant c in the language, we have the declaration

$$c : \iota$$

For each n-ary function symbol f in the language, we have a constructor

$$c : \overbrace{\iota \to \cdots \to \iota}^{n} \to \iota$$

For each n-ary predicate symbol P in the language, we have a constructor

$$P : \overbrace{\iota \to \cdots \to \iota}^{n} \to o$$

We finish the signature with the following declarations:

$$
\begin{array}{rcl}
\neg & : & o \to o \\
\wedge & : & o \to o \to o \\
\vee & : & o \to o \to o \\
\supset & : & o \to o \to o \\
\forall & : & (\iota \to o) \to o \\
\exists & : & (\iota \to o) \to o \\
true & : & o \to \mathbf{Type} \\
\mathtt{CONT} & : & \Pi p : o.\Pi q : o.(true(\neg p) \to true(q)) \to \\
& & (true(\neg p) \to true(\neg q)) \to true(p) \\
\mathtt{IMP} - \mathtt{I} & : & \Pi p : o.\Pi q : o.(true(p) \to true(q)) \to true(\supset\ p\, q) \\
\mathtt{IMP} - \mathtt{E} & : & \Pi p : o.\Pi q : o.true(\supset\ p\, q) \to true(p) \to true(q) \\
\mathtt{AND} - \mathtt{I} & : & \Pi p : o.\Pi q : o.true(p) \to true(q) \to true(\wedge\, p\, q) \\
\mathtt{AND} - \mathtt{EL} & : & \Pi p : o.\Pi q : o.true(\wedge\, p\, q) \to true(p) \\
\mathtt{AND} - \mathtt{ER} & : & \Pi p : o.\Pi q : o.true(\wedge\, p\, q) \to true(q) \\
\mathtt{OR} - \mathtt{IL} & : & \Pi p : o.\Pi q : o.true(p) \to true(\vee\, p\, q) \\
\mathtt{OR} - \mathtt{IR} & : & \Pi p : o.\Pi q : o.true(q) \to true(\vee\, p\, q) \\
\mathtt{OR} - \mathtt{E} & : & \Pi p : o.\Pi q : o.\Pi r : o.true(\vee\, p\, q) \to \\
& & (true(p) \to true(r)) \to (true(q) \to true(r)) \to true(r) \\
\mathtt{ALL} - \mathtt{I} & : & \Pi p : \iota \to o.(\Pi x : \iota.true(p\, x)) \to true(\forall\, p) \\
\mathtt{ALL} - \mathtt{E} & : & \Pi p : \iota \to o.\Pi t : \iota.true(\forall\, p) \to true(p\, t) \\
\mathtt{EX} - \mathtt{I} & : & \Pi p : \iota \to o.\Pi t : \iota.true(p\, t) \to true(\exists\, p) \\
\mathtt{EX} - \mathtt{E} & : & \Pi p : \iota \to o.\Pi q : o.true(\exists\, p) \to \\
& & (\Pi x : \iota.true(p\, x) \to true(q)) \to true(q)
\end{array}
$$

For each finite set of variables $X = \{x_1, \ldots, x_n\}$ with a canonical enumeration, we define a translation ϵ_X from the terms and formulas of first-order logic with free variables in X to the terms of ELF as follows:

$$\begin{array}{rcl}
\epsilon_X(x) & \equiv & x \qquad (x \in X) \\
\epsilon_X(c) & \equiv & c \\
\epsilon_X(f(t_1, \ldots, t_n)) & \equiv & f\,\epsilon_X(t_1) \cdots \epsilon_X(t_n) \\
\epsilon_X(P(t_1, \ldots, t_n)) & \equiv & P\,\epsilon_X(t_1) \cdots \epsilon_X(t_n) \\
\epsilon_X(\neg\phi) & \equiv & \neg\,\epsilon_X(\phi) \\
\epsilon_X(\phi \wedge \psi) & \equiv & \wedge\,\epsilon_X(\phi)\,\epsilon_X(\psi) \\
\epsilon_X(\phi \vee \psi) & \equiv & \vee\,\epsilon_X(\phi)\,\epsilon_X(\psi) \\
\epsilon_X(\phi \rightarrow \psi) & \equiv & \supset\,\epsilon_X(\phi)\,\epsilon_X(\psi) \\
\epsilon_X(\forall x\phi) & \equiv & \forall(\lambda x : \iota.\epsilon_{X\cup\{x\}}(\phi)) \\
\epsilon_X(\exists x\phi) & \equiv & \exists(\lambda x : \iota.\epsilon_{X\cup\{x\}}(\phi))
\end{array}$$

We define Γ_X to be the context

$$x_1 : \iota, \ldots, x_n : \iota\,.$$

For each finite set of occurrence markers $\Delta = \{\xi_1, \ldots, \xi_m\}$ with a canonical enumeration, where ξ_i is an occurrence marker for ϕ_i, we define a translation $\epsilon_{X,\Delta}$ from the proof expressions with free variables in X and free occurrence markers in Δ to the terms of ELF as follows:

$$\begin{array}{rcl}
\epsilon_{X,\Delta}(\xi_i) & \equiv & \xi_i \\
\epsilon_{X,\Delta}((\mathrm{cont})(\phi,\psi,[\xi]\Pi,[\xi]\Pi')) & \equiv & \mathtt{CONT}\,\epsilon_X(\phi)\,\epsilon_X(\psi) \\
 & & (\lambda\xi : \mathit{true}(\epsilon_X(\neg\phi)).\epsilon_{X,\Delta\cup\{\xi\}}(\Pi)) \\
 & & (\lambda\xi : \mathit{true}(\epsilon_X(\neg\phi)).\epsilon_{X,\Delta\cup\{\xi\}}(\Pi')) \\
\epsilon_{X,\Delta}((\rightarrow I)(\phi,\psi,[\xi]\Pi)) & \equiv & \mathtt{IMP}-\mathtt{I}\,\epsilon_X(\phi)\,\epsilon_X(\psi) \\
 & & (\lambda\xi : \mathit{true}(\epsilon_X(\phi)).\epsilon_{X,\Delta\cup\{\xi\}}(\Pi)) \\
\epsilon_{X,\Delta}((\rightarrow E)(\phi,\psi,\Pi,\Pi')) & \equiv & \mathtt{IMP}-\mathtt{E}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi)\,\epsilon_{X,\Delta}(\Pi') \\
\epsilon_{X,\Delta}((\wedge I)(\phi,\psi,\Pi,\Pi')) & \equiv & \mathtt{AND}-\mathtt{I}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi)\,\epsilon_{X,\Delta}(\Pi') \\
\epsilon_{X,\Delta}((\wedge El)(\phi,\psi,\Pi)) & \equiv & \mathtt{AND}-\mathtt{EL}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi) \\
\epsilon_{X,\Delta}((\wedge Er)(\phi,\psi,\Pi)) & \equiv & \mathtt{AND}-\mathtt{ER}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi) \\
\epsilon_{X,\Delta}((\vee Il)(\phi,\psi,\Pi)) & \equiv & \mathtt{OR}-\mathtt{IL}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi) \\
\epsilon_{X,\Delta}((\vee Ir)(\phi,\psi,\Pi)) & \equiv & \mathtt{OR}-\mathtt{IR}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi) \\
\epsilon_{X,\Delta}((\vee E)(\phi,\psi,\chi,\Pi,[\xi]\Pi_1,[\xi']\Pi_2)) & \equiv & \mathtt{OR}-\mathtt{E}\,\epsilon_X(\phi)\,\epsilon_X(\psi)\,\epsilon_X(\chi) \\
 & & (\lambda\xi : \mathit{true}(\epsilon_X(\phi)).\epsilon_{X,\Delta\cup\{\xi\}}(\Pi_1)) \\
 & & (\lambda\xi' : \mathit{true}(\epsilon_X(\psi)).\epsilon_{X,\Delta\cup\{\xi'\}}(\Pi_2)) \\
\epsilon_{X,\Delta}((\forall I)([x]\phi,[x]\Pi)) & \equiv & \mathtt{ALL}-\mathtt{I}\,(\lambda x : \iota.\epsilon_{X\cup\{x\}}(\phi)) \\
 & & (\lambda x : \iota.\epsilon_{X\cup\{x\},\Delta}(\Pi)) \\
\epsilon_{X,\Delta}((\forall E)([x]\phi,t,\Pi)) & \equiv & \mathtt{ALL}-\mathtt{E}\,(\lambda x : \iota.\epsilon_{X\cup\{x\}}(\phi))\,\epsilon_X(t)\,\epsilon_{X,\Delta}(\Pi) \\
\epsilon_{X,\Delta}((\exists I)([x]\phi,t,\Pi)) & \equiv & \mathtt{EX}-\mathtt{I}\,(\lambda x : \iota.\epsilon_{X\cup\{x\}}(\phi))\,\epsilon_X(t)\,\epsilon_{X,\Delta}(\Pi) \\
\epsilon_{X,\Delta}((\exists E)([x]\phi,\psi,\Pi,[x,\xi]\Pi')) & \equiv & \mathtt{EX}-\mathtt{E}\,(\lambda x : \iota.\epsilon_{X\cup\{x\}}(\phi))\,\epsilon_X(\psi)\,\epsilon_{X,\Delta}(\Pi) \\
 & & (\lambda x : \iota.\lambda\xi : \mathit{true}(\epsilon_{X\cup\{x\}}(\phi)). \\
 & & \epsilon_{X\cup\{x\},\Delta\cup\{\xi\}}(\Pi'))
\end{array}$$

We define $\Gamma_{X,\Delta}$ to be the context

$$x_1 : \iota, \ldots, x_n : \iota, \xi_1 : \mathit{true}(\epsilon_X(\phi_1)), \ldots, \xi_m : \mathit{true}(\epsilon_X(\phi_m))\,.$$

The following adequacy theorems are proven in [HHP93]:

Theorem C.1 *The encoding ϵ_X is a bijection between the terms of the first-order language with free variables in X and the canonical forms [normal forms] of type ι in Σ_{FOL} and Γ_X. Moreover, the encoding is compositional in the sense that for $t[x_1, \ldots, x_n]$ a term with free variables in $X = \{t_1, \ldots, t_n\}$ and $t_1, \ldots, t_n$ terms with free variables in Y,*

$$\epsilon_Y(t[t_1, \ldots, t_n]) \equiv [\epsilon_Y(t_1), \ldots, \epsilon_Y(t_n)/x_1, \ldots, x_n]\epsilon_X(t)\,.$$

Theorem C.2 *The encoding ϵ_X is a bijection between the formulas of first-order arithmetic with free variables among those in X and the canonical forms of type o in Σ_{FOL} and Γ_X.*

Moreover, the encoding is compositional in the sense that for $\phi[x_1, \ldots x_n]$ *with free variables in* $X = \{x_1, \ldots, x_n\}$ *and* $t_1, \ldots, t_n$ *with free variables in* Y*,*

$$\epsilon_Y(\phi[t_1, \ldots, t_n]) \equiv [\epsilon_Y(t_1), \ldots, \epsilon_Y(t_n)/x_1, \ldots, x_n]\epsilon_X(\phi)\,.$$

Theorem C.3 *For every first-order formula* ϕ*, the encoding* $\epsilon_{X,\Delta}$ *is a bijection between valid proofs of* ϕ *with respect to* (X, Δ) *[in our terminology: proof expressions for* ϕ *with free variables in* X *and free occurrence markers in* Δ*] to canonical terms of type true*$(\epsilon_X(\phi))$ *in* Σ_{FOL} *and* $\Gamma_{X,\Delta}$*. Furthermore,* $\epsilon_{X,\Delta}$ *is compositional in the sense that for any proof contexts* (X, Δ) *and* (X', Δ') *with* $X = \{x_1, \ldots, x_m\}$ *and* $\Delta = (\xi_1 : \phi_1, \ldots, \xi_n : \phi_n)$ *[in our terminology:* $\Delta = \{\xi_1, \ldots, \xi_n\}$*, with each* ξ_i *being an occurrence marker for* ϕ_i*], if* $t_1, \ldots, t_m$ *are first-order terms whose variables are all in* X' *and if* $\Pi_1, \ldots, \Pi_n$ *are valid proofs of* $\phi_1, \ldots, \phi_n$ *with respect to* (X', Δ')*, then for any proof expression* $\Pi[x_1, \ldots, x_m, \xi_1, \ldots, \xi_n]$ *valid with respect to* (X, Δ)*,*

$$\begin{array}{ll} \multicolumn{2}{l}{\epsilon_{X',\Delta'}(\Pi[t_1, \ldots, t_m, \Pi_1, \ldots, \Pi_n])} \\ \equiv & [\epsilon_{X'}(t_1), \ldots, \epsilon_{X'}(t_m), \epsilon_{X',\Delta'}(\Pi_1), \ldots, \epsilon_{X',\Delta'}(\Pi_n)/x_1, \ldots, x_m, \xi_1, \ldots, \xi_n] \\ & \epsilon_{X,\Delta}(\Pi)\,. \end{array}$$