Equivalences between Logics and their Representing Type Theories*

Philippa Gardner†

Abstract

We propose a new framework 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. These definitions are possible since we are able to distinguish in a generic way that part of the LF^+ entailment which corresponds to the underlying logic. This distinction does not seem to be possible with 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. We also show that our 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 [Gor87], LEGO [LP92], Alf [ACN90] and NuPrl [Con86]. 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 [HHP87] and Isabelle [Pau87]. 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. The new framework allows us to give, apparently for the first time, general definitions which capture how well a logic has been represented. These definitions are possible since we are able to distinguish, in a generic way, that part of the LF^+ entailment relation which corresponds to

*This paper is based on research from my thesis, called ‘Representing Logics in Type Theory’, published in 1992. A preliminary version appeared in ECS-LFCS-92-251; a shorter version appeared in the Fourth International Conference on Logic Programming and Automated Reasoning [Gar93].

the underlying logic. This distinction does not seem to be possible using other frameworks; in section 2 we discuss this point for LF. 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.

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 two definitions of representation: *adequate* representation, which defines when the consequence relation of a logic has been well-represented by the LF^+ entailment relation, and *natural* representation, which requires in addition that derivations have been well-represented. Our adequacy definition bears some relation to the notion of *uniform encoding* in LF defined in [HST89], 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. More reecently, Simpson has studied the semantic analysis of a related notion of adequacy [Sim92] for the type theory underlying Isabelle [Pau87] and λ-Prolog [MN86].

Summary We introduce the new framework LF^+ in section 2, and give examples to illustrate representation in this framework. Section 3 contains the formal justification for LF^+. We give an axiomatic account of a logic, which has just enough structure to present logics as indexed categories. Using this account, we define the notions of adequate and natural representation. We also give examples to illustrate these definitions and prove that certain logics cannot be well-represented in LF^+. In section 4, we show that our syntactic definitions of representation give rise to indexed isomorphisms.

2 The Logical Framework LF^+

The framework LF^+ is based on the Edinburgh Logical Framework (LF) of Harper, Honsell and Plotkin [HHP87]. Influenced by various AUTOMATH languages [Bru80] and by Martin-Löf's work on the foundations of intuitionistic logic [Mar85], LF constitutes an important advance in the study of logical frameworks. It is not possible, however, to provide general definitions of 'correct' representation using LF. These definitions are possible using LF^+.

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 a LF signature does not provide enough information to reconstruct the underlying logic. For example, a LF signature does not distinguish those types corresponding to the syntactic classes and those corresponding to judgements. It also does not distinguish the extra types which have no correspondence in the underlying logic, and which are often required as part of the machinery of the representation. It is therefore not possible to identify the part of the LF entailment relation which corresponds to the consequence relation of the underlying logic without appealing to that particular logic. In LF$^+$, we take advantage of the distinctions between types given by the universes of Pure Type Systems [Bar92] to provide a framework where such an identification is possible.

The type theory of LF$^+$ is a variant of the LF type theory which allows for extra distinctions between types. 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. 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.

In this section, we present the type theory of LF$^+$ using an extension of the Pure Type System presentation to allow for $\beta\eta$-equality and signatures. The meta-theoretic results necessary to make this extension rigorous can be found in [Geu92]. We give examples of representations in LF$^+$ to show the techniques required. These examples are also used to illustrate our definitions of adequate and natural representation given in section 3.

2.1 Pure Type Systems with $\beta\eta$-equality and signatures

The distinction between terms required by LF$^+$ exploits, and was partially inspired by, the techniques of Beradi [Ber90] and Terlouw [Ter89] in extending Barendregt's λ-cube to Pure Type Systems (PTSs) [Bar92]. The framework LF$^+$ is presented as a PTS with $\beta\eta$-equality, 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 is different. This difference is not surprising as signatures are used to specify logics, whereas one of the uses of contexts is to represent assumptions. In [HHP87], 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 [Pym92] and hence for LF$^+$. [1]

2.1 Definition A *specification of a PTS$_{\beta\eta}$ with signatures* is a quadruple $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ where

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

The set of *preterms* $\mathcal{T}$ of a PTS$_{\beta\eta}$ with signatures given by the specification $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ is defined using countably infinite sets of variables *Var* and constants *Const* with the abstract

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

syntax

$$A ::= u \mid x \mid a \mid \Pi x{:}A.B \mid \lambda x{:}A.B \mid AB,$$

where u is a universe, $x \in \mathit{Var}$ and $a \in \mathit{Const}$. It is useful to divide the sets Var 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. We let $\rhd_\beta$ and $\rhd_{\beta\eta}$ denote the reflexive and transitive closure of the standard one-step β- and $\beta\eta$-reductions on preterms, and let $=_\beta$ and $=_{\beta\eta}$ denote the corresponding equalities. A *precontext* Γ is a finite, possibly empty, sequence of the form $\langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$ with $x_i \in \mathit{Var}$ for all $i \in \{1, \ldots, n\}$. We write $\mathit{dom}(\Gamma) = \{x_1, \ldots, x_n\}$. We also use the analogous notions of *presignature* Σ and $\mathit{dom}(\Sigma)$.

The simple method for declaring constants is based on the standard approach used, for example, in the type theory defining LF [HHP87]. More motivation and different approaches are discussed in [Gar92].

2.2 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 $\quad \langle\,\rangle \vdash_{\langle\,\rangle} u : v \qquad (u, v) \in \mathcal{A}$

Signature $\quad \dfrac{\langle\,\rangle \vdash_\Sigma A : u}{\langle\,\rangle \vdash_{\Sigma, a:A} a : A} \qquad u \in \mathcal{U}, a \in \mathit{Const}^u, a \notin \mathit{dom}(\Sigma)$

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

context $\quad \dfrac{\Gamma \vdash_\Sigma A : v}{\Gamma, x : A \vdash_\Sigma x : A} \qquad v \in \mathcal{V}, x \in \mathit{Var}^v, x \notin \mathit{dom}(\Gamma)$

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

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

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

App $\quad \dfrac{\Gamma \vdash_\Sigma M : \Pi x{:}A.B \qquad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : B\{N/x\}}$

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

Given $\Gamma \vdash_\Sigma A : B$, we say that Σ is a *signature*, that Γ is a *context* and that A and B are *terms*. We sometimes write $\Gamma \vdash_\Sigma A : B : C$ to denote $\Gamma \vdash_\Sigma A : B$ and $\Gamma \vdash_\Sigma B : C$, and write $\Gamma \vdash^\zeta_\Sigma A : B$ to emphasise that the entailment $\Gamma \vdash_\Sigma A : B$ is valid in the PTS with specification ζ. Given a specification ζ, the $\mathrm{PTS}_{\beta\eta}$ with signature Σ, denoted by (ζ, Σ), is defined by restricting the entailments of the $\mathrm{PTS}_{\beta\eta}$ specified by ζ such that the entailments of interest are of the form $\Gamma \vdash^\zeta_\Sigma A : B$. A $\mathrm{PTS}_{\beta\eta}$ with signatures is *normalising* if every well-typed term in it reduces to a $\beta\eta$-normal form. A $\mathrm{PTS}_{\beta\eta}$ with signatures specified by $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$ is *functional* if $\mathcal{A}$ is a partial function from $\mathcal{U}$ to $\mathcal{U}$ and $\mathcal{R}$ is a partial function from $\mathcal{V} \times \mathcal{U}$ to $\mathcal{U}$. (That is, if $u : v$ and $u : w \in \mathcal{A}$ then $v \equiv w$, and if (u, v, w) and $(u, v, w') \in \mathcal{R}$ then $w \equiv w'$.) Geuvers shows that the standard type theoretic results hold for functional, normalising PTSs with $\beta\eta$-equality [Geu92], which extend Salvesen's results for LF with $\beta\eta$-equality [Sal90][2]. It is trivial to adapt these results to PTSs with signatures. The results required for this paper are stated below for an arbitrary functional, normalising $\mathrm{PTS}_{\beta\eta}$ with signatures.

2.3 Lemma (Weakening) If $\Gamma \vdash_\Sigma A : B$ and $\Gamma \subseteq \Delta$ for context Δ, then $\Delta \vdash_\Sigma A : B$.

2.4 Lemma (Substitution) If $\Gamma, x : A, \Delta \vdash_\Sigma B : C$ and $\Gamma \vdash_\Sigma M : A$, then $\Gamma, \Delta\{M/x\} \vdash_\Sigma B\{M/x\} : C\{M/x\}$.

2.5 Lemma (Subject Reduction) If $\Gamma \vdash_\Sigma A : B$ and $A \rhd_{\beta\eta} A'$, then $\Gamma \vdash_\Sigma A' : B$.

2.6 Lemma (Church-Rosser for $\beta\eta$-equality) If $\Gamma \vdash_\Sigma A : B$ and $A \rhd_{\beta\eta} C$ and $A \rhd_{\beta\eta} D$, then there exists a preterm E such that $C \rhd_{\beta\eta} E$ and $D \rhd_{\beta\eta} E$.

Recall that the Church-Rosser property for β-equality holds for the set of preterms. The corresponding result for $\beta\eta$-equality does not hold for the preterms.

2.7 Lemma (Congruence for $\beta\eta$-equality) If $\Gamma \vdash_\Sigma A : C$ and $\Gamma \vdash_\Sigma B : C$ and $A =_{\beta\eta} B$, then $A \rhd_{\beta\eta} D$ and $B \rhd_{\beta\eta} D$ for some term D.

2.2 The Type Theory LF^+

LF^+ uses three universes, called *Sort*, *Extra* and *Judge*, in place of the single LF universe *Type*, which allows enough distinctions between LF^+ terms to make the definitions of adequate and natural representation feasible.

2.8 Definition The framework LF^+ is the $\mathrm{PTS}_{\beta\eta}$ with signatures given by the specification $(\mathcal{U}, \mathcal{V}, \mathcal{A}, \mathcal{R})$, where

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

[2] Salvesen has also shown that the Church-Rosser property holds for a wide class of PTSs [Sal91].

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

The idea of splitting the universe *Type* of LF into three motivates the choice of $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{A}$. Some justification of the set of rules $\mathcal{R}$ is required. 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 a wide variety of logics. In contrast, various predicative intuitionistic logics can be presented as type theories using the propositions-as-types paradigm [CF58, Bru80, How80], by equating Π-abstraction with universal quantification.

Alternatives to this choice of rules are discussed in [Gar92]. For example, it seems reasonable to assume that the syntax of a logic does not depend on the derivations of the logic; for this reason we have omitted the rules $(\mathit{Judge}, \mathit{Sort}, \mathit{Extra})$ and $(\mathit{Judge}, \mathit{Extra}, \mathit{Extra})$. A natural example of a logic where formulae depend on proofs is a first-order logic with a choice operator: that is, given a proof p of $\exists x.\phi(x)$ we obtain a term t dependent on p such that $\phi(t)$ is true. Such a logic would include a syntactic class of proofs, and judgements linking proofs with formulae. Our assumption does not therefore restrict such a logic. Also, notice that we include $(\mathit{Sort}, \mathit{Kind}, \mathit{Kind})$ and $(\mathit{Extra}, \mathit{Kind}, \mathit{Kind})$, but not $(\mathit{Judge}, \mathit{Kind}, \mathit{Kind})$. As the examples below illustrate, the first two rules are used to form judgements. We do not include the rule $(\mathit{Judge}, \mathit{Kind}, \mathit{Kind})$, since it would correspond in the logic to syntactic classes or judgements depending on derivations.

2.3 Representation in LF$^+$

We sketch three examples of representations in LF$^+$: natural-deduction first-order logic [Pra65] has a direct representation, higher-order logic [Chu40] has a representation which requires extra constants to represent the syntax of the logic, and Hilbert-style S_4 [Che80] has a representation which requires extra constants to represent the consequence relation. These examples are also used to illustrate our definitions of adequate and natural representation in the next section. Further examples can be found in [Gar92], or adapted from the examples in [AHMP92].

2.9 Example We consider a fragment of natural-deduction first-order logic with arithmetic, whose terms and formulae are given by abstract grammar

$$\begin{array}{ll} \text{terms} & t ::= x \mid 0 \mid \mathit{succ}(t) \mid +(t)(t') \\ \text{formulae} & \phi ::= (t{=}t') \mid \phi \supset \psi \mid \forall x.\phi, \end{array}$$

and we consider the rules:

$$\frac{\begin{array}{c}(\phi)\\ \vdots \\ \psi\end{array}}{\phi \supset \psi} \supset I \qquad \frac{\phi \supset \psi \quad \phi}{\psi} \supset E \qquad \frac{\phi}{\forall x.\phi} \forall I^* \qquad \frac{\forall x.\phi}{\phi\{t/x\}} \forall E$$

where * denotes that x does not occur free in the assumptions. This fragment is enough to illustrate the ideas behind the LF$^+$ representation of first-order logic.

The specification in LF$^+$ of the above fragment of first-order logic with the theory of arithmetic, denoted by Σ_{Fol}, contains the constants

$$\begin{array}{l} \iota : \mathit{Sort} \\ o : \mathit{Extra}, \end{array}$$

whose inhabitants correspond to the well-formed arithmetic terms and formulae respectively. In general, inhabitants of *Sort* should correspond to syntactic classes containing variables. Syntactic classes which do not contain variables should be represented by inhabitants of *Extra*. This distinction between the syntactic classes is required to give a precise link between the consequence relation of the logic and the corresponding LF$^+$ entailment relation. For example, the consequence relation of natural-deduction first-order logic does not contain formulae variables. This use of the universe *Extra* is, however, comparatively minor; more interesting uses are illustrated by the higher-order logic and Hilbert-style S_4 examples given below. We also declare the constant

$$\mathit{true} \quad : \quad o \to \mathit{Judge},$$

where LF$^+$ terms of the form $\mathit{true}(\phi)$ for $\phi : o$ correspond to the judgement that formulae are true in first-order logic. In the corresponding LF representation of first-order logic, it is not possible to distinguish those LF terms corresponding to judgements and those corresponding to syntactic classes without appealing to the particular LF constants used, since all these terms inhabit the universe *Type*.

The rest of the specification follows the techniques used to represent first-order logic in LF and is given below; for a detailed account of the techniques involved, see [HHP87] or [Gar92]. The terms and formulae are represented using the constants

$$\begin{array}{ccl} 0 & : & \iota \\ \mathit{succ} & : & \iota \to \iota \\ + & : & \iota \to \iota \to \iota \\ = & : & \iota \to \iota \to o \\ \supset & : & o \to o \to o \\ \forall & : & (\iota \to o) \to o. \end{array}$$

The rules given above are represented by the constants

$$\begin{array}{rcl} \supset I & : & \Pi\phi, \psi{:}o.(\mathit{true}(\phi) \to \mathit{true}(\psi)) \to \mathit{true}(\phi \supset \psi) \\ \supset E & : & \Pi\phi, \psi{:}o.\mathit{true}(\supset(\phi)(\psi)) \to \mathit{true}(\phi) \to \mathit{true}(\psi) \\ \forall I & : & \Pi F{:}\iota \to o.(\Pi x{:}\iota.\mathit{true}(Fx)) \to \mathit{true}(\forall(\lambda x{:}\iota.Fx)) \\ \forall E & : & \Pi F{:}\iota \to o.\Pi t{:}\iota.\mathit{true}(\forall(F)) \to \mathit{true}(Ft) \end{array}$$

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

2.10 Example The syntax of Church's higher-order logic [Chu40] 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

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

which provide an obvious link between the domains and the terms in *dom*. We associate with each inhabitant of *dom* a LF$^+$ 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. The inhabitants of $obj(\alpha)$ are constructed in a similar fashion to the inhabitants of ι and o given in the previous example. The full LF$^+$ specification of higher-order logic can be found in [Gar92].

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.

2.11 Example [Hilbert-style S_4] The representation of Hilbert-style S_4 is an example where extra constants are used to represent the consequence relation of the logic in LF$^+$. The formulae are given by the abstract grammar:

$$\phi ::= X \mid \phi \supset \psi \mid \Box\phi,$$

where X denotes a formula variable, and we consider the rules:

A1	$\phi \supset (\psi \supset \phi)$
A2	$(\phi \supset (\psi \supset \theta)) \to ((\phi \supset \psi) \to (\phi \supset \theta))$
A3	$\Box\phi \supset \phi$
A4	$\Box(\phi \supset \psi) \supset (\Box\phi \supset \Box\psi)$

A5 $\Box\phi \supset \Box\Box\phi$

MP $$\frac{\phi \qquad \phi \supset \psi}{\psi}$$

NEC* $$\frac{\phi}{\Box\phi}$$

where $*$ indicates the side-condition that ϕ is a theorem.

The difficulty of representing Hilbert-style S_4 in LF^+ (or LF) lies with the Nec-rule. This rule cannot be represented directly by the standard method of declaring a constant *nec* inhabiting $\Pi\phi{:}o.true(\phi) \to true(\Box\phi)$ since such a constant would force the inhabitation of $true(\Box\phi)$ in any context entailing $true(\phi)$. The solution [Avr91] centres on a logic, denoted by $\mathcal{L}_{new}$, with the same syntax as S_4 and judgements of the form $\phi\, true$ and $\phi\, valid$ for a formula ϕ, with the intuition that $\phi\, valid$ in $\mathcal{L}_{new}$ corresponds to ϕ being a theorem in Hilbert-style S_4, and $\phi\, true$ in $\mathcal{L}_{new}$ corresponds to ϕ being true in Hilbert-style S_4.

Using Avron's approach, Hilbert-style S_4 is represented in LF by first specifying $\mathcal{L}_{new}$ in LF and then, in the accompanying adequacy theorem, limiting the correspondence to those LF terms representing truth judgements [AHMP92]. This example shows that in LF it is possible for one signature to specify different logics, in this case Hilbert-style S_4 and $\mathcal{L}_{new}$. Using LF^+, this phenomenon cannot occur if the consequence relations of both logics have been well-represented. Thus, in particular, the specification of Hilbert-style S_4 in LF^+ is different from the specification of $\mathcal{L}_{new}$. The difference occurs in the universes which the terms corresponding to $\phi\, true$ and $\phi\, valid$ inhabit. For the LF^+ representation of Hilbert-style S_4, we declare the constants $true : o \to Judge$ and $valid : o \to Extra$, which indicate that the terms of the form $true(\phi)$ correspond to the judgements of Hilbert-style S_4 whilst the terms of the form $valid(\phi)$ are extra terms given by the representation. (In the signature specifying $\mathcal{L}_{new}$ in LF^+, the constants *true* and *valid* both inhabit $o \to Judge$.) The specification of Hilbert-style S_4, denoted by Σ_{Mod}, is given in figure 1; a detailed explanation can be found in [Gar92].

3 Adequate and Natural Representation

In this section, we provide a formal justification for defining the new framework LF^+. The examples in section 2.3 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, and *natural representation*, which gives some measure that the proof structure has been preserved during representation. 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.

$$
\begin{array}{lcl}
o & : & Sort \\
\\
\supset & : & o \to o \to o \\
\Box & : & o \to o \\
\\
true & : & o \to Judge \\
valid & : & o \to Extra \\
\\
C & : & \Pi\phi{:}o.valid(\phi) \to true(\phi) \\
A1 & : & \Pi\phi,\psi{:}o.valid(\phi \supset (\psi \supset \phi)) \\
A2 & : & \Pi\phi,\psi,\theta{:}o.valid((\phi \supset (\psi \supset \theta)) \to ((\phi \supset \psi) \to (\phi \supset \theta))) \\
A3 & : & \Pi\phi{:}o.valid(\Box\phi \supset \phi) \\
A4 & : & \Pi\phi,\psi{:}o.valid(\Box(\phi \supset \psi) \supset (\Box\phi \supset \Box\psi)) \\
A5 & : & \Pi\phi{:}o.valid(\Box\phi \supset \Box\Box\phi) \\
MP_V & : & \Pi\phi,\psi{:}o.valid(\phi) \to valid(\phi \supset \psi) \to valid(\psi) \\
Nec & : & \Pi\phi{:}o.valid(\phi) \to valid(\Box\phi) \\
MP_T & : & \Pi\phi,\psi{:}o.true(\phi) \to true(\phi \supset \psi) \to true(\psi)
\end{array}
$$

Figure 1: The LF^+ specification of Hilbert-style S_4, denoted by Σ_{Mod}.

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 notions of free variables and simultaneous substitution must be defined at this abstract level. First some notation is required. Let T^c denote the set of term expressions and Var^c denote the set of variables inhabiting syntactic class $c \in S$. We write $T = \bigcup_{c \in S} T^c$. Let J denote the set of judgements of the logic. We define a *substitution function* as a function

$$\alpha : \textstyle\bigcup_{c \in S} Var^c \to T,$$

such that α is almost everywhere the identity and

1. $x \in Var^c$ implies $\alpha(x) \in T^c$.

Using this function α, we define the notion of *simultaneous substitution of* α in a term or judgement by the functions

$$sub_\alpha : T \to T \ \text{ and } \ Sub_\alpha : J \to J,$$

with the following properties:

2. $t \in T^c$ implies $sub_\alpha(t) \in T^c$;

3. $x \in \bigcup_{c \in S} Var^c$ implies $sub_\alpha(x) = \alpha(x)$;

4. $sub_{id} = id_T$ and $Sub_{id} = id_J$;

5. $sub_\alpha \circ sub_\beta = sub_\gamma$ and $Sub_\alpha \circ Sub_\beta = Sub_\gamma$, if $\gamma(x) = sub_\alpha(\beta(x))$ for all $x \in \bigcup_{c \in S} Var^c$;

6. $\alpha = \beta$ implies $sub_\alpha = sub_\beta$ and $Sub_\alpha = Sub_\beta$.

Notice that property 1 follows from properties 2 and 3. Let $t\{s_1/x_1, \dots s_n/x_n\}$ (sometimes denoted by $t\{\vec{s}/\vec{x}\}$) for $t \in T$ denote the term expression $sub_\alpha(t)$, where $\alpha(x_i) = s_i$ for $i \in \{1, \dots, n\}$ and $\alpha(y) = y$ for $y \notin \{x_1, \dots, x_n\}$. Similarly, we let $j\{s_1/x_1, \dots s_n/x_n\}$, or $j\{\vec{s}/\vec{x}\}$, denote the judgement $Sub_\alpha(j)$.

We also define the *free variable functions*

$$fv : T \to \mathcal{P}(\textstyle\bigcup_{c \in S} Var^c) \;\text{ and }\; Fv : J \to \mathcal{P}(\textstyle\bigcup_{c \in S} Var^c)$$

satisfying the properties:

7. $fv(x) = \{x\}$ for $x \in \bigcup_{c \in S} Var^c$;

8. $fv(sub_\alpha(t)) = \bigcup_{x \in fv(t)} fv(\alpha(x))$ and $Fv(Sub_\alpha(j)) = \bigcup_{x \in Fv(j)} fv(\alpha(x))$;

9. $\alpha|_{fv(t)} = \beta|_{fv(t)}$ implies $sub_\alpha(t) = sub_\beta(t)$, and $\alpha|_{Fv(j)} = \beta|_{Fv(j)}$ implies $Sub_\alpha(j) = Sub_\beta(j)$, where $\alpha|_A$ for $A \subseteq \bigcup_{c \in S} Var^c$ denotes the restriction of function α to domain A.

Notice that property 6 follows from property 9.

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 [Avr91]), our definition depends on the free variables of the judgements under consideration. This refinement of the consequence relation is necessary, since we aim to link the consequence relation of a logic with its corresponding LF$^+$ entailment relation, and 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 $\Gamma \vdash_{\{\vec{x}\}} j$, where j is a judgement, Γ is a multiset of judgements and $\{\vec{x}\}$ is a set of distinct variables of the logic with $Fv(j) \cup Fv(\Gamma) \subseteq \{\vec{x}\}$, and which satisfies

1. (reflexivity) $j \vdash_{\{\vec{x}\}} j$ if $Fv(j) = \{\vec{x}\}$;

2. (variable weakening) $\Gamma \vdash_{\{\vec{x}\}} j$ and $y \notin \{\vec{x}\}$ implies $\Gamma \vdash_{\{\vec{x},y\}} j$;

3. (substitution) $\Gamma \vdash_{\{\vec{x},\vec{y}\}} j$ implies $\Gamma\{\vec{t}/\vec{y}\} \vdash_{\{\vec{x}\} \cup fv(\vec{t})} j\{\vec{t}/\vec{y}\}$, where if Γ is the multiset $\{j_1, \dots, j_m\}$ then $\Gamma\{\vec{t}/\vec{y}\}$ denotes the multiset $\{j_1\{\vec{t}/\vec{y}\}, \dots, j_m\{\vec{t}/\vec{y}\}\}$, and if $\{\vec{t}/\vec{y}\}$ denotes $\{t_1/y_1, \dots, t_n/y_n\}$ then $fv(\vec{t})$ denotes $\bigcup_{i=1}^n fv(t_i)$;

4. (cut) $\Gamma \vdash_{\{\vec{x}\}} j$ and $\Delta, j \vdash_{\{\vec{x}\}} k$ imply $\Gamma \cup \Delta \vdash_{\{\vec{x}\}} k$.

A consequence relation satisfying *weakening* is a consequence relation which also satisfies

5. (weakening) $\Gamma \vdash_{\{\vec{x}\}} j$ and $Fv(k) \subseteq \{\vec{x}\}$ imply $\Gamma \cup \{k\} \vdash_{\{\vec{x}\}} j$.

A consequence relation satisfying *contraction* is a consequence relation which also satisfies

6. (contraction) $\Gamma, j, j \vdash_{\{\vec{x}\}} k$ implies $\Gamma, j \vdash_{\{\vec{x}\}} k$.

This abstract notion of a logic gives us enough structure to present logics as strict indexed categories (definition 4.4).

3.2 Type theoretic preliminaries: the $\beta\eta$-long normal forms

Our analysis of representations of logics in LF$^+$ is given up to $\beta\eta$-equivalence. In particular, we concentrate on LF$^+$ terms in $\beta\eta$-long normal form with respect to the appropriate signature and context, which are canonical elements for the equivalence classes under $\beta\eta$-equality. The intuition is that the terms in $\beta\eta$-long normal form with respect to some signature and context are *fully applied*. For example, in the LF$^+$ representation of first-order logic specified by Σ_{Fol} (example 2.9), we associate the formula $\forall x.(y = x)$ with the LF$^+$ term $\forall(\lambda x' : \iota.(= (y')(x')))$ in context $y' : \iota$, rather than the β-normal form $\forall(= (y'))$. The constant $=: \iota \to \iota \to o$ is fully applied in the first term, but not in the second. Our characterisation of terms in $\beta\eta$-long normal form depends on the notion of *β-normal form* of a term and the *arity* of the universes, constants and variables with respect to the appropriate signature and context. This characterisation corresponds to the definition of canonical normal form for LF [HHP87].

3.2 Definition Let ζ be an arbitrary functional, normalising PTS$_{\beta\eta}$ with signatures. A pre-term A is in *β-normal form* if it contains no subterms of the form $(\lambda x : A_1.A_2)B$. Let $A \rhd_\beta B$ such that B is in β-normal form. Then B is the *canonical β-normal form* of A.

3.3 Definition Let $\Gamma \vdash^{\mathrm{LF}^+}_{\Sigma} A : B$.

1. The *arity* of universe u in A with respect to $(\Sigma; \Gamma)$ is 0.

2. The *arity* of free variable or constant @ in A with respect to $(\Sigma; \Gamma)$ is the number of Πs in the prefix of C', where $@ : C$ is declared in Σ or Γ, and C' is the canonical β-normal form of C.

3. The *arity* of bound variable x in A with respect to $(\Sigma; \Gamma)$ is the number of Πs in the prefix of C', where C is the type accompanying the binding occurrence of x and C' is the canonical β-normal form of C.

3.4 Definition

1. Let $\Gamma \vdash^{\mathrm{LF}^+}_{\Sigma} A : B$. The term A is in *$\beta\eta$-long normal form* with respect to $(\Sigma; \Gamma)$ if it has shape
$$\lambda x_1 : A_1 \ldots \lambda x_n : A_n.\Pi y_1 : B_1 \ldots \Pi y_m : B_m.@M_1 \ldots M_k$$
where $n, m, k \geq 0$, the term @ is a universe, constant or variable of arity k with respect to $(\Sigma; \Gamma)$, and

(a) each A_i for $i \in \{1, \ldots, n\}$ is in $\beta\eta$-long normal form with respect to

$$(\Sigma; \Gamma, x_1{:}A_1, \ldots, x_{i-1}{:}A_{i-1});$$

(b) each B_j for $j \in \{1, \ldots, m\}$ is in $\beta\eta$-long normal form with respect to

$$(\Sigma; \Gamma, x_1{:}A_1, \ldots, x_n{:}A_n, y_1{:}B_1, \ldots, y_{j-1}{:}B_{j-1});$$

(c) each M_r for $r \in \{1, \ldots, k\}$ is in $\beta\eta$-long normal form with respect to

$$(\Sigma; \Gamma, x_1{:}A_1, \ldots, x_n{:}A_n, y_1{:}B_1, \ldots, y_m{:}B_m).$$

2. Let $\Gamma \vdash_\Sigma^{\mathrm{LF}^+} A : B$. A term A' is a *$\beta\eta$-long normal form of A with respect to* $(\Sigma; \Gamma)$ if $\Gamma \vdash_\Sigma A' : B$, the term A' is in $\beta\eta$-long normal form with respect to $(\Sigma; \Gamma)$ and $A =_{\beta\eta} A'$.

3. Let $\Gamma \vdash_\Sigma^{\mathrm{LF}^+}$ A:B such that Γ is $\langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$. The context Γ is in $\beta\eta$-long normal form with respect to Σ if each A_i for $i \in \{1, \ldots, n\}$ is in $\beta\eta$-long normal form with respect to $(\Sigma; \langle x_1{:}A_1, \ldots, x_{i-1}{:}A_{i-1} \rangle)$.

The key property of $\beta\eta$-long normal forms is that they provide canonical terms for the equivalence classes under $\beta\eta$-equality. The proof of this property is non-trivial, and can be adapted from results in [DHW93] and [Gar93a].

3.5 THEOREM Let $\Gamma \vdash_\Sigma^{\mathrm{LF}^+} A : B$. The $\beta\eta$-long normal form of A with respect to $(\Sigma; \Gamma)$ exists and is unique.

3.3 Adequate Representations

We now give the definition of *adequate representation*, which characterises when the consequence relation of a logic has been well-represented in LF^+. Our definition 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. The 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 using LF^+ terms in $\beta\eta$-long normal form.

First some notation is required. Recall that, for an arbitrary logic, we distinguish the set S of syntactic classes containing variables. The variables are partitioned by the syntactic classes in S. For example, in higher-order logic we have variables x^ι and y^o in the syntactic classes ι and o respectively. The sort variables of LF^+, however, are not partitioned; the sorts they inhabit are determined by the contexts in which they are declared. For example, in the representation of higher-order logic in LF^+ given in example 2.10, we have the freedom to declare the contexts $x : obj(\iota)$ and $x : obj(o)$ for sort variable $x \in Var^{Sort}$. We obtain a precise link between variables of the logic and sort variables by introducing a countably infinite set of variables of the logic, denoted by Var^{Log}, which is not partitioned by the syntactic classes. We then write x^c to declare that $x \in \mathit{Var}^{\mathit{Log}}$ inhabits syntactic class c, and let $T(\vec{x})$ and $J(\vec{x})$ denote the sets of term expressions and judgements with free variables in $\{\vec{x}\}$, where $\vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n} \rangle$ and the x_i are distinct variables in $\mathit{Var}^{\mathit{Log}}$. Our slightly non-standard approach allows us the freedom to declare a variable in any syntactic class in S, just as we can declare a LF$^+$ sort variable to inhabit any sort[3].

3.6 Definition Let *Log* be an arbitrary logic specified in LF$^+$ by Σ_{Log}. An *encoding* $[\![\_]\!]$ of *Log* in (LF$^+$, Σ_{Log}) consists of a function

$$[\![\_]\!]^S : S \to \mathcal{T},$$

and families of functions

$$[\![\_]\!]^T_{\vec{x}} : T(\vec{x}) \to \mathcal{T} \text{ and } [\![\_]\!]^J_{\vec{x}} : J(\vec{x}) \to \mathcal{T},$$

indexed by finite sequences of distinct variables $\vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n} \rangle$, such that

1. $c \in S$ implies $\langle\,\rangle \vdash_{\Sigma_{\mathit{Log}}} [\![c]\!]^S : \mathit{Sort}$, where $[\![c]\!]^S$ is in $\beta\eta$-long normal form with respect to $(\Sigma_{\mathit{Log}}\,; \langle\,\rangle)$;
2. $[\![x_i]\!]^T_{\vec{x}} = x'_i$ for $i \in \{1, \ldots, n\}$, where we distinguish a bijection $(\_)' : \mathit{Var}^{\mathit{Log}} \to \mathit{Var}^{\mathit{Sort}}$;
3. for each term expression t from syntactic class c and judgement j, both with free variables contained in $\{x_1^{c_1}, \ldots, x_n^{c_n}\}$, we have

   $$\Gamma_{\vec{x}} \vdash_{\Sigma_{\mathit{Log}}} [\![t]\!]^T_{\vec{x}} : [\![c]\!]^S;$$
   $$\Gamma_{\vec{x}} \vdash_{\Sigma_{\mathit{Log}}} [\![j]\!]^J_{\vec{x}} : \mathit{Judge},$$

   where $\Gamma_{\vec{x}}$ is $\langle x'_1 : [\![c_1]\!]^S, \ldots, x'_n : [\![c_n]\!]^S \rangle$, and $[\![t]\!]^T_{\vec{x}}$ and $[\![j]\!]^J_{\vec{x}}$ are in $\beta\eta$-long normal form with respect to $(\Sigma_{\mathit{Log}}\,; \Gamma_{\vec{x}})$;
4. the functions $[\![\_]\!]^T_{\vec{x}} : T(\vec{x}) \to \mathcal{T}$ and $[\![\_]\!]^J_{\vec{x}} : J(\vec{x}) \to \mathcal{J}$ are *compositional*: that is, for term expressions $t \in T(\vec{y})$ and $s_1, \ldots, s_r \in T(\vec{x})$, and judgement $j \in J(\vec{y})$,

   $$[\![t\{\vec{s}/\vec{y}\}]\!]^T_{\vec{x}} = [\![t]\!]^T_{\vec{y}}\{[\![s_1]\!]^T_{\vec{x}}/y'_1, \ldots, [\![s_n]\!]^T_{\vec{x}}/y'_n\}$$
   $$[\![j\{\vec{s}/\vec{y}\}]\!]^J_{\vec{x}} = [\![j]\!]^J_{\vec{y}}\{[\![s_1]\!]^T_{\vec{x}}/y'_1, \ldots, [\![s_n]\!]^T_{\vec{x}}/y'_n\}$$

5. the interpretation is *sound*: that is, for sequence $\langle j_1, \ldots, j_m \rangle$ of judgements of the logic,

   $$\{j_1, \ldots, j_m\} \vdash_{\{\vec{x}\}} j \text{ implies } \Gamma_{\vec{x}}, p_1 : [\![j_1]\!]^J_{\vec{x}}, \ldots, p_m : [\![j_m]\!]^J_{\vec{x}} \vdash_{\Sigma_{\mathit{Log}}} \_ : [\![j]\!]^J_{\vec{x}},$$

   where $\Gamma_{\vec{x}}$ is defined in part 3, the $p_1, \ldots, p_m$ are distinct variables in $\mathit{Var}^{\mathit{Judge}}$ and $\_ : [\![j]\!]^J_{\vec{x}}$ denotes the inhabitation of LF$^+$ term $[\![j]\!]^J_{\vec{x}}$.

[3] An alternative approach is to work with equivalences up to renaming of variables. This approach is technically more difficult, and so we choose not to use it.

We shall sometimes omit the superscripts on $[\![\_]\!]^S$, $[\![\_]\!]^T_{\vec{x}}$ and $[\![\_]\!]^J_{\vec{x}}$, when the domain is apparent.

Notice that 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 derivations. 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 regarding the structure of derivations. In our definition of natural representation (definition 3.12), the inhabitants of LF$^+$ judgements correspond to derivations.

An *adequate* encoding provides an exact correspondence between the consequence relation of a logic and part of the entailment relation of the relating type theory. In order to provide the correspondence from the representing type theory to the underlying logic, we identify the following sets of LF$^+$ terms:

$$\begin{aligned}
&\mathit{sort}^{\beta\eta}_{\Gamma} = \{A \text{ such that } \Gamma \vdash_{\Sigma} A : \mathit{Sort} \text{ and } A \text{ is in } \beta\eta\text{-long normal form wrt. } (\Sigma;\Gamma)\};\\
&\mathit{texp}^{\beta\eta}_{\Gamma} = \{M \text{ such that } \Gamma \vdash_{\Sigma} M : A : \mathit{Sort},\ M \text{ is in } \beta\eta\text{-long normal form wrt. } (\Sigma;\Gamma)\};\\
&\mathit{judge}^{\beta\eta}_{\Gamma} = \{J \text{ such that } \Gamma \vdash_{\Sigma} J : \mathit{Judge} \text{ and } J \text{ is in } \beta\eta\text{-long normal form wrt. } (\Sigma;\Gamma)\}.
\end{aligned}$$

Given encoding $[\![.]\!]$ and using the sets of LF$^+$ terms distinguished above, we are now able to be more precise with the ranges of the function $[\![\_]\!]^S$, and the functions $[\![\_]\!]^T_{\vec{x}}$ and $[\![\_]\!]^J_{\vec{x}}$. Let $\Gamma_{\vec{x}}$ denote the contexts of sorts $\langle x'_1 : [\![c_1]\!]^S, \ldots, x'_n : [\![c_n]\!]^S\rangle$. We write $[\![\_]\!]^T_{\vec{x}} : T(\vec{x}) \to \mathit{texp}^{\beta\eta}_{\Gamma_{\vec{x}}}$ and $[\![\_]\!]^J_{\vec{x}} : J(\vec{x}) \to \mathit{judge}^{\beta\eta}_{\Gamma_{\vec{x}}}$ to denote the functions extensionally equal to $[\![\_]\!]^T_{\vec{x}}$ and $[\![\_]\!]^J_{\vec{x}}$, but with the more precise ranges. These are well-defined by condition 2 in definition 3.6. We also write $[\![\_]\!] : S \to \mathit{sort}^{\beta\eta}_{\langle\,\rangle}$. These functions play a central role in the definition of an adequate encoding, which we now give.

3.7 Definition An encoding $[\![.]\!]$ of *Log* in (LF$^+$, Σ_{Log}) is *adequate* when

1. $[\![\_]\!] : S \to \mathit{sort}^{\beta\eta}_{\langle\,\rangle}$ is a bijection;
2. for each finite sequence $\vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n}\rangle$ of variables, the functions $[\![\_]\!]_{\vec{x}} : T(\vec{x}) \to \mathit{texp}^{\beta\eta}_{\Gamma_{\vec{x}}}$ and $[\![\_]\!]_{\vec{x}} : J(\vec{x}) \to \mathit{judge}^{\beta\eta}_{\Gamma_{\vec{x}}}$ are bijections;
3. the interpretation is *complete*; that is, for sequences $\vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n}\rangle$ and $\langle j_1, \ldots, j_m\rangle$ of variables and judgements of the logic respectively,
$$\Gamma_{\vec{x}}, p_1 : [\![j_1]\!]_{\vec{x}}, \ldots, p_m : [\![j_m]\!]_{\vec{x}} \vdash_{\Sigma_{Log}} \_ : [\![j]\!]_{\vec{x}} \text{ implies } \{j_1, \ldots, j_m\} \vdash_{\{\vec{x}\}} j,$$
where the $p_1, \ldots, p_m$ are distinct variables in $\mathit{Var}^{\mathit{Judge}}$ and $\_ : [\![j]\!]_{\vec{x}}$ denotes the inhabitation of LF$^+$ term $[\![j]\!]_{\vec{x}}$.

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

The representations of first-order logic, higher-order logic and Hilbert-style S_4 sketched in section 3.1 are all adequate. Ideally, the correspondence between a logic and its representation in a framework should be immediately apparent, although it is not clear that this goal is compatible with the aim of representing a wide variety of logics. With LF$^+$, the correspondence between a well-represented logic and the representing type theory is usually obvious, although some work must be done to show that the conditions stipulated in definitions 3.6 and 3.7 are satisfied.

3.8 THEOREM The signature Σ_{Fol} provides an adequate representation of first-order logic in LF$^+$.

Proof We give the encoding $[\![\_]\!]$ of first-order logic in (LF$^+$, Σ_{Fol}). The technical details required to show that $[\![\_]\!]$ is an adequate encoding are straightforward (see [Gar92] for details). The function $[\![\_]\!] : S \to sort^{\beta\eta}_{\langle\,\rangle}$ is:

$$[\![term]\!] \quad = \quad \iota.$$

For each sequence $\vec{x} = \langle x_1, \ldots, x_n\rangle$ of variables of first-order logic (we omit the superscripts as there is only one syntactic class containing variables), the function $[\![\_]\!]_{\vec{x}} : T(\vec{x}) \to texp^{\beta\eta}_{\Gamma_{\vec{x}}}$ is defined inductively on the structure of $t \in T(\vec{x})$ as follows:

$$\begin{array}{rcll}
[\![x]\!]_{\vec{x}} & = & x', & \qquad x \in \{\vec{x}\} \\
[\![0]\!]_{\vec{x}} & = & 0 & \\
[\![succ(t)]\!]_{\vec{x}} & = & succ([\![t]\!]_{\vec{x}}) & \\
[\![t+s]\!]_{\vec{x}} & = & +([\![t]\!]_{\vec{x}})([\![s]\!]_{\vec{x}}) &
\end{array}$$

where $(\_)' : Var^{Log} \to Var^{Sort}$ is a bijection and $\Gamma_{\vec{x}}$ is $\langle x_1' : \iota, \ldots, x_n' : \iota\rangle$.

Similarly, for each sequence of variables $\vec{x} = \langle x_1, \ldots, x_n\rangle$, the function $[\![\_]\!]_{\vec{x}} : J(\vec{x}) \to judge^{\beta\eta}_{\Gamma_{\vec{x}}}$ is given by $[\![\phi]\!]_{\vec{x}} = true(\langle\!\langle\phi\rangle\!\rangle_{\vec{x}})$ for formula ϕ, where $\langle\!\langle\_\rangle\!\rangle_{\vec{x}} : F(\vec{x}) \to o^{\beta\eta}_{\Gamma_{\vec{x}}}$, with $F(\vec{x})$ denoting the set of formulae with free variables in $\{\vec{x}\}$, is defined inductively as follows:

$$\begin{array}{rcl}
\langle\!\langle t = s\rangle\!\rangle_{\vec{x}} & = & =([\![t]\!]_{\vec{x}})([\![s]\!]_{\vec{x}}) \\
\langle\!\langle \phi \supset \psi\rangle\!\rangle_{\vec{x}} & = & \supset(\langle\!\langle\phi\rangle\!\rangle_{\vec{x}})(\langle\!\langle\psi\rangle\!\rangle_{\vec{x}}) \\
\langle\!\langle \forall y.\phi\rangle\!\rangle_{\vec{x}} & = & \forall(\lambda y' {:} \iota.\langle\!\langle\phi\rangle\!\rangle_{\vec{x},y})
\end{array}$$

□

The representations of higher-order logic and Hilbert-style S_4 sketched in section 2.3 are also adequate. We state the result without proof; the details can be found in [Gar92].

3.9 THEOREM The signatures Σ_{Hol} and Σ_{Mod} provide adequate representations of higher-order logic and Hilbert-style S_4 respectively.

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.

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

Proof Let $[\![.]\!]$ be an adequate encoding of *Log* in (LF$^+$, Σ_{Log}). We show that the cut property holds for *Log*. The other properties in definition 3.1 hold in a similar fashion. Assume that the relations $\{j_1, \ldots, j_m\} \vdash^{Log}_{\{\vec{x}\}} j$ and $\{j, k_1, \ldots, k_r\} \vdash^{Log}_{\{\vec{x}\}} k$ hold. Then we have

$$\Gamma_{\vec{x}}, p_1 : [\![j_1]\!]_{\vec{x}}, \ldots, p_m : [\![j_m]\!]_{\vec{x}} \vdash_{\Sigma_{Log}} \pi : [\![j]\!]_{\vec{x}} \text{ and ;}$$

$$\Gamma_{\vec{x}}, q : [\![j]\!]_{\vec{x}}, q_1 : [\![k_1]\!]_{\vec{x}}, \ldots, q_r : [\![k_r]\!]_{\vec{x}} \vdash_{\Sigma_{Log}} \pi' : [\![k]\!]_{\vec{x}},$$

where $\Gamma_{\vec{x}}$ is the context $\langle x'_1 : [\![c_1]\!], \ldots, x'_n : [\![c_n]\!]\rangle$, and without loss of generality we assume that $\{p_1, \ldots, p_m\} \cap \{q_1, \ldots, q_r\} = \emptyset$. It is a straightforward matter to show, using the substitution lemma, that

$$\Gamma_{\vec{x}}, p_1 : [\![j_1]\!]_{\vec{x}}, \ldots, p_m : [\![j_m]\!]_{\vec{x}}, q_1 : [\![k_1]\!]_{\vec{x}}, \ldots, q_r : [\![k_r]\!]_{\vec{x}} \vdash_{\Sigma_{Log}} \pi'\{\pi/q\} : [\![k]\!]_{\vec{x}}.$$

Since $[\![.]\!]$ is an adequate encoding, we have $\{j_1, \ldots, j_m, k_1, \ldots, k_r\} \vdash^{Log}_{\{\vec{x}\}} k$. Hence, *Log* satisfies the cut property of definition 3.1. $\Box$An immediate corollary is that there are no adequate LF$^+$ representations of the standard consequence relations of linear [Gir87] and relevance [Dun84] logics of the form $\phi_1, \ldots, \phi_m \vdash_{\{\vec{x}\}} \phi$, where the $\phi_1, \ldots, \phi_m, \phi$ are formulae and $\{\vec{x}\}$ is a set of variables denoting formulae. In recent work, Miller, Plotkin and Pym have been investigating a type theory for representing logics [MPP92], which incorporates ideas from linear logic to adapt the standard notion of context so that these consequence relations can be well-represented.

3.4 Natural Representations

Our definition of *natural representation* extends the notion of adequate representation to require, in addition, a correspondence between derivations in the logic and LF$^+$ terms inhabiting judgements. This extension gives some indication that the proof system of a logic can be mimicked by its representation in LF$^+$, and provides a full generalisation to arbitrary logics of the adequacy theorems accompanying the LF representations in [HHP87] for particular logics.

Following [HHP87], our definition of natural representation focuses on the notion of a *consequence relation of proofs*. This notion is defined by extending the syntax of the logic to incorporate a set of *proof expressions*, which includes an infinite set of proof variables. The definitions of simultaneous substitution and free variables for proof expressions can be given in a similar fashion to the definitions in section 3.1 for term expressions and judgements. The consequence relation of proofs identifies the *valid* proof expressions, with the intuition that valid proof expressions correspond to derivations in the logic. In order to define natural representations, it is enough for us to give an abstract characterisation of the consequence relation of proofs. Our intention is for the consequence relation of proofs to be constructed by adapting the proof system of the logic to identify those proof expressions that are valid. For example, associated with the $\supset E$-rule of first-order logic are proof expressions of the form $\supset E(\phi)(\psi)(p)(q)$, such that $\supset E(\phi)(\psi)(p)(q)$ is valid whenever p denotes a valid proof expression for $\phi \supset \psi$, and q denotes a valid proof expression for ϕ.

A *consequence relation of proofs* is a ternary relation written in the form $\Gamma \vdash_{\{\vec{x}\}} \pi : j$, where Γ is a set of proof assumptions of the form $\{p_1{:}j_1, \ldots, p_m{:}j_m\}$ for $m \geq 0$ such that the p_i are distinct proof variables, the $j_1, \ldots, j_m$ and j are judgements whose free variables are contained in the distinct set of variables $\{\vec{x}\}$, and π is a proof expression with free variables in $\{\vec{x}\}$ and free proof variables in $\{p_1, \ldots, p_m\}$. We say that such a proof expression π is *valid* for judgement j under Γ. The consequence relation of proofs must satisfy:

1. (reflexivity) $p : j \vdash_{\{\vec{x}\}} p : j$ if $Fv(j) \subseteq \{\vec{x}\}$;

2. (variable weakening) $\Gamma \vdash_{\{\vec{x}\}} p : j$ and $y \notin \{\vec{x}\}$ imply $\Gamma \vdash_{\{\vec{x},y\}} p : j$;

3. (substitution of term expressions) $\Gamma \vdash_{\{\vec{x},\vec{y}\}} p : j$ implies $\Gamma\{\vec{t}/\vec{y}\} \vdash_{\{\vec{x}\}\cup fv(\vec{t})} p\{\vec{t}/\vec{y}\} : j\{\vec{t}/\vec{y}\}$, where $\Gamma\{\vec{t}/\vec{y}\}$ is $\{p_1 : j_1\{\vec{t}/\vec{y}\}, \ldots, p_m : j_m\{\vec{t}/\vec{y}\}\}$, and $fv(\vec{t}) = \bigcup_{i=1}^{n} fv(t_i)$ if $\{\vec{t}/\vec{y}\}$ denotes $\{t_1/y_1, \ldots, t_n/y_n\}$;

4. (substitution of proof expressions) $\Gamma \vdash_{\{\vec{x}\}} \pi : j$ and $\Delta \cup \{p : j\} \vdash_{\{\vec{x}\}} \sigma : k$ imply (with appropriate renaming to avoid conflicting proof variables) $\Gamma \cup \Delta \vdash_{\{\vec{x}\}} \sigma\{\pi/p\} : k$.

A consequence relation of proofs which satisfies *weakening* is a consequence relation of proofs which also satisfies

5. (weakening) $\Gamma \vdash_{\{\vec{x}\}} p : j$ and $Fv(k) \subseteq \{\vec{x}\}$ and $q \notin dom(\Gamma)$ implies $\Gamma \cup \{q : k\} \vdash_{\{\vec{x}\}} p : j$.

A consequence relation of proofs which satisfies *contraction* is a consequence relation of proofs which also satisfies

6. (contraction) $\Gamma \cup \{p_1 : j, p_2 : j\} \vdash_{\{\vec{x}\}} \pi : k$ implies $\Gamma \cup \{q : j\} \vdash_{\{\vec{x}\}} \pi\{q/p_1, q/p_2\} : j$ if $q \notin dom(\Gamma)$.

Just as in the definition of adequate encoding, we first define the notion of *strong encoding*, which extends the definition of encoding (definition 3.6), and then define when a strong encoding is natural. First, we fix some notation. Let Var^{Proof} denote the countably infinite set of proof variables of the logic, and let $P(\vec{x}, \Gamma)$ denote the set of proof expressions with free variables in the sequence of variables $\vec{x}$, and free proof variables in the sequence of proof assumptions Γ.

3.11 Definition Let *Log* be a logic specified in LF$^+$ by Σ_{Log} . A *strong* encoding $[\![\_]\!]$ of *Log* in (LF$^+$, Σ_{Log}) consists of an encoding $[\![\_]\!]$, together with a family of functions $[\![\_]\!]^P_{\vec{x};\Delta} : P(\vec{x}, \Delta) \to \mathcal{T}$, indexed by finite sequences of distinct variables $\vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n}\rangle$ and proof assumptions $\Delta = \langle p_1{:}j_1, \ldots, p_m{:}j_m\rangle$, and such that:

1. $[\![p]\!]^P_{\vec{x};\Delta} = h(p)$, where we distinguish a bijection $h : Var^{Proof} \to Var^{Judge}$;

2. the $[\![\_]\!]^P_{\vec{x};\Delta}$ are *compositional*; that is, for proof expressions $\pi \in P(\vec{y}, \Theta)$ and $\sigma_1, \ldots, \sigma_m \in P(\vec{x}, \Delta)$, and term expressions $t_1, \ldots, t_n \in T(\vec{x})$, we have

$$\begin{aligned}&[\![\pi\{\vec{t}/\vec{y}, \vec{\sigma}/\vec{p}\}]\!]^P_{\vec{x};\Delta}\\ &\quad= [\![\pi]\!]^P_{\vec{y};\Theta}\{[\![t_1]\!]_{\vec{x}}/y_1', \ldots, [\![t_n]\!]_{\vec{x}}/y_n', [\![\sigma_1]\!]^P_{\vec{x};\Delta}/h(p_1), \ldots, [\![\sigma_m]\!]^P_{\vec{x};\Delta}/h(p_m)\}\end{aligned}$$

3. the interpretation is *sound*: that is, for $\Delta = \{p_1{:}j_1, \ldots, p_m{:}j_m\}$,

$$\Delta \vdash^{Log} \pi : j \text{ implies } \Gamma_{\vec{x}}, \Gamma_\Delta \vdash^{\mathrm{LF}^+}_{\Sigma_{Log}} [\![\pi]\!]^P_{\vec{x};\Delta} : [\![j]\!]_{\vec{x}},$$

where the context $\Gamma_{\vec{x}}$ is $\langle x_1' : [\![c_1]\!], \ldots, x_n' : [\![c_n]\!]\rangle$, and the precontext Γ_Δ is $\langle h(p_1) : [\![j_1]\!]_{\vec{x}}, \ldots h(p_m) : [\![j_m]\!]_{\vec{x}}\rangle$;

Again, we sometimes omit the superscript on $[\![\_]\!]^{P}_{\vec{x};\Delta}$ when the domain is apparent.

Before we give the definition of natural representation, we require some notation. The definition uses the set of LF^{+} terms

$$\mathit{proof}^{\beta\eta}_{\Gamma} = \{p \text{ such that } \Gamma \vdash_{\Sigma} p : J : \mathit{Judge} \text{ and } p \text{ is in } \beta\eta\text{-long normal form wrt. } (\Sigma;\Gamma)\}.$$

For sequences of variables $\vec{x}$ and proof assumptions Δ, let $VP(\vec{x},\Delta)$ denote the subset of $P(\vec{x},\Delta)$ consisting of valid proof expressions. In (LF^{+}, Σ_{Log}), the valid proof expressions correspond to inhabitants of judgements; the proof expressions as a whole are not represented. We therefore restrict $[\![\_]\!]^{P}_{\vec{x};\Delta}$ to the valid proof expressions, and define the function $[\![\_]\!]^{P}_{\vec{x};\Delta} : VP(\vec{x},\Delta) \rightarrow \mathit{proof}^{\beta\eta}_{\Gamma_{\vec{x}},\Gamma_{\Delta}}$ as the function extensionally equal to $[\![\_]\!]^{P}_{\vec{x};\Delta}$, but restricted to the domain $VP(\vec{x},\Delta)$ and given with the more precise range.

3.12 Definition A strong encoding $[\![\_]\!]$ of *Log* in (LF^{+}, Σ_{Log}) is *natural* if

1. the encoding $[\![.]\!]$ of *Log* in (LF^{+}, Σ_{Log}) is an adequate encoding;
2. for finite sequences of variables $\vec{x}$ and proof assumptions Δ, the function $[\![\_]\!]_{\vec{x};\Delta} : VP(\vec{x},\Delta) \rightarrow \mathit{proof}^{\beta\eta}_{\Gamma_{\vec{x}},\Gamma_{\Delta}}$ is a bijection;
3. the interpretation is *complete*: for finite sequences of variables $\vec{x} = \langle x_1^{c_1},\ldots,x_n^{c_n}\rangle$ and proof assumptions $\Delta = \langle p_1{:}j_1,\ldots,p_m{:}j_m\rangle$, we have
$$\Gamma_{\vec{x}},\Gamma_{\Delta} \vdash_{\Sigma_{Log}} [\![p]\!]_{\vec{x};\Delta} : [\![j]\!]_{\vec{x}} \text{ implies } \Delta \vdash^{Log}_{\{\vec{x}\}} p : j.$$

We say that the logic is *naturally represented* in LF^{+} if there is a natural encoding of *Log* in (LF^{+}, Σ_{Log}).

The adequate representations of first-order logic and higher-order logic sketched in section 2.3 are also natural. To prove this, one must define a language of proof expressions for the logics and provide proof systems for deriving valid proof expressions, with the property that the valid proof expressions correspond to derivations in the logics. We state the theorem; the details can be found in [Gar92].

3.13 Theorem The signatures Σ_{Fol} and Σ_{Hol} sketched in section 2.3 provide natural representations of first-order logic and higher-order logic respectively in LF^{+}.

3.14 Example In [Gar92], we show that the LF^{+} representation of Hilbert-style S_4 discussed in example 2.11 is adequate, but not natural. The intuition behind this (without resorting to the technical detail of proof expressions) is that the number of constants in Σ_{Mod} which specify the proof system of Hilbert-style S_4 is more than the number of rules in the proof system. By the compositional property of definition 3.11, this means that the functions from valid proof expressions to inhabitants in $\beta\eta$-long normal form of judgements cannot be bijections.

Just as in the case of adequate representations, the meta-theory associated with proofs in a logic represented naturally in LF^{+} must be compatible with the meta-theory for the framework.

3.15 THEOREM Logics whose LF$^+$ representations are natural must have consequence relations of proofs which satisfy weakening and contraction.

Proof The proof follows in a similar fashion to that of theorem 3.10. $\square$

3.16 EXAMPLE Natural-deduction S_4 [Pra65] does not have a natural representation in LF$^+$ since, although its consequence relation satisfies weakening and contraction, its derivations do not define a consequence relation of proofs. The problem occurs with the Nec-rule

$$\frac{\phi}{\Box\phi} \text{ all the assumptions must be boxed}$$

which results in derivations that cannot be composed. This is illustrated (without resorting to the technical detail of proof expressions) using the derivations:

$$\frac{\Box\phi}{\Box\Box\phi}\ Nec$$

$$\frac{\phi \supset \Box\phi \qquad \phi}{\Box\phi}\ MP$$

Substituting the second derivation for the premise of the first, we obtain

$$\cfrac{\cfrac{\phi \supset \Box\phi \qquad \phi}{\Box\phi}\ Nec}{\Box\Box\phi}\ ?$$

which is not a derivation since the last line is not an instance of the Nec-rule.

Remark Our approach for studying naturality is based on that found in [HHP87]. An alternative approach is to investigate a consequence relation of *sequents* of the form

$$seq_1, \ldots, seq_n \vdash seq,$$

where seq_i for $i \in \{1, \ldots, n\}$ and seq have the form $j_1, \ldots, j_n \Rightarrow_{\vec{x}} j$ for judgements $j_1, \ldots, j_n, j$, which may contain schematic variables. Similar consequence relations have also been studied by Aczel [Acz92]. The advantage of this approach is that it captures the notion of the existence of a derivation without adapting the logic to incorporate proof expressions. The characterisation of this consequence relation in LF$^+$ is left for future research.

4 Adequate and natural representations give rise to indexed isomorphisms

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 are reinforced in this section by showing that our syntactic definition has a direct categorical formulation as an indexed isomorphism. It is known that the mathematical structure common to the logics under consideration can be captured by the structure of strict indexed categories [PS78] (or split fibrations [Ben85]), 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 give rise to indexed functors, such that adequate encodings correspond to indexed isomorphisms. This result both confirms that our approach is a natural one, and provides a link between type-theoretic and categorical approaches to frameworks. The analogous result for natural representations follows in a similar fashion: see [Gar92].

4.1 Logics and their representing type theories as indexed categories

In this section we provide the methodology for presenting logics and their representing type theories as (strict) indexed categories. For our purposes, we choose to concentrate on indexed categories rather than fibrations, since it is more natural to present a logic by considering first the syntax, which provides the indexing, and then the consequence relation. First, we require some definitions regarding indexed categories. A clear exposition of fibrations and indexed categories can be found, for example, in [BW90].

4.1 Definition Let C be a category. A *strict indexed category* is a functor $F : C^{op} \to Cat$ where Cat is the category of small categories. The category C is the *base* category and, for $c \in obj(C)$, the *fibre* over c is the category $F(c)$.

All the indexed categories discussed in this section are strict, so whenever we refer to an indexed category we assume it is strict.

4.2 Definition Let $F : A^{op} \to Cat$ and $G : B^{op} \to Cat$ be indexed categories. An *indexed functor* from F to G is a pair (σ_{base}, σ) consisting of a functor $\sigma_{base} : A \to B$ (called the *base* functor) and a natural transformation $\sigma : F \to G \circ \sigma_{base}^{op}$.

4.3 Definition An *indexed isomorphism* is an indexed functor, whose base functor is an isomorphism, and whose natural transformation is a natural isomorphism.

Our presentation of logics as indexed categories is based on the categorical presentation of various particular logics, initiated by Lawvere [Law70] but generalised to a wide class of logics. It concentrates on the abstract view of logics as consequence relations given in section 3.1.

4.4 Definition Let *Log* denote an arbitrary logic, whose consequence relation satisfies weakening and contraction. The *indexed category given by Log* is denoted by $\mathcal{L} : A^{op} \to Cat$ and defined as follows. The base category A is given by:

objects: finite sequences of the form $\langle x_1^{c_1}, \ldots, x_n^{c_n} \rangle$, where the x_i are distinct variables in Var^{Log} ;

morphisms: finite tuples of term expressions $(t_1, \ldots, t_n) : \vec{x} \to \vec{y} = \langle y_1^{c_1}, \ldots, y_n^{c_n} \rangle$ such that, for each $i \in \{1, \ldots, n\}$, the t_i and y_i inhabit the same syntactic class and $fv(t_i) \subseteq \{\vec{x}\}$;

composition: if $(t_1, \ldots, t_n) : \vec{x} \to \vec{y} = \langle y_1, \ldots, y_n \rangle$ and $(s_1, \ldots, s_m) : \vec{y} \to \vec{z}$ then $(s_1, \ldots, s_m) \circ (t_1, \ldots, t_n)$ is $(s_1\{\vec{t}/\vec{y}\}, \ldots, s_m\{\vec{t}/\vec{y}\}) : \vec{x} \to \vec{z}$;

identity: $(x_1, \ldots, x_n) : \vec{x} \to \vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n} \rangle$.

For each $\vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n} \rangle$ in $obj(A)$, the fibre $\mathcal{L}(\vec{x})$ is given by:

objects: finite sequences of judgements with free variables in $\{\vec{x}\}$;

morphisms: $\langle j_1, \ldots, j_m \rangle \to \langle k_1, \ldots, k_p \rangle$ whenever $\{j_1, \ldots, j_m\} \vdash_{\{\vec{x}\}}^{Log} k_i$ for $i \in \{1, \ldots, p\}$.

For morphism $(t_1, \ldots, t_n) : \vec{y} \to \vec{x} = \langle x_1^{c_1}, \ldots, x_n^{c_n} \rangle$ in A, the functor $\mathcal{L}((t_1, \ldots, t_n)^{op} : \vec{x} \to \vec{y}) = (\vec{t})^* : \mathcal{L}(\vec{x}) \to \mathcal{L}(\vec{y})$ is defined as follows:

$(\vec{t})^*(\langle j_1, \ldots, j_m \rangle) = \langle j_1\{\vec{t}/\vec{x}\}, \ldots, j_m\{\vec{t}/\vec{x}\} \rangle$;
$(\vec{t})^*(\langle j_1, \ldots, j_m \rangle \to \langle k_1, \ldots, k_p \rangle) = \langle j_1\{\vec{t}/\vec{x}\}, \ldots, j_m\{\vec{t}/\vec{x}\} \rangle \to \langle k_1\{\vec{t}/\vec{x}\}, \ldots, k_p\{\vec{t}/\vec{x}\} \rangle$.

This definition is shown to be valid using the properties of simultaneous substitution and the consequence relation given in section 3.1.

We do not use the standard categorical approach for presenting type theories. Our presentation is motivated by the use of the type theory as a framework for representing logics, and utilises the fact that we are able to determine in a general way that part of the type theory which corresponds to the underlying logic.

4.5 Definition Let $(\mathrm{LF}^+, \Sigma_{Log})$ be the type theory representing a logic. The *indexed category given by* $(\mathrm{LF}^+, \Sigma_{Log})$ and denoted by $\mathcal{E} : B^{op} \to Cat$ is defined as follows. The base category is given by:

objects: contexts of sorts in $\beta\eta$-long normal form;

morphisms: finite tuples of LF^+ terms $(t_1, \ldots, t_n) : \Gamma_S \to \Delta_S = \langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$, such that $\Gamma_S \vdash_{\Sigma_{Log}} t_i : A_i\{t_1, \ldots, t_{i-1}/x_1, \ldots, x_{i-1}\}$ for $i \in \{1, \ldots, n\}$, and each t_i is in $\beta\eta$-long normal form with respect to $(\Sigma_{Log}\,; \Gamma_S)$;

composition: for morphisms $(t_1, \ldots, t_n) : \Gamma_S \to \Delta_S = \langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$ and $(s_1, \ldots, s_m) : \Delta_S \to \Theta_S$, their composite $(s_1, \ldots, s_m) \circ (t_1, \ldots, t_n)$ is $(s_1\{\vec{t}/\vec{x}\}, \ldots, s_m\{\vec{t}/\vec{x}\}) : \Gamma_S \to \Theta_S$;

identity: $(x_1, \ldots, x_n) : \Delta_S \to \Delta_S = \langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$.

For each $\Gamma_S \in obj(B)$, the fibre $\mathcal{E}(\Gamma_S)$ is the preorder category given by:

objects: finite sequences of judgements $\langle J_1, \ldots, J_m \rangle$ with $J_i \in judge_{\Gamma_S}^{\beta\eta}$ for $i \in \{1, \ldots, m\}$;

morphisms: $\langle J_1,\dots,J_m\rangle \to \langle K_1,\dots,K_r\rangle$ whenever $\Gamma_S, p_1{:}J_1,\dots,p_m{:}J_m \vdash_{\Sigma_{Log}} \_ : K_j$ for $j \in \{1,\dots,r\}$, where $\_ : K_j$ denotes the inhabitation of judgement K_j.

For each morphism $(t_1,\dots,t_n) : \Delta_S \to \Gamma_S = \langle x_1{:}A_1,\dots,x_n{:}A_n\rangle$ in B, the functor $\mathcal{E} : ((t_1,\dots,t_n)^{op} : \Gamma_S \to \Delta_S) = (\vec{t})^* : \mathcal{E}(\Gamma_S) \to \mathcal{E}(\Delta_S)$ is given by:

$(\vec{t})^*(\langle J_1,\dots,J_m\rangle) = \langle J_1\{\vec{t}/\vec{x}\},\dots,J_m\{\vec{t}/\vec{x}\}\rangle$;
$(\vec{t})^*(\langle J_1,\dots,J_m\rangle \to \langle K_1,\dots,K_r\rangle) = \langle J_1\{\vec{t}/\vec{x}\},\dots,J_m\{\vec{t}/\vec{x}\}\rangle \to \langle K_1\{\vec{t}/\vec{x}\},\dots,K_r\{\vec{t}/\vec{x}\}\rangle$

The fact that this definition is valid follows from the meta-theoretic results of LF$^+$.

4.2 Adequate representations give indexed isomorphisms

We are now in a position to show the main result of this section, namely that the syntactic definition of encodings gives rise to indexed functors, with the property that the encodings are adequate if and only if the functors are isomorphisms.

4.6 Definition Assume that *Log* is an arbitrary logic, whose consequence relation satisfies weakening and contraction. Let $[\![\_]\!]$ be an encoding of *Log* in (LF$^+$, Σ_{Log}), and let the indexed categories determined by *Log* and (LF$^+$, Σ_{Log}) be $\mathcal{L} : A^{op} \to Cat$ and $\mathcal{E} : B^{op} \to Cat$ respectively. The *indexed functor determined by* $[\![.]\!]$ and denoted by $(e_{base}, e) : \mathcal{L} \to \mathcal{E}$ consists of the base functor $e_{base} : A \to B$ and natural transformation $e : \mathcal{L} \to \mathcal{E} \circ e_{base}$, where

$$e_{base}(\langle x_1^{c_1},\dots,x_n^{c_n}\rangle) = \langle x_1' : [\![\sigma_1]\!],\dots,x_n' : [\![\sigma_n]\!]\rangle;$$
$$e_{base}((t_1,\dots,t_n) : \vec{x} \to \vec{y}) = ([\![t_1]\!]_{\vec{x}},\dots,[\![t_n]\!]_{\vec{x}}) : e_{base}(\vec{x}) \to e_{base}(\vec{y}),$$

and, for each $\vec{x} \in obj(A)$,

$$e_{\vec{x}}(\langle j_1,\dots,j_n\rangle) = \langle [\![j_1]\!]_{\vec{x}},\dots,[\![j_n]\!]_{\vec{x}}\rangle;$$
$$e_{\vec{x}}(\langle j_1,\dots,j_n\rangle \to \langle k_1,\dots,k_m\rangle) = \langle [\![j_1]\!]_{\vec{x}},\dots,[\![j_n]\!]_{\vec{x}}\rangle \to \langle [\![k_1]\!]_{\vec{x}},\dots,[\![k_m]\!]_{\vec{x}}\rangle.$$

The indexed functor determined by encoding $[\![.]\!]$ is well-defined by the properties of the encoding.

Not all indexed functors give rise to encodings. For example, there is no guarantee that an indexed functor preserves the ordering or length of tuples. We believe that a more detailed analysis of the structure of these indexed categories (in particular, the categorical interpretation of sequences and contexts) will yield a two-way correspondence. This analysis is beyond the scope of this paper. We are, however, able to deduce that the indexed functor determined by an encoding is an indexed isomorphism if and only if the encoding is adequate. This strong correspondence is feasible since we are dealing with a particular indexed functor given by the encoding, which preserves the ordering and length of tuples as the following lemma states.

4.7 Lemma Assume that *Log* is an arbitrary logic, whose consequence relation satisfies weakening and contraction. Let $[\![.]\!]$ be an encoding of *Log* in (LF$^+$, Σ_{Log}) such that the indexed categories determined by *Log* and (LF$^+$, Σ_{Log}) are $\mathcal{L} : A^{op} \to Cat$ and $\mathcal{E} : B^{op} \to Cat$ respectively. Let the indexed functor $(e_{base}, e) : \mathcal{L} \to \mathcal{E}$ determined by the encoding be an indexed isomorphism with inverse $(f_{base}, f) : \mathcal{E} \to \mathcal{L}$.

1. Given
$$f_{base}((\vec{s},t,\vec{u}) : \Gamma_S \to \Delta_S) = (\vec{s}\,',t',\vec{u}\,') : f_{base}(\Gamma_S) \to f_{base}(\Delta_S);$$
$$f_{base}((\vec{v},t,\vec{w}) : \Gamma'_S \to \Delta'_S) = (\vec{v}\,',t'',\vec{w}\,') : f_{base}(\Gamma'_S) \to f_{base}(\Delta'_S),$$
where $(\vec{s},t,\vec{u})$ and $(\vec{v},t,\vec{w})$ denote two arbitrary morphisms in B containing LF$^+$ term t, we have
   (a) the lengths of Γ_S and $f_{base}(\Gamma_S)$, and of Δ_S and $f_{base}(\Delta_S)$ are the same;
   (b) the lengths of $\vec{s}$ and $\vec{s}'$, and of $\vec{u}$ and $\vec{u}\,'$ are the same;
   (c) $t' = t''$.
2. For each $\Gamma_S \in obj(B)$, given
$$f_{\Gamma_S}(\langle \vec{J},K,\vec{L}\rangle) = \langle \vec{J}',K',\vec{L}'\rangle;$$
$$f_{\Gamma_S}(\langle \vec{M},K,\vec{N}\rangle) = \langle \vec{M}',K'',\vec{N}'\rangle,$$
where $\langle \vec{J},K,\vec{L}\rangle$ and $\langle \vec{M},K,\vec{N}\rangle$ denote two arbitrary objects of $\mathcal{E}(\Gamma_S)$ containing $K \in judge^{\beta\eta}_{\Gamma_S}$, we have:
   (a) the lengths of $\vec{J}$ and $\vec{J}'$, and of $\vec{L}$ and $\vec{L}'$ are the same;
   (b) $K' = K''$.

Proof (Sketch) By the definition of (e_{base}, e) we know that the functor e_{base} and, for all $\vec{x} \in obj(A)$, the functors $e_{\vec{x}}$ preserve the order and length of sequences and tuples. This yields parts 1a, 1b and 2a. Parts 1c and 2b follow from that fact that (f_{base}, f) is inverse to (e_{base}, e). □

We are now in a position to show the main result of this section, namely that adequate encodings correspond to indexed isomorphisms.

4.8 Theorem Assume that Log is an arbitrary logic, whose consequence relation satisfies weakening and contraction. Let $[\![.]\!]$ be an encoding of Log in (LF$^+$, Σ_{Log}), and let $(e_{base}, e) : \mathcal{L} \to \mathcal{E}$ be the indexed functor determined by $[\![.]\!]$, where $\mathcal{L} : A^{op} \to Cat$ and $\mathcal{E} : B^{op} \to Cat$. Then $[\![.]\!]$ is adequate if and only if (e_{base}, e) is an indexed isomorphism.

Proof (Sketch) First, assume that $[\![.]\!]$ is an adequate encoding. Let $f : Var^{Sort} \to Var^{Log}$ denote the inverse of $(\_)' : Var^{Log} \to Var^{Sort}$, and consider the function $[\_] : sort^{\beta\eta}_{\langle\,\rangle} \to S$, and functions $[\_]_{\Gamma_S} : texp^{\beta\eta}_{\Gamma_S} \to T(\vec{x}_{\Gamma_S})$ and $[\_]_{\Gamma_S} : judge^{\beta\eta}_{\Gamma_S} \to J(\vec{x}_{\Gamma_S})$, for each context of sorts Γ_S in $\beta\eta$-long normal form, which are inverse to $[\![\_]\!]^S$, and to $[\![\_]\!]^T_{\vec{x}_{\Gamma_S}}$ and $[\![\_]\!]^J_{\vec{x}_{\Gamma_S}}$ respectively, where if Γ_S is $\langle x_1{:}A_1, \ldots, x_n{:}A_n\rangle$ then $\vec{x}_{\Gamma_S}$ is the sequence $\langle f(x_1)^{[A_1]}, \ldots, f(x_n)^{[A_n]}\rangle$. We use these functions to define an indexed functor $(f_{base}, f) : \mathcal{E} \to \mathcal{L}$ which is the inverse indexed functor of (e_{base}, e).

The base functor f_{base} is given as follows:

$$f_{base}(\langle x_1{:}A_1, \ldots, x_n{:}A_n\rangle) = \langle f(x_1)^{[A_1]}, \ldots, f(x_n)^{[A_n]}\rangle,$$
$$f_{base}((t_1, \ldots, t_m) : \Gamma_S \to \Delta_S) = ([t_1]_{\Gamma_S}, \ldots, [t_m]_{\Gamma_S}) : f_{base}(\Gamma_S) \to f_{base}(\Delta_S),$$

and, for each context of sorts Γ_S in $\beta\eta$-long normal form, the natural transformation $f : \mathcal{E} \to \mathcal{L} \circ f_{base}$ is defined, for each $\Gamma_S \in obj(B)$, by

$$f_{\Gamma_S}(\langle J_1, \ldots, J_m \rangle) = \langle [J_1]_{\Gamma_S}, \ldots, [J_m]_{\Gamma_S} \rangle;$$
$$f_{\Gamma_S}(\langle J_1, \ldots, J_m \rangle \to \langle K_1, \ldots, K_p \rangle) = \langle [J_1]_{\Gamma_S}, \ldots, [J_m]_{\Gamma_S} \rangle \to \langle [K_1]_{\Gamma_S}, \ldots, [K_p]_{\Gamma_S} \rangle.$$

That (f_{base}, f) provides an indexed functor from $\mathcal{E}$ to $\mathcal{L}$ follows from the conditions satisfied by $[\_]$, $[\_]_{\Gamma_S}$ and $[\_]_{\Gamma_S}$ for $\Gamma_S \in obj(B)$. The proof that the indexed functor (f_{base}, f) is inverse to (e_{base}, e) is technical, but not difficult, and uses the fact that, for each $\vec{x} \in obj(A)$, the functions $[\_]$, $[\_]_{\Gamma_{\vec{x}}}$ and $[\_]_{\Gamma_{\vec{x}}}$ are inverse to $[\![\_]\!]^S$, $[\![\_]\!]^T_{\vec{x}}$ and $[\![\_]\!]^J_{\vec{x}}$ respectively. The details can be found in [Gar92].

Now assume that (e_{base}, e) is an indexed isomorphism. We show that encoding $[\![.]\!]$ is adequate. Let $(f_{base}, f) : \mathcal{E} \to \mathcal{L}$ be the inverse indexed functor of (e_{base}, e). Define $[\_] : sort^{\beta\eta}_{\langle\,\rangle} \to S$ and, for each $\Gamma_S \in obj(B)$, the functions $[\_]_{\Gamma_S} : texp^{\beta\eta}_{\Gamma_S} \to T$ and $[\_]_{\Gamma_S} : judge^{\beta\eta}_{\Gamma_S} \to J$ as follows:

1. $[A] = c$ for each $A \in sort^{\beta\eta}_{\langle\,\rangle}$, where $f_{base}(\langle x : A \rangle) = \langle y^c \rangle$;
2. $[t]_{\Gamma_S} = t'$ for each $t \in sort^{\beta\eta}_{\Gamma_S}$, where $f_{base}((x_1, \ldots, x_n, t) : \Gamma_S \to \Gamma_S, x : A) = (y_1, \ldots, y_n, t') : f_{base}(\Gamma_S) \to f_{base}(\Gamma_S, x : A)$;
3. $[j]_{\Gamma_S} = j'$ for each $j \in judge^{\beta\eta}_{\Gamma_S}$, where $f_{\Gamma_S}(\langle j \rangle) = \langle j' \rangle$.

It is technical, but not difficult, to show that the functions $[\_]$, $[\_]_{\Gamma_S}$ and $[\_]_{\Gamma_S}$, for each $\Gamma_S \in obj(B)$, are well-defined and are inverse to the functions $[\![\_]\!]^S$, $[\![\_]\!]^T_{\vec{x}_{\Gamma_S}}$ and $[\![\_]\!]^J_{\vec{x}_{\Gamma_S}}$ respectively, where if Γ_S is $\langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$, then $\vec{x}_{\Gamma_S}$ is $\langle f(x_1)^{[\![A_1]\!]}, \ldots, f(x_n)^{[\![A_n]\!]} \rangle$. This result, plus the completeness condition in definition 3.7, are proved using lemma 4.7. Again, the details can be found in [Gar92]. □

The analogous result to theorem 4.8 for natural representations, see [Gar92], follows in a similar fashion by adapting the indexed categories determined by the logic and its representing type theory to give an explicit account of the derivations of the logic and the terms inhabiting LF$^+$ judgements, and then showing that natural representations give rise to indexed isomorphisms.

5 Concluding Remarks

We have advocated the need for general definitions to describe how well a logic has been represented in a logical framework. Based on ideas from [HHP87], the new framework LF$^+$ is introduced in order to provide such definitions. Two definitions are given: *adequate* representation, which defines when the consequence relation of a logic has been well-represented in the LF$^+$ entailment relation, and *natural* representation, which provides some measure that the derivations of the logic have been well-represented. Our arguments are reinforced by showing that these syntactic definitions have a simple formulation as indexed isomorphisms.

Other definitions of 'correct' representation should be explored. For example, our approach for studying naturality is based on that found in [HHP87]. An alternative approach is to investigate the representation of a consequence relation of *sequents*, which may contain schematic variables. Similar consequence relations have also been studied by Aczel [Acz92]. One advantage of this consequence relation is that it captures the notion of the existence of derivations without adapting the logic. This approach should also lead to weaker notions of naturality. The investigation of this consequence relation and its representation in LF^{+} is left for future research.

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

References

[Acz92] P. Aczel. Schematic Consequence, *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, Båstad, 1992.

[ACN90] L. Augustsson, T. Coquand and B. Nordström. A Short Description of Another Logical Framework, *Proceedings of the First Workshop on Logical Frameworks*, ed.s Huet and Plotkin, pp 39–42, 1990.

[AHMP92] A. Avron, F. Honsell, I.A. Mason and R. Pollack. Using Typed Lambda Calculus to Implement Formal Systems on a Machine, *Journal of Automated Reasoning*, Vol. 9, pp 309–354, 1992.

[Avr91] A. Avron. Simple Consequence Relations, *Journal of Information and Computation*, Vol. 92, pp 105-139, 1991.

[Bar92] H. Barendregt. Lambda Calculi with Types, *Handbook of Logic in Computer Science*, Oxford University Press, Vol. 2, pp 117–309, 1992.

[Ben85] J. Bénabou. Fibred Categories and the Foundations of Naive Category Theory, *Journal of Symbolic Logic*, Vol. 50, pp 10-37, 1985.

[Ber90] S. Beradi. *Type dependence and constructive mathematics*, Ph.D. thesis, Mathematical Institute, Torino, 1990.

[BW90] M. Barr and C. Wells, *Category Theory for Computing Science*, Prentice Hall, 1990.

[Bru80] N.G. de Bruijn. A Survey of the Project Automath, in [SH80], pp 589–606, 1980.

[Che80] B.F. Chellas. *Modal Logic: an introduction*, Cambridge University Press, 1980.

[Chu40] A. Church. A formulation of the simple theory of types, *Journal of Symbolic Logic*, Vol. 5, pp 56-68, 1940.

[Con86] R.L. Constable et al. *Implementing Mathematics with the NuPrl Proof Development System*, Prentice-Hall, 1986.

[CF58] H.B. Curry and R. Feys. *Combinatory Logic*, North Holland, 1958.

[DHW93] G. Dowek, G. Huet and B. Werner. On the Definition of the η-long normal form in Type Systems of the Cube, *Proceedings of the 1993 Workshop on Types for Proofs and Programs*, Nijmejen, 1993.

[Dun84] J.M. Dunn. Relevant Logic and Entailment, *Handbook of Philosophical Logic*, eds. D. Gabbay and F. Guenthner, Vol. 3, pp 117–224, 1984.

[Gar92] P.A. Gardner. *Representing Logics in Type Theory*, Ph.D. Thesis, Technical Report ECS-LFCS-92-227, Edinburgh University, 1992.

[Gar93] P.A. Gardner. A New Type Theory for Representing Logics, *LNAI 698: Proceedings of the Fourth International Conference on Logic Programming and Automated Reasoning*, Springer-Verlag, ed. Voronkov, pp 146–157, 1993.

[Gar93a] P.A.Gardner. *The Construction of $\beta\eta$-long normal forms in Dependent Type Theories*, manuscript, 1993a.

[Geu92] J.H. Geuvers. The Church–Rosser Property for $\beta\eta$-reduction in Typed Lambda Calculi, *Proceedings of the Seventh Annual Symposium on Logic in Computer Science*, pp 453–460, 1992.

[Gir87] J.Y. Girard. Linear Logic, *Journal of Theoretical Computer Science*, Vol. 50, pp 1-102, 1987.

[Gor87] 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.

[HHP87] R. Harper, F. Honsell and G. Plotkin. A Framework for Defining Logics, *Journal of the Association for Computing Machinery*, Vol. 40, Part 1, pp 143-184, 1992. Preliminary version in the *Proceedings of the Second Annual Symposium on Logic in Computer Science*, pp 194-204, 1987.

[HST89] R. Harper, D. Sannella and A. Tarlecki. *Structure and Representation in LF*. Technical Report ECS-LFCS-89-75, Edinburgh University, 1989. Preliminary version in the *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pp 226-237, 1989.

[How80] W.A. Howard. The Formulae-as-types Notion of Construction, in [SH80], pp 479–490, 1980.

[Law70] F.W. Lawvere. Equality in Hyperdoctrines and Comprehension Schema as an Adjoint Functor, *Applications of Categorical Algebra*, American Mathematical Society, pp 1–14, 1970.

[LP92] Z. Luo and R. Pollack. *LEGO Proof Development System: User's Manual*, Technical Report ECS-LFCS-92-211, Edinburgh University, 1992.

[Mar85] P. Martin-Löf. *On the Meanings of the Logical Constants and the Justifications of the Logical Laws*, Technical Report 2, Università di Siena, 1985.

[MN86] D. Miller and G. Nadathur. Higher-order Logic Programming, *LNCS 225: Proceedings of the Third International Logic Programming Conference,* Springer-Verlag, ed. Shapiro, pp 448–462, 1986.

[MPP92] D. Miller, G. Plotkin and D. Pym *A Relevant Analysis of Natural Deduction*, in preparation; talk at the 1992 Workshop on Types for Proofs and Programs, Båstad, 1992.

[PS78] 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.

[Pau87] L. Paulson. The Foundations of a Generic Theorem Prover, *Journal of Automatic Reasoning*, Vol. 5, pp 363 – 397, 1987.

[Pra65] D. Prawitz. *Natural Deduction: A Proof-theoretical study*, Almquist and Wiksell, Stockholm, 1965.

[Pym92] D. Pym. A Unification Algorithm for the $\lambda\Pi$-calculus, *Journal of the Foundations of Computer Science*, Vol. 3, No. 3, pp 333–378, 1992.

[Sal90] A. Salvesen. The Church-Rosser Property for LF with $\beta\eta$-reduction, talk at the *First Workshop on Logical Frameworks*, Båstad, 1990.

[Sal91] A. Salvesen. *The Church-Rosser Property for Pure Type Systems with $\beta\eta$-reduction*, submitted for journal publication, 1991.

[SH80] J.P. Seldin and J.R. Hindley, editors. *To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, 1980.

[Sim92] A. Simpson. Kripke Semantics for a Logical Framework, *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, Båstad, 1992.

[Tar56] A. Tarski. *Logic, Semantics and Metamathematics*, Oxford University Press, 1956.

[Ter89] J. Terlouw. *Een nadere bewijstheoretishe analyse van GSTT's*, Internal Report, Faculty of Mathematics and Computer Science, University of Nijmegen, 1989.