A New Type Theory for Representing Logics

Philippa Gardner *
University of Edinburgh

Abstract. We propose a new type theory for representing logics, called LF^+ and based on the Edinburgh Logical Framework. The new framework allows us to give, apparently for the first time, general definitions which capture how well a logic has been represented. Using our definitions, we show that, for example, first-order logic can be well-represented in LF^+, whereas linear and relevant logics cannot. These syntactic definitions of representation have a simple formulation as indexed isomorphisms, which both confirms that our approach is a natural one, and provides a link between type-theoretic and categorical approaches to frameworks.

1 Introduction

Much effort has been devoted to building systems for supporting the construction of formal proofs in various logics: examples of such systems include HOL [17], LEGO [22], NuPrl [8] and Alf [1]. Existing implementations for particular logics cannot easily be adapted to other logics. It is therefore desirable to seek a framework for representing logics, which unifies the structure common to a wide variety of logics. The aim of such a framework is to provide insights into the important theoretical question of what a logic is, and to yield general rather than logic-specific implementations of these logics.

Type theories have emerged as leading candidates for frameworks: examples include the Edinburgh Logical Framework [18] and Isabelle [27]. When using type theories in this way, the method of representation is necessarily *informal*, due to the variations in the styles of presentations of the logics under consideration; in fact, some logics cannot be well-represented because the meta-theory of the logic is incompatible with the meta-theory of the type theory. It is therefore necessary to provide criteria which determine when a representation is correct. We propose a new framework, called LF^+ and based on the Edinburgh Logical Framework of Harper, Honsell and Plotkin [18]. The new framework allows us to give, apparently for the first time, general definitions which capture how well a logic has been represented. A key property that permits such definitions is that logics with different consequence relations are represented in LF^+ using different specifications; this property apparently does not hold for other frameworks. Using our definitions, we show that, for example, natural deduction first-order logic can be well-represented in LF^+, whereas linear and relevant logics cannot. These syntactic definitions of representation have a simple formulation as indexed isomorphisms, which both confirms that our approach is a natural one, and provides a link between type-theoretic and categorical approaches to frameworks.

* Supported by a SERC Research Grant and the Laboratory for the Foundations of Computer Science, Edinburgh University.

The Edinburgh Logical Framework (LF), influenced by various AUTOMATH languages [6] and by Martin-Löf's work on the foundations of intuitionistic logic [23], constitutes an important advance in the study of logical frameworks. A logic is specified in LF by a signature declaring a finite set of constants that gives the syntax, judgements and inference rules of the logic; LF together with this signature forms the representing type theory. Each signature is accompanied by an adequacy theorem, which provides some confirmation that the consequence relation and proof structure have been well-represented. However, these adequacy theorems only apply to particular logics; they cannot be stated more generally for a wide class of logics. This is because information is lost during representation owing to the fact that the types have many different purposes. For example, from an LF signature we cannot distinguish between the terms and judgements. It is also not possible to identify those LF types which form part of the machinery of the representation, and therefore have no correspondence with the underlying logic. In LF^+, we take advantage of the distinctions between types given by the universes of Pure Type Systems [4] to provide a framework where such information is available. Using these distinctions, we are indeed able to identify that part of the representing type theory which corresponds to the underlying logic without reference to specific signatures, and so provide the general definitions of correct representation we seek.

There are many possible definitions of 'correct' representation, which depend on the amount of structure we wish to preserve. In this paper, we concentrate on the definition of *adequate* representation, which captures what it means for the *consequence relation* to be well-represented; the techniques given here can be adapted to also take the proof structure into account (see [14] and [13]). Our definition bears some relation to the notion of *uniform encoding* in LF defined in [19], which essentially involves tagging the LF signatures to indicate the types of interest. Using LF^+, we immediately know the part of the entailment relation we require, and so this 'extra-logical' tagging is not necessary. Recently, Simpson has studied the semantic analysis of a related notion of adequacy [32] for the type theory underlying Isabelle [27] and λ-Prolog [24].

This paper is organised as follows. In section 2, we introduce the type theory LF^+, and provide examples of representations. Section 3 contains our definition of adequate representation. In the last section, we sketch the categorical justification for our syntactic approach.

2 The Type Theory LF^+

The type theory of LF^+ is a variant of the LF type theory which allows for extra distinctions between types given by the universe structure: it has three universes, called *Sort*, *Extra* and *Judge*, in place of the single LF universe *Type*. The intention is for the terms of the logic to be represented using *Sort*, the judgements to be represented using *Judge*, and the universe *Extra* to contain the extra types which have no immediate correspondence with the underlying logic. LF^+ is presented as a Pure Type System with $\beta\eta$-equality ($\mathrm{PTS}^{\beta\eta}$), adapted to distinguish between signatures and contexts (definition A.2 in the appendix).

2.1 DEFINITION The framework LF^+ is the $\mathrm{PTS}^{\beta\eta}$ with signatures specified by $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$, where

$$\begin{array}{ll} \mathcal{U} = & \{Sort, Extra, Judge, Kind\} \\ \mathcal{V} = & \{Sort, Extra, Judge\} \\ \mathcal{A} = & \{Sort : Kind, Extra : Kind, Judge : Kind\} \\ \mathcal{R} = & \{(Sort, Kind, Kind), (Extra, Kind, Kind)\} \cup \{(Judge, Judge, Extra)\} \\ & \cup\{(s_1, s_2, Extra) : s_1, s_2 \in \{Sort, Extra\}\} \end{array}$$

An LF^+ type A is a *kind* if $\Gamma \vdash^{LF^+}_{\Sigma} A : Kind$ for some context Γ and signature Σ. Similarly, a type A is a *sort* or *judgement* if it inhabits the appropriate universe with respect to some context and signature. If $\Gamma \vdash^{LF^+}_{\Sigma} A : Extra$ then A is called an *extra type*. We call variable x a *sort variable* if $x \in Var^{Sort}$.

An important point to note is that the Π-abstraction of sorts, extra types and judgements all inhabit $Extra$. This is because we view Π-abstraction as part of the machinery of the meta-theory, rather than as having a direct correspondence in the object logic, since the aim is to capture many logics. In contrast, various predicative intuitionistic logics can be presented as type theories using the propositions-as-types paradigm [10] [6] [20], by equating Π-abstraction with universal quantification. Alternatives to the above choice of rules are discussed in [14].

We now discuss the representations of first-order and higher-order logic in LF^+. Other examples can be found in [13], or adapted from examples in [2].

2.2 EXAMPLE The specification in LF^+ of natural deduction first-order logic with the theory of arithmetic, denoted by Σ_{Fol}, contains the constants

$$\iota : Sort \qquad o : Extra,$$

whose inhabitants correspond to the well-formed arithmetic expressions and formulae respectively, together with the constant

$$true \quad : \quad o \to Judge,$$

where LF^+ terms of the form $true(\phi)$ for $\phi : o$ correspond to the judgements that formulae are true in first-order logic . The type ι is declared in $Sort$, since LF^+ terms in sort ι correspond to arithmetic expressions: in particular, sort variables correspond to variables of the logic. In first-order logic, the notion of a formula being well-formed is rarely distinguished syntactically from the judgement that a formula is true. This distinction is made explicit in the LF^+ representation. The constant $true$ inhabits $o \to Judge$ and constructs the LF^+ judgements, whereas the type o is declared as an extra type.

The rest of the specification follows the techniques used to represent first-order logic in LF. We declare constants $+ : \iota \to (\iota \to \iota) : Extra$ and $\forall : (\iota \to o) \to o : Extra$ for addition and universal quantification respectively. The term $+(t)(s)$, for $s, t : \iota$, is in sort ι and corresponds to an arithmetic expression. The term $\forall(\lambda x : \iota.\phi)$ for $\phi : o$ inhabits extra type o, and $true(\forall(\lambda x : \iota.\phi))$ is an

LF^+ judgement which therefore corresponds to a judgement that a formula of the form $\forall x.\phi'$ is true in first-order logic. The implication introduction rule of first-order logic is given by the constant

$$\supset I \ : \ \Pi\phi,\psi{:}o.(true(\phi) \to true(\psi)) \to true(\phi \supset \psi) \ : \ Extra,$$

so that $\supset I(\phi)(\psi)(\lambda p{:}true(\phi).q)$ inhabits LF^+ judgement $true(\phi \supset \psi)$, for $\phi : o$, $\psi : o$ and $q : true(\psi)$, and corresponds to a proof that a formula with shape $\phi' \supset \psi'$ is true in the underlying logic.

Contrast this representation with the LF representation of first-order logic where all the types inhabit the single universe *Type*. With the LF^+ representation, we can distinguish in a general way those LF^+ variables corresponding to the variables of the logic: they are the sort variables. We can also identify the LF^+ types corresponding to the judgements of the logic: they are the inhabitants of *Judge*. This general correspondence is not possible with LF, since the LF types ι, o and $true(\phi)$ for $\phi : o$ all inhabit *Type*. It is also not possible to identify the types which are meaningless from a first-order logic perspective: for example, the term $o \to \iota$ inhabits *Type* in LF. In LF^+ the same type inhabits *Extra*, and so we know immediately that it has no correspondence in the underlying logic.

The representation of first-order logic is simple and direct; not all representations are so easy. For example, extra types are required to represent the syntax of higher-order logic (see below) and extra types are also used to represent Hilbert-style S_4 [14]. These extra types cannot be identified in LF as we only know that they inhabit *Type*; in LF^+ they inhabit *Extra*. In fact, with LF this phenomenon sometimes results in the undesirable property that logics with different consequence relations are specified by the same signature (see [14]); this cannot occur using LF^+ if the consequence relations of both logics have been well-represented.

2.3 EXAMPLE The syntax of Church's higher-order logic [7] is based on simply typed λ-calculus:

$$\begin{array}{rl} \text{domains} & \alpha ::= \iota \,|\, o \,|\, \alpha \Rightarrow \alpha \\ \text{terms} & t ::= x^\alpha \,|\, (\lambda x^\alpha.t^\beta)^{\alpha\Rightarrow\beta} \,|\, (t^{\alpha\Rightarrow\beta}s^\alpha)^\beta \,|\, (\forall(t^{\alpha\Rightarrow o}))^o \,|\, (t^o \supset s^o)^o. \end{array}$$

The domains, viewed as syntactic classes, cannot be represented directly in LF^+ as there are infinitely many of them. In the signature specifying higher-order logic in LF^+, denoted by Σ_{Hol}, we declare the constants

$$\begin{array}{lcllcl} dom & : & Extra & \iota & : & dom \\ o & : & dom & \Rightarrow & : & dom \to dom \to dom \end{array}$$

which provide an obvious link between the domains and the terms in *dom*. We associate with each inhabitant of *dom* a term, identified with the objects of that domain, given by the constant

$$obj \quad : \quad dom \to Sort.$$

For each $\alpha : dom$, it is the term $obj(\alpha)$ which represents a domain of higher-order logic, rather than α itself, since inhabitants of $obj(\alpha)$ correspond to the terms of the logic. Thus $obj(\alpha)$ is a sort, and term $\alpha : dom$ is considered an extra term as the universes suggest. A more detailed account can be found in [13].

The above example demonstrates the technique of using the *Extra* universe to represent extra constants. Notice that, in the representations of first-order and higher-order logic in LF^+, the term corresponding to the syntactic class of formulae is the extra term o in the first representation, and sort $obj(o)$ in the second. The former distinguishes between the first-order terms and formulae, whereas the latter treats a formula as any other term expression. This mirrors precisely the behaviour of formulae in first-order and higher-order logic.

3 Adequate Representations

In this section, we provide a formal justification for defining the new framework LF^+. The examples in section 2 illustrate how to identify in a general way that part of the LF^+ entailment relation which corresponds to the underlying logic. This identification can be used to provide general definitions of correct representation. The definitions vary depending on how much structure of the logic one wishes to capture. We focus on the notion of an *adequate representation*, which states when the consequence relation of the logic has been well-represented; our techniques can be extended to also account for proofs (see [14]). An immediate result is that if the consequence relation has been well-represented, then the meta-theory of the consequence relation and the LF^+ entailment relation must be compatible: an obvious requirement, which could not be proved using LF.

3.1 Logical Preliminaries

In order to analyse representations of logics in LF^+, we require some standard terminology for the logics under consideration. This terminology is kept at an abstract level so that our definitions of representation apply to a wide variety of logics presented with different syntactic styles. For the purposes of this paper, logics consist of syntax, judgements and a consequence relation. The syntax is based on a possibly infinite set of syntactic classes with the subset S of syntactic classes containing variables distinguished. The inhabitants of the syntactic classes are called *expressions*, with those expressions inhabiting members of S called the *term expressions*. The usual notions of free variables and simultaneous substitution apply: the details are given in [14]. We write $fv(a)$ to denote the set of free variables in expression or judgement a, and $a[t_1, \ldots, t_n/x_1, \ldots, x_n]$ to denote the simultaneous substitution in a of term expressions $t_1, \ldots, t_n$ for distinct variables $x_1, \ldots, x_n$.

We focus on an abstract definition of consequence relation of a logic, with the intention that it is formed using the proof system of the logic. Unlike the usual definition of consequence relation (see for example [3]), we must take the variables into account since variables must be declared explicitly in type theory.

3.1 DEFINITION The *consequence relation* of a logic is a ternary relation written in the form $\Gamma \vdash_X J$, where J is a judgement, Γ is a multiset of judgements and X is a set of variables of the logic with $fv(J) \cup fv(\Gamma) \subseteq X$, and which satisfies

1. (reflexivity) $\Gamma \vdash_X J$ if $J \in \Gamma$;
2. (cut) $\Gamma \vdash_X J$ and $\Delta, J \vdash_X K$ imply $\Gamma, \Delta \vdash_X K$;
3. (substitution) $\Gamma \vdash_X J$ implies $\Gamma[\overline{t}/\overline{x}] \vdash_{X/\{\overline{x}\} \cup \bigcup_{i=1}^{n} fv(t_i)} J[\overline{t}/\overline{x}]$ where $[\overline{t}/\overline{x}]$ denotes $[t_1, \ldots, t_n/x_1, \ldots, x_n]$.

A consequence relation with *weakening* is a consequence relation which also satisfies the condition

4. (weakening) $\Gamma \vdash_X J$ and $\Gamma \subseteq \Delta$, with $fv(\Delta) \subseteq Y$ and $X \subseteq Y$, imply $\Delta \vdash_Y J$.

A consequence relation with *contraction* is a consequence relation which also satisfies the condition

5. (contraction) $\Gamma, J, J \vdash_X K$ implies $\Gamma, J \vdash_X K$.

3.2 Definition of Adequate Representation

The definition of an adequate representation is given for an arbitrary logic represented in LF^+. It provides a precise correspondence between the consequence relation of the logic, and that part of the LF^+ entailment relation given by the sorts and judgements. This correspondence identifies variables of the represented logic with sort variables, preserves substitution, and gives a sound and complete interpretation of the consequence relation in the entailment relation. Our definition is given in two parts. First, we define an *encoding* which gives the correspondence from the logic to the type theory. We then define an *adequate encoding*, which gives the correspondence the other way. These definitions are given up to $\beta\eta$-equivalence of LF^+ terms. We use the $\beta\eta$-long normal forms, which are canonical elements for the $\beta\eta$-equivalence classes under $\beta\eta$-equality (see [11]).

Some notation is required. Let $\mathcal{T}_\Sigma$ denote the set of preterms of (LF^+, Σ); we omit the subscript when the signature is apparent. Let Var^{Sort} and Var^{Judge} denote the sets of LF^+ sort and judgement variables respectively. Let S denote the set of syntactic classes containing variables of the logic, and let $T(X)$ and $J(X)$ denote the sets of term expressions and judgements with free variables in X, where X is a sequence of variables of the logic. Distinguish bijections $f^c : Var^c \to Var^{Sort}$ for each $c \in S$, where Var^c denotes the set of variables inhabiting c.

3.2 DEFINITION Let LOG be an arbitrary logic specified in LF^+ by Σ_{Log}. An *encoding* of LOG in $(\mathrm{ELF}^+, \Sigma_{Log})$ is a triple (η, ξ, δ) where $\eta : S \to \mathcal{T}$ is a function satisfying $\langle\,\rangle \vdash_{\Sigma_{Log}} \eta(c) : Sort$, for all $c \in S$, such that $\eta(c)$ is in $\beta\eta$-long normal form with respect to $(\Sigma_{Log}; \langle\,\rangle)$. Both ξ and δ are families of functions $\xi_X : T(X) \to \mathcal{T}$ and $\delta_X : J(X) \to \mathcal{T}$, indexed by finite sequences of variables of the logic $X = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$, such that:

1. $\xi_X(x) = f^c(x)$ for x^c in X;

2. for each term expression t from syntactic class $\sigma \in S$ and judgement j, both with free variables in the sequence $X = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$, we have

$$\Gamma_X \vdash_{\Sigma_{Log}} \xi_X(t) : \eta(\sigma) \qquad \Gamma_X \vdash_{\Sigma_{Log}} \delta_X(j) : Judge,$$

   where Γ_X is $\langle \xi_X(x_1) : \eta(\sigma_1), \ldots, \xi_X(x_n) : \eta(\sigma_n)\rangle$, and $\xi_X(t)$ and $\delta_X(j)$ are in $\beta\eta$-long normal form with respect to $(\Sigma_{Log}; \Gamma_X)$;

3. the ξ_X and δ_X are *compositional*: that is, for term expressions $t \in T(Y)$ and $s_1, \ldots, s_n \in T(X)$, and judgement $j \in J(Y)$,

$$\xi_X(t[\overline{s}/\overline{x}]) = \xi_Y(t)[\overline{\xi_X(s)}/\overline{\xi_Y(x)}] \qquad \delta_X(j[\overline{s}/\overline{x}]) = \delta_Y(j)[\overline{\xi_X(s)}/\overline{\xi_Y(x)}];$$

4. the interpretation is *sound*: that is, for sequences $X = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$ and $\langle j_1, \ldots, j_m\rangle$ of variables and judgements of the logic respectively,

$$\{j_1, \ldots, j_m\} \vdash_{\{x_1, \ldots, x_n\}} j \text{ implies}$$

$$\Gamma_X, p_1 : \delta_X(j_1), \ldots, p_m : \delta_X(j_m) \vdash_{\Sigma_{Log}} \_ : \delta_X(j),$$

   where Γ_X is $\langle \xi_X(x_1) : \eta(\sigma_1), \ldots, \xi_X(x_n) : \eta(\sigma_n)\rangle$, the $p_1, \ldots, p_m$ are distinct variables in Var^{Judge} and $\_ : \delta_X(j)$ denotes the inhabitation of LF^+ term $\delta_X(j)$.

The encoding definition depends on certain properties of the logics under consideration. We assume the syntactic classes do not depend on variables. We also assume that the term expressions and the judgements do not contain information regarding proof. In the above definition the soundness condition is only concerned with inhabitation of LF^+ terms, since the standard consequence relation of the logic contains no information about the derivations. When extending this definition to also account for the proof structure [14], the inhabitants of LF^+ judgements must correspond to proofs.

An *adequate* encoding provides an exact correspondence between a consequence relation of a logic and part of the entailment relation of the relating type theory. This is important since it not only states that we get a sound and complete interpretation of the consequence relation in the entailment relation, but also that we can recover the logic from the representing type theory. The

definition of adequate representation requires the following distinctions between LF^+ terms:

$sort_\Gamma^{\beta\eta} = \{A : \Gamma \vdash_\Sigma A : Sort$ and A is in $\beta\eta$-long normal form wrt $(\Sigma;\Gamma)\}$;
$texp_\Gamma^{\beta\eta} = \{M : \Gamma \vdash_\Sigma M : A : Sort$ and M is in $\beta\eta$-long normal form wrt $(\Sigma;\Gamma)\}$;
$judge_\Gamma^{\beta\eta} = \{J : \Gamma \vdash_\Sigma J : Judge$ and J is in $\beta\eta$-long normal form wrt $(\Sigma;\Gamma)\}$.

Using these distinctions and given encoding (η, ξ, δ), we can be more precise with the ranges of η, and of ξ_X and δ_X for sequence of variables X. Let Γ_X denote the contexts of sorts $\langle \xi_X(x_1) : \eta(\sigma_1), \ldots, \xi_X(x_n) : \eta(\sigma_n)\rangle$, where X is sequence $\langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$ of variables of the logic. We write $\xi_X : T(X) \to texp_{\Gamma_X}^{\beta\eta}$ and $\delta_X : J(X) \to judge_{\Gamma_X}^{\beta\eta}$ to denote the functions extensionally equal to ξ_X and δ_X, but with the more precise ranges. These are well-defined by condition 2 in definition 3.2. We also write $\eta : S \to sort_{\langle\rangle}^{\beta\eta}$. These functions play a central role in the definition of an adequate encoding.

3.3 DEFINITION An encoding (η, ξ, δ) is *adequate* when $\eta : S \to sort_{\langle\rangle}^{\beta\eta}$ is a bijection, and

1. for each finite sequence $X = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$ of variables, the functions $\xi_X : T(X) \to texp_{\Gamma_X}^{\beta\eta}$ and $\delta_X : J(X) \to judge_{\Gamma_X}^{\beta\eta}$ are bijections;

2. the interpretation is *complete*; that is, for sequences $X = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$ and $\langle j_1, \ldots, j_m\rangle$ of variables and judgements of the logic respectively,

$$\Gamma_X, p_1 : \delta_X(j_1), \ldots, p_m : \delta_X(j_m) \vdash_{\Sigma_{Log}} \_ : \delta_X(j) \text{ implies}$$

$$\{j_1, \ldots, j_m\} \vdash_{\{x_1, \ldots, x_n\}} j,$$

where the $p_1, \ldots, p_m$ are distinct variables in Var^{Judge} and $\_ : \delta_X(j)$ denotes the inhabitation of LF^+ term $\delta_X(j)$.

We say that the logic is *adequately represented* in LF^+ if there is an adequate encoding of LOG in (LF^+, Σ_{Log}).

3.4 THEOREM The signatures Σ_{Fol} and Σ_{Hol} (examples 2.2 and 2.3) provide adequate LF^+ representations of first-order and higher-order logic respectively.

The proof of this theorem is technical, but not difficult. Essentially, it involves presenting correspondences between the logics and their representing type theories, which satisfy the conditions stipulated: see [13] for details.

It is intuitively clear that, for a logic to be well-represented in a framework, the meta-theory of the logic and the framework must be compatible. We are at last able to capture this intuition as the following theorem states.

3.5 THEOREM Logics which are adequately represented in LF^+ have consequence relations which satisfy weakening and contraction.

An immediate corollary is that there are no adequate LF^+ representations of the standard consequence relations of linear [16] and relevance [12] logics. In recent work, Miller, Plotkin and Pym have been investigating a type theory for representing logics [25], which incorporates ideas from linear logic to adapt the standard notion of context so that these consequence relations can be well-represented.

4 Categorical Account

We have argued that the syntactic definition of adequate representation defines when the consequence relation of a logic has been well-represented in the representing type theory. Our arguments can be reinforced by showing that our syntactic definition has a direct categorical formulation as an indexed isomorphism; again these results can be extended to account for proofs. It is known that the mathematical structure common to the logics under consideration can be captured by the structure of strict indexed categories [26] (or split fibrations [5]), whose base categories are given by term expressions and whose fibres are given by consequence relations. By utilising the fact that we are able to identify in a general way that part of the LF^+ entailment relation which corresponds to the underlying logic, we define indexed categories for the representing type theories, whose base categories are defined using sorts, and whose fibres are defined using LF^+ judgements. Encodings then determine indexed functors, and we obtain the following result.

4.1 THEOREM Let (η, ξ, δ) be an encoding of LOG in $(LF^+; \Sigma_{Log})$. The encoding is adequate if and only if the indexed functor determined by (η, ξ, δ) is an indexed isomorphism.

This result both confirms that our approach is a natural one, and provides a link between type-theoretic and categorical approaches to frameworks. More details can be found in [13] and [14].

Acknowledgements My thanks go to Gordon Plotkin and John Power for many helpful discussions.

References

1. L. Augustsson, T. Coquand and B. Nordström. A Short Description of Another Logical Framework, *1st Workshop on Logical Frameworks*, ed.s Huet and Plotkin, pp 39–42, 1990.
2. A. Avron, F. Honsell, I.A. Mason and R. Pollack. Using Typed Lambda Calculus to Implement Formal Systems on a Machine, *Automated Reasoning*, Vol. 9, pp 309–354, 1992.
3. A. Avron. Simple Consequence Relations, *Information and Computation*, Vol. 1, pp 105-139, 1991.
4. H. Barendregt. Lambda Calculi with Types, *Handbook of Logic in Computer Science*, OUP, Vol. 2, pp 117–309, 1991.

5. J. Bénabou. Fibred Categories and the Foundations of Naive Category Theory, *Symbolic Logic*, Vol. 50, pp 10-37, 1985.
6. N.G. de Bruijn. A Survey of the Project Automath, in [31], pp 589–606, 1980.
7. A. Church. A formulation of the simple theory of types, *Symbolic Logic*, Vol. 5, pp 56-68, 1940.
8. R.L. Constable et al. *Implementing Mathematics with the NuPrl Proof Development System*, Prentice-Hall, 1986.
9. T. Coquand and G. Huet. The Calculus of Constructions, *Information and Computation*, Vol. 76, pp 95–120, 1988.
10. H.B. Curry and R. Feys. *Combinatory Logic*, North Holland, 1958.
11. G. Dowek and G. Huet. On the Definition of the η-long normal form in Type Systems of the Cube, submitted for publication.
12. J.M. Dunn. Relevant Logic and Entailment, *Handbook of Philosophical Logic*, eds. D. Gabbay and F. Guenthner, Vol. 3, pp 117–224, 1984.
13. P.A. Gardner. *Representing Logics in Type Theory*, Ph.D. Thesis, Edinburgh University, 1992.
14. P.A. Gardner. *Equivalences between Logics and their Representing Type Theories*, full paper, submitted for publication.
15. J.H. Geuvers. The Church–Rosser property for $\beta\eta$-reduction in typed lambda calculus, *Logic in Computer Science*, pp 453–460, 1992.
16. J.Y. Girard. Linear Logic, *Theoretical Computer Science*, Vol. 50, pp 1-102, 1987.
17. M.J.C. Gordon. HOL: A Proof Generating System for Higher-Order Logic. *VSLI Specification, Verification and Synthesis*, ed.s Birtwistle and Subrahmanyam, Kluwer Academic Publishers, pp 73–128, 1987.
18. R. Harper, F. Honsell and G. Plotkin. A Framework for Defining Logics, *Association for Computing Machinery*, Vol. 40, Part 1, pp 143-184, 1992. Preliminary version in *Logic in Computer Science*, pp 194-204, 1987.
19. R. Harper, D. Sannella and A. Tarlecki. *Structure and Representation in LF.* Technical Report ECS-LFCS-89-75, Edinburgh University, 1989. Abridged version in *Logic in Computer Science*, pp 226-237, 1989.
20. W.A. Howard. The Formulae-as-types Notion of Construction, in [31], pp 479–490, 1980.
21. F.W. Lawvere. Equality in Hyperdoctrines and Comprehension Schema as an Adjoint Functor, *Applications of Categorical Algebra*, American Mathematical Society, pp 1–14, 1970.
22. Z. Luo and R. Pollack. *LEGO Proof Development System: User's Manual*, Technical Report ECS-LFCS-92-211, Edinburgh University, 1992.
23. P. Martin-Löf. *On the Meanings of the Logical Constants and the Justifications of the Logical Laws*, Technical Report 2, Università di Siena, 1985.
24. D. Miller and G. Nadathur. Higher-order Logic Programming, *LNCS 225: 3rd International Logic Programming Conference* Springer, ed. Shapiro, pp 448–462, 1986.
25. D. Miller, G. Plotkin and D. Pym *A Relevant Analysis of Natural Deduction*, in preparation.

26. R. Paré and D. Schumacher. Abstract Families and the Adjoint Functor Theorems, *Indexed Categories and their Applications*, ed.s Johnstone and Paré, pp 1–125, 1978.
27. L. Paulson. The Foundations of a Generic Theorem Prover, *Automatic Reasoning*, Vol. 5, pp 363 – 397, 1987.
28. D. Prawitz. *Natural Deduction: A Proof-theoretical study.* Almquist and Wiksell, Stockholm, 1965.
29. A Unification Algorithm for the $\lambda\Pi$-calculus, *Foundations of Computer Science*, Vol. 3, No. 3, pp 333–378, 1992.
30. A. Salvesen. *The Church-Rosser Property for Pure Type Systems with $\beta\eta$-reduction*, submitted for publication.
31. J.P. Seldin and J.R. Hindley, editors. *To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, 1980.
32. A. Simpson. Kripke Semantics for a Logical Framework, *1992 Workshop on Types for Proofs and Programs*, Båstad, 1992.

A PTS$^{\beta\eta}$s with signatures

In this appendix, we define Pure Type Systems with $\beta\eta$-equality [15] [30], adapted to distinguish between signatures and contexts. This adaptation is necessary to give a precise representation of LF$^+$, since the formation of signatures and contexts are different: signatures are used to specify logics, one of the uses of contexts is to represent assumptions. In [18], LF is presented as a type theory with β-equality. The stronger $\beta\eta$-equality allows for a smoother correspondence between the logic and its representing type theory, since every well-typed term is convertible to a unique canonical element, and also simplifies considerably the unification problem for LF [29] and hence for LF$^+$. [1]

A.1 DEFINITION A *specification of a PTS$^{\beta\eta}$ with signatures* is a quadruple $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ where

1. $\mathcal{U}$ is a set, called the set of *universes*;
2. $\mathcal{V} \subseteq \mathcal{U}$, called the set of *variable universes*;
3. $\mathcal{A} \subseteq \mathcal{U} \times \mathcal{U}$ is the set of *axioms*;
4. $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{U} \times \mathcal{U}$ is the set of *rules.*

The set of preterms, $\mathcal{T}$, of a PTS$^{\beta\eta}$ with signatures given by specification $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ is defined using countably infinite sets of variables Var and constants $Const$ with the abstract syntax $\mathcal{T} ::= x \mid u \mid a \mid \Pi x{:}\mathcal{T}.\mathcal{T} \mid \lambda x{:}\mathcal{T}.\mathcal{T} \mid \mathcal{T}\mathcal{T}$, where u is a universe, $x \in Var$ and $a \in Const$. It is useful to divide the sets Var

[1] The meta-theoretic results for $\beta\eta$-equality where open problems when the LF paper was written.

and $Const$ into disjoint infinite subsets Var^v and $Const^u$ for $v \in \mathcal{V}$ and $u \in \mathcal{U}$. Arbitrary variables and constants are denoted by x, y, z and a, b, c respectively.

The simple method for declaring constants is based on the standard approach used, for example, in the type theory defining LF [18]; more motivation and different approaches are discussed in [13].

A.2 DEFINITION The *PTS$^{\beta\eta}$ with signatures*, specified by $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ is defined by the following proof system:

AXIOM $\quad \langle\,\rangle \vdash_\Sigma u : v \qquad u : v \in \mathcal{A}$

SIGNATURE
$$\frac{\langle\,\rangle \vdash_\Sigma A : u}{\langle\,\rangle \vdash_{\Sigma, a:A} a : A} \qquad u \in \mathcal{U}, a \in Const^u, a \notin dom(\Sigma)$$

$$\frac{\langle\,\rangle \vdash_\Sigma A : u \qquad \langle\,\rangle \vdash_\Sigma B : C}{\langle\,\rangle \vdash_{\Sigma, a:A} B : C} \qquad u \in \mathcal{U}, a \in Const^u, a \notin dom(\Sigma)$$

CONTEXT
$$\frac{\Gamma \vdash_\Sigma A : v}{\Gamma, x : A \vdash_\Sigma x : A} \qquad v \in \mathcal{V}, x \in Var^v, x \notin dom(\Gamma)$$

$$\frac{\Gamma \vdash_\Sigma A : v \qquad \Gamma \vdash_\Sigma B : C}{\Gamma, x : A \vdash_\Sigma B : C} \qquad v \in \mathcal{V}, x \in Var^v, x \notin dom(\Gamma)$$

Π -RULE
$$\frac{\Gamma \vdash_\Sigma A : u \qquad \Gamma, x : A \vdash_\Sigma B : v}{\Gamma \vdash_\Sigma \Pi x{:}A.B : w} \qquad (u, v, w) \in \mathcal{R}$$

λ -RULE
$$\frac{\Gamma \vdash_\Sigma \Pi x{:}A.B : u \qquad \Gamma, x : A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda x{:}A.M : \Pi x{:}A.B} \qquad u \in \mathcal{U}$$

APP
$$\frac{\Gamma \vdash_\Sigma M : \Pi x{:}A.B \qquad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : B[N/x]}$$

CONV
$$\frac{\Gamma \vdash_\Sigma A : B \qquad \Gamma \vdash_\Sigma C : u}{\Gamma \vdash_\Sigma A : C} \qquad B =_{\beta\eta} C, u \in \mathcal{U}$$

where $=_{\beta\eta}$ denotes $\beta\eta$-equality on preterms. A PTS$^{\beta\eta}$ with signature Σ, denoted by (ζ, Σ), is defined by restricting the entailments of the PTS$^{\beta\eta}$ given by specification ζ such that the entailments of interest are of the form $\Gamma \vdash^\zeta_\Sigma A : B$. Based on ideas from [30], Geuvers [15] proves the standard type theoretic results for functional, normalising PTS$^{\beta\eta}$s including the substitution lemma, the Church-Rosser property, subject reduction and strengthening ; it is trivial to adapt these results to PTS$^{\beta\eta}$s with signatures. A PTS$^{\beta\eta}$ with signatures is *normalising* if every well-typed term in it reduces to a $\beta\eta$-normal form. A PTS$^{\beta\eta}$ with signatures specified by $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ is *functional* if $\mathcal{A}$ is a function from $\mathcal{U}$ to $\mathcal{U}$ and $\mathcal{R}$ is a function from $\mathcal{V} \times \mathcal{U}$ to $\mathcal{U}$. (That is, if $u : c$ and $u : w \in \mathcal{A}$ then $u \equiv w$ and if (u, v, w) and $(u, v, w') \in \mathcal{R}$ then $w \equiv w'$.) The type theory of LF$^+$ is a functional, normalising PTS$^{\beta\eta}$.